The 54th Annual Meeting of the Association for Computational Linguistics

Proceedings of the Conference, Vol. 1 (Long Papers)

August 7-12, 2016
Berlin, Germany

Platinum Sponsors

Gold Sponsors

Silver Sponsors

Bronze Sponsors

Order copies of this and other ACL proceedings from:

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

ISBN 978-1-945626-00-5 (Volume 1)
ISBN 978-1-945626-01-2 (Volume 2)

Preface: General Chair

In my welcome to participants in this year's conference handbook, I especially welcomed those for which it was their first ACL. I expressed the hope that the conference fulfilled their expectations and remained in their memory as a great start. Trying to imagine the first experience of a present-day ACL, the magnitude of the whole event may be a bit overwhelming - our field is on an expanding trajectory, and even a selection of the best work fills a great number of parallel sessions over a number of days; plus, there are the workshops and tutorials to quench many topical thirsts. This ACL again promises to be a next peak in a progressive development.

ACL Conferences are the product of many people working together, kindly offering their services to the community at large. ACL-2016 is no exception to this. I would like to thank each and every person who has volunteered their time to make the event possible. I am deeply impressed with the sense of community that organizing an ACL brings about.

Priscilla Rasmussen, the ACL Business Manager, and the 2015 ACL Executive Committee (Chris Manning, Pushpak Bhattacharyya, Joakim Nivre, Graeme Hirst, Dragomir Radev, Gertjan van Nood, Min-Yen Kan, Herman Ney, and Yejin Choi) have been instrumental in setting ACL-2016 in motion and in guiding the ACL-2016 team along the path from concept to execution. Without the collective memory and hands-on guidance of the committee, an ACL conference will never happen.

The ACL-2016 team was formidable in building all the components of the conference and connecting them together in an impressive programme: Katrin Erk and Noah Smith (Programme Committee Chairs); Valia Kordoni, Markus Egg (Local Arrangements Chairs) who brought together a fantastic local organization team; Sabine Schulte im Walde and Jun Zhao (Workshop Chairs), Alexandra Birch and Willem Zuidema (Tutorial Chairs); Hai Zhao, Yusuke Miyao, and Yannick Versley (Publication Chairs); Tao Lei, He He, and Will Roberts (Student Research Workshop Chairs), Yang Liu, Chris Biemann, and Gosse Bouma (Faculty Advisors for the Student Research Workshop), Marianna Apidianaki and Sameer Pradhan (Demonstration Chairs), Barbara Plank (Publicity Chair), Florian Kunneman and Matt Post (Conference Handbook Team), and Yulia Grishina (Student Volunteer Coordinator).

The Program Chairs selected outstanding invited speakers: Mark Steedman (University of Edinburgh) and Amber Boydstun (University of California, Davis).

I am deeply grateful to our sponsors for their generous contributions, allowing the conference not to become prohibitively expensive: Google, Baidu, Amazon (Platinum Sponsors); Bloomberg, Facebook, eBay, Elsevier, Microsoft Research, and Maluuba (Gold Sponsors); Huawei Technologies, Zalando SE (Silver Sponsors); Nuance, Grammarly, Voicebox, Yandex, and Textkernel (Bronze Sponsors).

Finally, I would like to express my deep appreciation for the hard work carried out by all area chairs, workshop organizers, tutorial presenters, and the massive army of reviewers. Kudos to all.

Welcome to ACL-2016!

Antal van den Bosch
General Chair

Preface: Program Committee Co-Chairs

Welcome to the 54th Annual Meeting of the Association for Computational Linguistics! This year, ACL received 825 long paper submissions (a new record) and 463 short paper submissions.[1] Of the long papers, 231 were accepted for presentation at ACL—116 as oral presentations and 115 as poster presentations. 97 short papers were accepted—49 as oral and 48 as poster presentations. In addition, ACL also features 25 presentations of papers accepted in the *Transactions of the Association for Computational Linguistics* (TACL). With 353 paper presentations at the main conference, this is the largest ACL program to date.

In keeping with the tremendous growth of our field, we introduced some changes to the conference. Oral presentations were shortened to fifteen (twelve) minutes for long (short) papers, plus time for questions. While this places a greater demand on speakers to be concise, we believe it is worth the effort, allowing far more work to be presented orally. We also took advantage of the many halls available at Humboldt University and expanded the number of parallel talks during some conference sessions.

We introduced a category of outstanding papers to help recognize the highest quality work in the community this year. The 11 outstanding papers (9 long, 2 short, 0.85% of submissions) represent a broad spectrum of exciting contributions; they are recognized by especially prominent placement in the program. From these, a best paper and an IBM-sponsored best student paper have been selected; those will be announced in the awards session on Wednesday afternoon.

Following other recent ACL conferences, submissions were reviewed under different categories and using different review forms for empirical/data-driven, theoretical, applications/tools, resources/evaluation, and survey papers. We introduced special fields in the paper submission form for authors to explicitly note the release of open-source implementations to enable reproducibility, and to note freely available datasets. We also allowed authors to submit appendices of arbitrary length for details that would enable replication; reviewers were not expected to read this material.

Another innovation we explored during the review period was the scheduling of short paper review before long paper review. While this was planned to make the entire review period more compact (fitting between the constraints of NAACL 2016 and EMNLP 2016 at either end), we found that reviewing short papers first eliminated many of the surprises for the long paper review process.

We sought to follow recently-evolved best practices in planning the poster sessions, so that the many high-quality works presented in that format will be visible and authors and attendees benefit from the interactions during the two poster sessions.

ACL 2016 will have two distinguished invited speakers: Amber Boydstun (Associate Professor of Political Science at the University of California, Davis) and Mark Steedman (Professor of Cognitive Science at the University of Edinburgh). We are grateful that they accepted our invitations and look forward to their presentations.

There are many individuals we wish to thank for their contributions to ACL 2016, some multiple times:

[1] These numbers exclude papers that were not reviewed due to formatting, anonymity, or double submission violations (9 short and 21 long papers) or that were withdrawn prior to review (approximately 59 short and 52 long papers).

- The 38 area chairs who recruited reviewers, led the discussion about each paper, carefully assessed each submission, and authored meta-reviews to guide final decisions: Miguel Ballesteros, David Bamman, Steven Bethard, Jonathan Berant, Gemma Boleda, Ming-Wei Chang, Wanxiang Che, Chris Dyer, Ed Grefenstette, Hannaneh Hajishirzi, Minlie Huang, Mans Hulden, Heng Ji, Jing Jiang, Zornitsa Kozareva, Marco Kuhlmann, Yang Liu, Annie Louis, Wei Lu, Marie-Catherine de Marneffe, Gerard de Melo, David Mimno, Meg Mitchell, Daichi Mochihashi, Graham Neubig, Naoaki Okazaki, Simone Ponzetto, Matthew Purver, David Reitter, Nathan Schneider, Hinrich Schuetze, Thamar Solorio, Lucia Specia, Partha Talukdar, Ivan Titov, Lu Wang, Nianwen Xue, and Grace Yang.
- Our full program committee of 884 hard-working individuals who reviewed the conference's 1,288 submissions (including secondary reviewers).
- The ACL coordinating committee members, especially Yejin Choi, Graeme Hirst, Chris Manning, and Shiqi Zhao, who answered many questions as they arose during the year.
- TACL editors-in-chief Mark Johnson, Lillian Lee, and Kristina Toutanova, for coordinating with us on TACL presentations at ACL.
- Ani Nenkova and Owen Rambow, program co-chairs of NAACL 2016, and Michael Strube, program co-chair of ACL 2015, who were generous with advice.
- Yusuke Miyao, Yannick Versley, and Hai Zhao, our well-organized publication chairs, and the responsive team at Softconf led by Rich Gerber.
- Valia Kordoni and the local organization team, especially webmaster Kostadin Cholakov.
- Antal van den Bosch, our general chair, who kept us coordinated with the rest of the ACL 2016 team and offered guidance whenever we needed it.
- Antal van den Bosch, Claire Cardie, Pascale Fung, Ray Mooney, and Joakim Nivre, who carefully reviewed papers under consideration for outstanding and best paper recognition.
- Priscilla Rasmussen, who knows everything about how to make ACL a success.

We hope that you enjoy ACL 2016 in Berlin!

ACL 2016 program co-chairs
Katrin Erk, University of Texas
Noah A. Smith, University of Washington

ACL 2016 Organizing Committee

General Chair

Antal van den Bosch, Radboud University Nijmegen

Program Co-Chairs

Katrin Erk, University of Texas at Austin
Noah A. Smith, University of Washington

Local Organizing Committee

Valia Kordoni (local chair), Humboldt University
Markus Egg (local co-chair), Humboldt University
Kostadin Cholakov, Humboldt University
Maja Popović, Humboldt University
Manfred Stede, University of Potsdam
Anke Lüdeling, Humboldt University
Manfred Krifka, Humboldt University
Ulf Leser, Humboldt University
Yulia Grishina (Student Volunteer Coordinator), University of Potsdam

Workshop Chairs

Jun Zhao, Chinese Academy of Sciences
Sabine Schulte im Walde, IMS Stuttgart

Tutorial Chairs

Alexandra Birch, University of Edinburgh
Willem Zuidema, University of Amsterdam

Publication Chairs

Yannick Versley, Ruprecht-Karls-Universität Heidelberg
Hai Zhao, Shanghai Jiao Tong University
Yusuke Miyao, National Institute of Informatics, Japan

Demonstration Chairs

Sameer Pradhan, Boulder Learning
Marianna Apidianaki, LIMSI-CNRS

Student Research Workshop Chairs

Will Roberts, Humboldt-Universität zu Berlin
Tao Lei, Massachusetts Institute of Technology
He He, University of Maryland

Faculty Advisors to the Student Research Workshop

Yang Liu, Tsinghua University

Chris Biemann, Technische Universität Darmstadt
Gosse Bouma, University of Groningen

Student Volunteer Coordinator

Yulia Grishina, University of Potsdam

Publicity Chair

Barbara Plank, University of Groningen

Conference Handbook Chair

Florian Kunneman, Radboud University Nijmegen

Business Manager

Priscilla Rasmussen, ACL

Professional Conference Organiser

Con gressa GmbH, Berlin

ACL 2016 Program Committee

Omri Abend, Steven Abney, Amjad Abu-Jbara, Heike Adel, Stergos Afantenos, Nitish Aggarwal, Željko Agić, Eneko Agirre, Cem Akkaya, Iñaki Alegria, Laura Alonso Alemany, Enrique Alfonseca, Afra Alishahi, Alexandre Allauzen, David Alvarez-Melis, Waleed Ammar, Jacob Andreas, Ion Androutsopoulos, Gabor Angeli, Masahiro Araki, Ron Artstein, Yoav Artzi, Kristjan Arumae, Nicholas Asher, AiTi Aw,

JinYeong Bak, Collin Baker, Alexandra Balahur, Timothy Baldwin, Borja Balle, Miguel Ballesteros, David Bamman, Srinivas Bangalore, Mohit Bansal, Libby Barak, Marco Baroni, Regina Barzilay, Roberto Basili, David Batista, Timo Baumann, Frederic Bechet, Daniel Beck, Núria Bel, I. Beltagy, Anja Belz, Fabrício Benevenuto, Jonathan Berant, Taylor Berg-Kirkpatrick, Sabine Bergler, Raffaella Bernardi, Robert Berwick, Steven Bethard, Archna Bhatia, Pushpak Bhattacharyya, Klinton Bicknell, Ann Bies, Alexandra Birch, Anders Björkelund, Frédéric Blain, Nate Blaylock, Bernd Bohnet, Ondřej Bojar, Ester Boldrini, Gemma Boleda, Danushka Bollegala, Claire Bonial, Georgeta Bordea, Antoine Bordes, Johan Bos, Samuel R. Bowman, Johan Boye, Kristy Boyer, Cem Bozsahin, S.R.K. Branavan, António Branco, Adrian Brasoveanu, Elia Bruni, Christian Buck, Paul Buitelaar, Harry Bunt, Hendrik Buschmeier,

Aoife Cahill, Lynne Cahill, José Camacho-Collados, Burcu Can, Nicola Cancedda, Hailong Cao, Kris Cao, Yunbo Cao, Claire Cardie, Xavier Carreras, John Carroll, Paula Carvalho, Francisco Casacuberta, Helena Caseli, Tommaso Caselli, Steve Cassidy, Irene Castellon, Giuseppe Castellucci, Asli Celikyilmaz, Özlem Çetinoğlu, Mauro Cettolo, Arun Chaganty, Hou-pong Chan, Jane Chandlee, Angel Chang, Kai-Wei Chang, Ming-Wei Chang, Wanxiang Che, Ciprian Chelba, Berlin Chen, Boxing Chen, Chen Chen, Hsin-Hsi Chen, Kehai Chen, Kuan-Yu Chen, Qian Chen, Wenliang Chen, Xixian Chen, Yun-Nung Chen, Zhiyu Chen, Colin Cherry, Sean Chester, Jackie Chi Kit Cheung, David Chiang, Hai Leong Chieu, Laura Chiticariu, Kyunghyun Cho, Eunsol Choi, Yejin Choi, Janara Christensen, Christos Christodoulopoulos, Grzegorz Chrupała, Alexander Clark, Stephen Clark, Ann Clifton, Maximin Coavoux, Moreno Coco, Trevor Cohn, Sandra Collovini, Miriam Connor, John Conroy, Paul Cook, Ann Copestake, Ryan Cotterell, Benoit Crabbé, Danilo Croce, James Cross, Montse Cuadros, Heriberto Cuayahuitl, Silviu-Petru Cucerzan, Xiaodong Cui, Jennifer Culbertson, Aron Culotta,

Walter Daelemans, Robert Daland, Amitava Das, Dipanjan Das, Hal Daumé III, Brian Davison, Munmun De Choudhury, Adrià de Gispert, Marie-Catherine de Marneffe, Gerard de Melo, Claudio Delli Bovi, Vera Demberg, Pascal Denis, Michael Denkowski, Leon Derczynski, Aliya Deri, Nina Dethlefs, Ann Devitt, Jacob Devlin, Giuseppe Di Fabbrizio, Mona Diab, Brian Dillon, Georgiana Dinu, Dmitriy Dligach, Simon Dobnik, Ellen Dodge, A. Seza Doğruöz, Doug Downey, Eduard Dragut, Mark Dras, Markus Dreyer, Gregory Druck, Lan Du, Kevin Duh, Jesse Dunietz, Long Duong, Nadir Durrani, Greg Durrett, Chris Dyer, Marc Dymetman,

Koji Eguchi, Patrick Ehlen, Vladimir Eidelman, Andreas Eisele, Jacob Eisenstein, Jason Eisner, Michael Elhadad, Micha Elsner, Nikolaos Engonopoulos, Katrin Erk, Miquel Esplà-Gomis, Paula Estrella, Keelan Evanini,

Federico Fancellu, Rui Fang, Manaal Faruqui, Adam Faulkner, Benoit Favre, Anna Feldman, Naomi Feldman, Raquel Fernandez, Daniel Fernández-González, Simone Filice, Katja Filippova, Orhan Firat, Nicholas FitzGerald, Margaret Fleck, Fabian Flöck, Enrique Flores, Evandro Fonseca, Mikel Forcada, Markus Forsberg, George Foster, James Foulds, Bob Frank, Michael Frank, Stefan L. Frank, Stella Frank, Alexander

Fraser, Dayne Freitag, Annemarie Friedrich, Akinori Fujino,

Yarin Gal, Michel Galley, Carlos Gallo, Björn Gambäck, Michael Gamon, Kuzman Ganchev, Kavita Ganesan, Juri Ganitkevitch, Jianfeng Gao, Wei Gao, Miguel A. García-Cumbreras, Claire Gardent, Matt Gardner, Dan Garrette, Milica Gasic, Kallirroi Georgila, Daniel Gildea, Kevin Gimpel, Maite Giménez, Jonathan Ginzburg, Dimitra Gkatzia, Goran Glavaš, Amir Globerson, Yoav Goldberg, Dan Goldwasser, Sharon Goldwater, Carlos Gómez-Rodríguez, Edgar González Pellicer, Matthew R. Gormley, Cyril Goutte, Joao Graca, Yvette Graham, Roger Granada, Edouard Grave, Nancy Green, Spence Green, Ed Grefenstette, Denis Griffis, Tom Griffiths, Liane Guillou, Curry Guinn, James Gung, Jiang Guo, Weiwei Guo, Yuhong Guo, Abhijeet Gupta, Iryna Gurevych, Yoan Gutierrez, Kelvin Guu,

Nizar Habash, Gholamreza Haffari, Hannaneh Hajishirzi, Dilek Hakkani-Tur, John Hale, David Hall, Keith Hall, William L. Hamilton, Michael Hammond, Bo Han, Greg Hanneman, Christian Hardmeier, Sasa Hasan, Helen Hastie, Claudia Hauff, Katsuhiko Hayashi, Daqing He, Luheng He, Xiaodong He, Yifan He, Kenneth Heafield, Carmen Heger, Michael Heilman, James Henderson, Matthew Henderson, Aurélie Herbelot, Karl Moritz Hermann, Daniel Hershcovich, Jack Hessel, Felix Hill, Keikichi Hirose, Graeme Hirst, Julian Hitschler, Anna Hjalmarsson, Kai Hong, Liangjie Hong, Andrea Horbach, Julian Hough, Dirk Hovy, Bo-June (Paul) Hsu, Liang Huang, Minlie Huang, Ruihong Huang, Mans Hulden, Tim Hunter, Jena D. Hwang,

Ignacio Iacobacci, Ryu Iida, Diana Inkpen, Naoya Inoue, Hitoshi Isahara, Hideki Isozaki, Mohit Iyyer,

Cassandra L. Jacobs, Guillaume Jacquet, Evan Jaffe, Jagadeesh Jagarlamudi, Zunaira Jamil, Peter Jansen, Laura Jehl, Yacine Jernite, Heng Ji, Yangfeng Ji, Sittichai Jiampojamarn, Jing Jiang, Wenbin Jiang, Xin Jiang, Salud María Jiménez Zafra, Anders Johannsen, Richard Johansson, Michael Johnston, Kristiina Jokinen, Arne Jonsson, Aditya Joshi, Shafiq Joty, Dan Jurafsky, Filip Jurcicek, David Jurgens,

Mijail Kabadjov, Nobuhiro Kaji, Jaap Kamps, Min-Yen Kan, Hiroshi Kanayama, Mladen Karan, Tatsuya Kawahara, Frank Keller, Casey Kennington, Chloé Kiddon, Douwe Kiela, Jae Won Kim, Joon Kim, Jooyeon Kim, Suin Kim, Sunghwan Kim, Yoon Kim, Young-Bum Kim, Irwin King, Brian Kingsbury, Svetlana Kiritchenko, Julia Kiseleva, Alexandre Klementiev, Sigrid Klerke, Christoph Kling, Julien Kloetzer, Alistair Knott, Tomáš Kočiský, Philipp Koehn, Oskar Kohonen, Prasanth Kolachina, Alexander Koller, Mamoru Komachi, Kazunori Komatani, Rik Koncel-Kedziorski, Lingpeng Kong, Ioannis Konstas, Hahn Koo, Stefan Kopp, Moshe Koppel, Valia Kordoni, Anna Korhonen, Bhushan Kotnis, Zornitsa Kozareva, Julia Kreutzer, Jayant Krishnamurthy, Mark Kroell, Kriste Krstovski, Canasai Kruengkrai, Germán Kruszewski, Lun-Wei Ku, Sicong Kuang, Marco Kuhlmann, Jonas Kuhn, Wu Kui, Shankar Kumar, Nate Kushman, Ilia Kuznetsov, Polina Kuznetsova, Tom Kwiatkowski, Sandra Kübler,

Man Lan, Guy Lapalme, Mirella Lapata, Shalom Lappin, Birger Larsen, Alex Lascarides, Alon Lavie, Angeliki Lazaridou, Phong Le, Joseph Le Roux, Kenton Lee, Moontae Lee, Sungjin Lee, Yoong Keok Lee, Tao Lei, Oliver Lemon, Alessandro Lenci, Lori Levin, Gina-Anne Levow, Omer Levy, Mike Lewis, Chen Li, Fangtao Li, Hang Li, Jing Li, Jiwei Li, Junhui Li, Shoushan Li, Sujian Li, Wenjie Li, Xiaolong Li, Zhenghua Li, Kexin Liao, Xiangwen Liao, Chin-Yew Lin, Hui Lin, Victoria Lin, Wang Ling, Xiao Ling, Christina Lioma, Pierre Lison, Diane Litman, Marina Litvak, Biao Liu, Fei Liu, Jiangming Liu, Jing Liu, Kang Liu, Qun Liu, Yang Liu, Varvara Logacheva, Lucelene Lopes, Adam Lopez, Oier Lopez de Lacalle, Aurelio Lopez-Lopez, Annie Louis, Bin Lu, Jing Lu, Wei Lu, Andy Luecking, Michal Lukasik, Jiaming Luo, Minh-Thang Luong, Franco M. Luque,

Shaoping Ma, Qingsong Ma, Shuming Ma, Wolfgang Macherey, Nitin Madnani, Wolfgang Maier, Prodromos Malakasiotis, Suresh Manandhar, Gideon Mann, Christopher D. Manning, Diego Marcheggiani, Daniel Marcu, Marco Marelli, Eugenio Martínez Cámara, Toni Marti, Patricio Martinez-Barco, Héctor Martínez Alonso, Eugenio Martínez Cámara, Fernando Martínez Santiago, Eugenio Martínez-Cámara, Fernando Javier Martínez Santiago, André F. T. Martins, Vivien Mast, Yuichiro Matsubayashi, Takuya Matsuzaki, Jonathan May, Diana Maynard, Diana McCarthy, David McClosky, Ryan McDonald, Tara McIntosh, Kathy McKeown, Louise McNally, Beata Megyesi, Yashar Mehdad, Oren Melamud, Arul Menezes, Fandong Meng, Helen Meng, Florian Metze, Haitao Mi, Yishu Miao, Tsvetomila Mihaylova, Tomas Mikolov, Timothy Miller, Tristan Miller, David Mimno, Meg Mitchell, Teruhisa Misu, Yusuke Miyao, Ryosuke Miyazaki, Daichi Mochihashi, Ashutosh Modi, Marie-Francine Moens, Samaneh Moghaddam, Saif Mohammad, Karo Moilanen, Luis Gerardo Mojica de la Vega, M. Dolores Molina-González, Manuel Montes, Andres Montoyo, Christof Monz, Taesun Moon, Roser Morante, Silvio Moreira, Alessandro Moschitti, Arjun Mukherjee, Philippe Muller, Yugo Murawaki, Gabriel Murray, Lluís Márquez, Thomas Müller,

Masaaki Nagata, Satoshi Nakamura, Mikio Nakano, Yukiko Nakano, Preslav Nakov, Courtney Napoles, Jason Naradowsky, Karthik Narasimhan, Vivi Nastase, Borja Navarro, Roberto Navigli, Silvia Necsulescu, Mark-Jan Nederhof, Sapna Negi, Matteo Negri, Aida Nematzadeh, Graham Neubig, Jun-Ping Ng, Vincent Ng, Dong Nguyen, Garrett Nicolai, Vlad Niculae, Luis Nieto Piña, Alexandr Nikitin, Zhou Nina, Joakim Nivre,

Brendan O'Connor, Timothy O'Donnell, Stephan Oepen, Kemal Oflazer, Alice Oh, Jong-Hoon Oh, Naoaki Okazaki, Hiroshi G. Okuno, Constantin Orasan, Vicente Ordonez, Miles Osborne, Diarmuid Ó Séaghdha, Mari Ostendorf, Ekaterina Ovchinnikova, Gozde Ozbal,

Sebastian Padó, John Paisley, Alexis Palmer, Martha Palmer, Sinno Jialin Pan, Denis Paperno, Ankur P. Parikh, Cecile Paris, Tommaso Pasini, Peyman Passban, Panupong Pasupat, Siddharth Patwardhan, Michael J. Paul, Adam Pauls, Ellie Pavlick, Bolette Pedersen, Hao Peng, Nanyun Peng, Gerald Penn, Casper Petersen, Slav Petrov, Miriam R. L. Petruck, Hieu Pham, Nghia The Pham, Olivier Pietquin, Mohammad Taher Pilehvar, Manfred Pinkal, Emily Pitler, Paul Piwek, Barbara Plank, Simone Ponzetto, Maja Popović, Fred Popowich, Matt Post, Christopher Potts, Vinod Prabhakaran, Vinodkumar Prabhakaran, Sameer Pradhan, Daniel Preoţiuc-Pietro, Laurent Prévot, Stephen Pulman, Matthew Purver, Sampo Pyysalo,

Xian Qian, Xiuming Qiao, Bing Qin, Guang Qiu, Xipeng Qiu, Chris Quirk,

Alessandro Raganato, Preethi Raghavan, Afshin Rahimi, Altaf Rahman, Jonathan Raiman, Gabriela Ramírez-de-la-Rosa, Ari Rappoport, Mohammad Sadegh Rasooli, Antoine Raux, Sujith Ravi, Marta Recasens, Sravana Reddy, Ines Rehbein, Roi Reichart, Drew Reisinger, David Reitter, Yafeng Ren, Zhaochun Ren, Steve Renals, Evelina Rennes, Corentin Ribeyre, Matthew Richardson, Sebastian Riedel, Jason Riesa, Verena Rieser, Stefan Riezler, German Rigau, Laura Rimell, Alan Ritter, Brian Roark, Molly Roberts, Tim Rocktäschel, Horacio Rodriguez, Stephen Roller, Laurent Romary, Carolyn Rose, Andrew Rosenberg, Sara Rosenthal, Paolo Rosso, Michael Roth, Sascha Rothe, Alexander Rudnicky, Josef Ruppenhofer, Alexander M. Rush,

Mrinmaya Sachan, Mehrnoosh Sadrzadeh, Markus Saers, Kenji Sagae, Horacio Saggion, Benoît Sagot, Magnus Sahlgren, Patrick Saint-Dizier, Avneesh Saluja, Marc Franco Salvador, Felipe Sánchez-Martínez, Germán Sanchis-Trilles, Urko Sanchez Sanz, Anoop Sarkar, Asad Sayeed, Carolina Scarton, David Schlangen,

Allen Schmaltz, Helmut Schmid, Nathan Schneider, Alexandra Schofield, William Schuler, Sabine Schulte im Walde, Sebastian Schuster, Hinrich Schütze, Lane Schwartz, Roy Schwartz, Chun Wei Seah, Djamé Seddah, Wolfgang Seeker, Yohei Seki, Rico Sennrich, Minjoon Seo, Hendra Setiawan, Izhak Shafran, Apurva Shah, Kashif Shah, Shiqi Shen, Chaitanya Shivade, Milad Shokouhi, Szymon Sidor, Avirup Sil, Carina Silberer, Mario J. Silva, Fabrizio Silvestri, Khalil Sima'an, Patrick Simianer, Serra Sinem Tekiroglu, Sameer Singh, Gabriel Skantze, Kevin Small, Noah Smith, Jan Šnajder, Jasper Snoek, Richard Socher, Anders Søgaard, Artem Sokolov, Thamar Solorio, Yan Song, Radu Soricut, Aitor Soroa, Lucia Specia, Caroline Sporleder, Vivek Srikumar, Edward Stabler, Gabriel Stanovsky, Manfred Stede, Mark Steedman, Amanda Stent, Brandon Stewart, Matthew Stone, Svetlana Stoyanchev, Veselin Stoyanov, Carlo Strapparava, Kristina Striegnitz, Markus Strohmaier, Patrick Sturt, Rajen Subba, Katsuhito Sudoh, Weiwei Sun, Xu Sun, Mihai Surdeanu, Jun Suzuki, John Sylak-Glassman, Gabriel Synnaeve, Stan Szpakowicz, Idan Szpektor,

Hiroya Takamura, Partha Talukdar, Akihiro Tamura, Duyu Tang, Jian Tang, Jiliang Tang, Mariona Taulé, Christoph Teichmann, Irina Temnikova, Joel Tetreault, Kapil Thadani, Ran Tian, Jörg Tiedemann, Ivan Titov, Takenobu Tokunaga, Katrin Tomanek, Kentaro Torisawa, David Vilar Torres, Kristina Toutanova, Isabel Trancoso, Harald Trost, Richard Tzong-Han Tsai, Reut Tsarfaty, Kostas Tsioutsiouliklis, Oren Tsur, Yoshimasa Tsuruoka, Yulia Tsvetkov, Gokhan Tur, Marco Turchi, Oscar Täckström,

L. Alfonso Urena Lopez, Olga Uryupina, Masao Utiyama,

Alessandro Valitutti, Lonneke van der Plas, Lucy Vanderwende, David Vandyke, Marten van Schijndel, Paola Velardi, Yannick Versley, Renata Vieira, Esaú Villatoro, Sami Virpioja, Andreas Vlachos, Rob Voigt, Svitlana Volkova, Titus von der Malsburg, Ngoc Thang Vu,

Marilyn Walker, Byron C. Wallace, Stephen Wan, Xiaojun Wan, Chong Wang, Hsin-Min Wang, Lu Wang, Mingxuan Wang, Pidong Wang, Quan Wang, Shaojun Wang, Tong Wang, Wen Wang, Wenya Wang, Yu-Chun Wang, Nigel Ward, Taro Watanabe, Andy Way, Bonnie Webber, Ingmar Weber, Furu Wei, Albert Weichselbraun, Keenon Werling, Michael White, Jason D. Williams, Shuly Wintner, Sam Wiseman, Guillaume Wisniewski, Kam-Fai Wong, Dekai Wu, Hua Wu, Ji Wu, Shumin Wu, Joern Wuebker,

Fei Xia, Rui Xia, Pengtao Xie, Shasha Xie, Wayne Xin Zhao, Chenyan Xiong, Deyi Xiong, Ruifeng Xu, Wang Xuancong, Nianwen Xue,

Yadollah Yaghoobzadeh, Grace Yang, Yi Yang, Zi Yang, Mark Yatskar, Wen-tau Yih, Wenpeng Yin, Anssi Yli-Jyrä, Dani Yogatama, Steve Young, Kai Yu, Xiaotian Yu,

Fabio Massimo Zanzotto, Alessandra Zarcone, Daniel Zeman, Xiaodong Zeng, Luke Zettlemoyer, Deniz Zeyrek, Ke Zhai, Aston Zhang, Chunyue Zhang, Congle Zhang, Hongyi Zhang, Lei Zhang, Meishan Zhang, Min Zhang, Qi Zhang, Sicong Zhang, Yuan Zhang, Yue Zhang, Zhihua Zhang, Chunyue Zhang, Bing Zhao, Kai Zhao, Shenglin Zhao, Tiejun Zhao, Tong Zhao, Guodong Zhou, Qiang Zhou, Xiaodan Zhu, Xiaoning Zhu, Patrick Ziering, Heike Zinsmeister, Willem Zuidema

Invited Talk I: Same Policy Issue, Different Portrayal: The Importance of Tone and Framing in Language

Amber E. Boydstun, University of California at Davis

Many policy issues at the center of politics are relatively fixed; they tend to involve the same basic questions that do not vary over time (e.g., should abortion be legal? should we execute people convicted of horrific crimes?). Political candidates, too, are (like the rest of us) the same people year after year (e.g., Hillary Clinton in 2016 is just an older version of Hillary Clinton in 2015). Yet when citizens consider a given political item (be it a policy issue, a candidate, or something else), they tend not to perceive that item in a fixed way over time. Rather, peoples perceptions of the item tend to depend on how that item is portrayed at that moment. Policy issues and political candidates alike can be portrayed differently through the use of different visual imagery but even more pervasively through variance in the text used to describe them. In this talk, I give a general overview of the importance of issue and candidate portrayals in political communication. I outline the extensive research that has already been done trying to identify different portrayals in text. And I discuss the many opportunities available today to researchers interested in tracking issue and candidate portrayals in text and in examining the effects of issue and candidate portrayals on public attitudes and voting behavior.

Bio: Amber Boydstun is an Associate Professor of Political Science at the University of California, Davis. Her work bridges multiple disciplines, including psychology, journalism, and computer science. Her core research examines the interaction between media and politics, with a focus on how different media portrayals of the same policy issue can prompt citizens and policymakers to respond to that issue in different ways. She uses lab experiments, large-scale media studies, and manual and computational text analysis to study how issues make the news; how issues are "framed" in the news; the dynamics of "media storms"; and how media coverage can shape public opinion and public policy on issues like immigration, gun control, same-sex marriage, and capital punishment. She is author of *Making the News* (Chicago) and co-author of *The Decline of the Death Penalty and the Discovery of Innocence* (Cambridge), as well as many journal articles. Boydstun works with scholars across the globe as a member of the Comparative Agendas Project,[1] a collaborative enterprise by political science and policy scholars to measure international government outputs. She serves on the editorial boards for the journal Political Communication, the Text as Data Association, and the Women Also Know Stuff[2] initiative. Most recently, she co-chaired the 2016 Visions in Methodology Conference. [3]

[1] http://www.comparativeagendas.info/

[2] http://womenalsoknowstuff.com/

[3] http://visionsinmethodology.org/conferences/2016-conference/

Invited Talk II: On Distributional Semantics
Mark Steedman, University of Edinburgh

The central problem in open domain-question answering from text is the problem of *entailment*. Given enough text, the answer is almost certain to be there, but is likely to be expressed differently than in the question—either in a paraphrase, or in a sentence that entails or implies the answer. We cannot afford to bridge this gap by open-ended theorem-proving search. Instead we need a semantics for natural language that directly supports common-sense inference, such as that *arriving somewhere* implies subsequently *being there*, and *invading* a country implies *attacking* it. We would like this semantics to be compatible with traditional logical operator semantics including quantification, negation and tense, so that *not being there* implies *not having arrived*, and *not attacking* implies *not invading*.

There have been many attempts to build such a semantics of content words by hand, from the generative semantics of the '60s to WordNet and other resources of the present. The '60s saw attempts based on generative semantics, while more recently, they have exploited WordNet and other computational resources. However, such systems have been incomplete and language-specific in comparison to the vastness of human common-sense reasoning. One consequence has been renewed interest in the idea of treating the semantics as "hidden", to be discovered through machine learning, an idea that has its origins in the "semantic differential" of Osgood, Suci, and Tannenbaum in the '50s.

There are two distinct modern approaches to the problem of data-driven or "distributional" semantics. The first, which I will call "collocational", is the direct descendant of the semantic differential. In its most basic form, the meaning of a word is taken to be a vector in a space whose dimensions are defined by the lexicon of the language, and whose magnitude is defined by counts of those lexical items within a fixed window on the string (although in practice the dimensionality is reduced and the relation to frequency less direct). Crucially, semantic composition is defined in terms of linear algebraic operations such as vector addition.

The second approach, which I will call "denotational", defines the meaning of a word in terms of the entities (or rather their designators) that it is predicated over and the ensembles of predications over entities of the same types, obtained by machine-reading with wide coverage parsers. Semantic composition is can then be defined as an applicative system, as in traditional formal semantics.

The talk reviews recent work in both collocation- and denotation- based distributional semantics, including some hybrid approaches that interpolate grammatical features with collocational representations, or use probabilistic logics over relations whose arguments denote vectors, and asks for each what dimensions of meaning are actually being represented. It argues that the two approaches are largely orthogonal on these dimensions. Collocational representations are good for representing ambiguity, with linear algebraic composition most effective at disambiguation and representing distributional similarity. Denotational representations represent something more like a traditional compositional semantics, but one in which the primitive relations correspond to those of a hidden language of logical form representing paraphrase and common-sense entailment directly.

To make this point, I will discuss recent work in which collocational distributional representations such as embeddings have been used as proxies for semantic features in models such as LSTM, to guide disambiguation during parsing, while a lexicalized denotation-based distributional semantics is used to support inference of entailment. I will show that this hybrid approach can be applied with a number of parsing mod-

els, including transition-based and supertagging, to support entailment-based QA with denotation-based distributional representations. I will discuss work at Edinburgh and elsewhere in which the semantics of paraphrases is represented by a single cluster identifier and common-sense inference (derived from a learned entailment graph) is built into the lexicon and projected by syntactic derivation, rather than delegated to a later stage of inference. The method can be applied cross-linguistically, in support of machine translation. Ongoing work extends the method to extract multi-word items, light-verb constructions, and an aspect-based semantics for temporal/causal entailment, and to the creation and interrogation of Knowledge Graphs and Semantic Nets via natural language.

Bio: Mark Steedman is Professor of Cognitive Science in the School of Informatics at the University of Edinburgh. Previously, he taught as Professor in the Department of Computer and Information Science at the University of Pennsylvania, which he joined as Associate Professor in 1988, after teaching at the Universities of Warwick and Edinburgh. His PhD is in Artificial Intelligence from the University of Edinburgh. He was a Alfred P. Sloan Fellow at the University of Texas at Austin in 1980/81, and a Visiting Professor at Penn in 1986/87. He is a Fellow of the American Association for Artificial Intelligence, the British Academy, the Royal Society of Edinburgh, the Association for Computational Linguistics, and the Cognitive Science Society, and a Member of the European Academy.

His research interests cover issues in computational linguistics, artificial intelligence, computer science and cognitive science, including syntax and semantics of natural language, wide-coverage parsing and question-answering, comprehension of natural language discourse by humans and by machine, grammar-based language modeling, natural language generation, and the semantics of intonation in spoken discourse. Much of his current NLP research is addressed to probabilistic parsing and robust semantics for question-answering using the CCG grammar formalism, including the acquisition of language from paired sentences and meanings by child and machine. He sometimes works with colleagues in computer animation using these theories to guide the graphical animation of speaking virtual or simulated autonomous human agents. Some of his research concerns the analysis of music by humans and machines.

Table of Contents

Conference Program

Monday, August 8, 2016

8:45–9:00 ***Opening session***

9:00–10:10 ***Invited talk I: Amber Boydstun***

10:10–10:40 ***Coffee break***

Session 1A: Semantic parsing I

10:40–11:00 *Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing*
James Goodman, Andreas Vlachos and Jason Naradowsky

11:00–11:20 *Data Recombination for Neural Semantic Parsing*
Robin Jia and Percy Liang

11:20–11:40 *Inferring Logical Forms From Denotations*
Panupong Pasupat and Percy Liang

11:40–12:00 *Language to Logical Form with Neural Attention*
Li Dong and Mirella Lapata

Session 1B: Information extraction

10:40–11:00 *Unsupervised Person Slot Filling based on Graph Mining*
Dian Yu and Heng Ji

11:20–11:40 *A Multi-media Approach to Cross-lingual Entity Knowledge Transfer*
Di Lu, Xiaoman Pan, Nima Pourdamghani, Shih-Fu Chang, Heng Ji and Kevin Knight

Session 1C: Machine translation I

10:40–11:00 *Models and Inference for Prefix-Constrained Machine Translation*
Joern Wuebker, Spence Green, John DeNero, Sasa Hasan and Minh-Thang Luong

11:00–11:20 *Modeling Coverage for Neural Machine Translation*
Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu and Hang Li

11:20–11:40 *Improving Neural Machine Translation Models with Monolingual Data*
Rico Sennrich, Barry Haddow and Alexandra Birch

11:40–12:00 *Graph-Based Translation Via Graph Segmentation*
Liangyou Li, Andy Way and Qun Liu

Session 1D: Word meaning I

10:40–11:00 *Incremental Acquisition of Verb Hypothesis Space towards Physical World Interaction*
Lanbo She and Joyce Chai

11:00–11:20 *Language Transfer Learning for Supervised Lexical Substitution*
Gerold Hintz and Chris Biemann

11:20–11:40 *Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning*
Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Brian MacWhinney and Chris Dyer

11:40–12:00 *Pointing the Unknown Words*
Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou and Yoshua Bengio

Monday, August 8, 2016 (continued)

Session 1E: Parsing I

11:00–11:20 *Generalized Transition-based Dependency Parsing via Control Parameters*
Bernd Bohnet, Ryan McDonald, Emily Pitler and Ji Ma

11:20–11:40 *A Transition-Based System for Joint Lexical and Syntactic Analysis*
Matthieu Constant and Joakim Nivre

11:40–12:00 *Neural Greedy Constituent Parsing with Dynamic Oracles*
Maximin Coavoux and Benoit Crabbé

Session 1F: Noncompositionality

10:40–11:00 *Literal and Metaphorical Senses in Compositional Distributional Semantic Models*
E.Dario Gutierrez, Ekaterina Shutova, Tyler Marghetis and Benjamin Bergen

11:00–11:20 *Idiom Token Classification using Sentential Distributed Semantics*
Giancarlo Salton, Robert Ross and John Kelleher

11:20–11:40 *Adaptive Joint Learning of Compositional and Non-Compositional Phrase Embeddings*
Kazuma Hashimoto and Yoshimasa Tsuruoka

11:40–12:00 *Metaphor Detection with Topic Transition, Emotion and Cognition in Context*
Hyeju Jang, Yohan Jo, Qinlan Shen, Michael Miller, Seungwhan Moon and Carolyn Rose

12:00–13:40 ***Lunch break***

Session 2A: Word vectors I

14:00–14:20 *Compressing Neural Language Models by Sparse Word Representations*
Yunchuan Chen, Lili Mou, Yan Xu, Ge Li and Zhi Jin

14:20–14:40 *Intrinsic Subspace Evaluation of Word Embedding Representations*
Yadollah Yaghoobzadeh and Hinrich Schütze

14:40–15:00 *On the Role of Seed Lexicons in Learning Bilingual Word Embeddings*
Ivan Vulić and Anna Korhonen

Session 2B: Events and schemas

13:40–14:00 *Liberal Event Extraction and Event Schema Induction*
Lifu Huang, Taylor Cassidy, Xiaocheng Feng, Heng Ji, Clare R. Voss, Jiawei Han and Avirup Sil

14:00–14:20 *Jointly Event Extraction and Visualization on Twitter via Probabilistic Modelling*
Deyu Zhou, Tianmeng Gao and Yulan He

14:20–14:40 *Using Sentence-Level LSTM Language Models for Script Inference*
Karl Pichotta and Raymond J. Mooney

14:40–15:00 *Two Discourse Driven Language Models for Semantics*
Haoruo Peng and Dan Roth

Session 2C: Sentiment analysis

13:40–14:00 *Sentiment Domain Adaptation with Multiple Sources*
Fangzhao Wu and Yongfeng Huang

14:00–14:20 *Connotation Frames: A Data-Driven Investigation*
Hannah Rashkin, Sameer Singh and Yejin Choi

14:20–14:40 *Bi-Transferring Deep Neural Networks for Domain Adaptation*
Guangyou Zhou, Zhiwen Xie, Jimmy Xiangji Huang and Tingting He

14:40–15:00 *Document-level Sentiment Inference with Social, Faction, and Discourse Context*
Eunsol Choi, Hannah Rashkin, Luke Zettlemoyer and Yejin Choi

Session 2D: Parsing II

13:40–14:00 *Active Learning for Dependency Parsing with Partial Annotation*
Zhenghua Li, Min Zhang, Yue Zhang, Zhanyi Liu, Wenliang Chen, Hua Wu and Haifeng Wang

14:20–14:40 *Dependency Parsing with Bounded Block Degree and Well-nestedness via Lagrangian Relaxation and Branch-and-Bound*
Caio Corro, Joseph Le Roux, Mathieu Lacroix, Antoine Rozenknop and Roberto Wolfler Calvo

Session 2E: Information retrieval

13:40–14:00 *Query Expansion with Locally-Trained Word Embeddings*
Fernando Diaz, Bhaskar Mitra and Nick Craswell

14:00–14:20 *Together we stand: Siamese Networks for Similar Question Retrieval*
Arpita Das, Harish Yenala, Manoj Chinnakotla and Manish Shrivastava

14:20–14:40 *News Citation Recommendation with Implicit and Explicit Semantics*
Hao Peng, Jing Liu and Chin-Yew Lin

Monday, August 8, 2016 (continued)

Session 2F: Phonology and morphology

13:40–14:00 *Grapheme-to-Phoneme Models for (Almost) Any Language*
Aliya Deri and Kevin Knight

14:20–14:40 *Neural Word Segmentation Learning for Chinese*
Deng Cai and Hai Zhao

14:40–15:00 *Transition-Based Neural Word Segmentation*
Meishan Zhang, Yue Zhang and Guohong Fu

15:00–15:30 ***Coffee break***

Session 3A: Question answering I

15:30–15:50 *A Parallel-Hierarchical Model for Machine Comprehension on Sparse Data*
Adam Trischler, Zheng Ye, Xingdi Yuan, Jing He and Philip Bachman

15:50–16:10 *Combining Natural Logic and Shallow Reasoning for Question Answering*
Gabor Angeli, Neha Nayak and Christopher D. Manning

16:10–16:30 *Easy Questions First? A Case Study on Curriculum Learning for Question Answering*
Mrinmaya Sachan and Eric Xing

16:30–16:50 *Improved Representation Learning for Question Answer Matching*
Ming Tan, Cicero dos Santos, Bing Xiang and Bowen Zhou

16:50–17:10 *Tables as Semi-structured Knowledge for Question Answering*
Sujay Kumar Jauhar, Peter Turney and Eduard Hovy

Session 3B: Sentence vectors

15:30–15:50 *Neural Summarization by Extracting Sentences and Words*
Jianpeng Cheng and Mirella Lapata

16:10–16:30 *Neural Networks For Negation Scope Detection*
Federico Fancellu, Adam Lopez and Bonnie Webber

16:30–16:50 *CSE: Conceptual Sentence Embeddings based on Attention Model*
Yashen Wang, Heyan Huang, Chong Feng, Qiang Zhou, Jiahui Gu and Xiong Gao

Session 3C: Parsing III

Session 3D: Dialog

15:30–15:50 *DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents*
Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li and Jianshe Zhou

15:50–16:10 *Investigating the Sources of Linguistic Alignment in Conversation*
Gabriel Doyle and Michael C. Frank

16:10–16:30 *Entropy Converges Between Dialogue Participants: Explanations from an Information-Theoretic Perspective*
Yang Xu and David Reitter

16:30–16:50 *Finding the Middle Ground - A Model for Planning Satisficing Answers*
Sabine Janzen, Wolfgang Maaß and Tobias Kowatsch

16:50–17:10 *A Sentence Interaction Network for Modeling Dependence between Sentences*
Biao Liu and Minlie Huang

Session 3E: Generation

15:30–15:50 *Towards more variation in text generation: Developing and evaluating variation models for choice of referential form*
Thiago Castro Ferreira, Emiel Krahmer and Sander Wubben

15:50–16:10 *How Much is 131 Million Dollars? Putting Numbers in Perspective with Compositional Descriptions*
Arun Chaganty and Percy Liang

16:10–16:30 *Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus*
Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville and Yoshua Bengio

16:30–16:50 *Latent Predictor Networks for Code Generation*
Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang and Andrew Senior

16:50–17:10 *Easy Things First: Installments Improve Referring Expression Generation for Objects in Photographs*
Sina Zarrieß and David Schlangen

Session 3F: Entities and coreference

15:30–15:50 *Collective Entity Resolution with Multi-Focal Attention*
Amir Globerson, Nevena Lazic, Soumen Chakrabarti, Amarnag Subramanya, Michael Ringaard and Fernando Pereira

16:30–16:50 *Which Coreference Evaluation Metric Do You Trust? A Proposal for a Link-based Entity Aware Metric*
Nafise Sadat Moosavi and Michael Strube

16:50–17:10 *Improving Coreference Resolution by Learning Entity-Level Distributed Representations*
Kevin Clark and Christopher D. Manning

Session 3G: Topics

15:30–15:50 *Effects of Creativity and Cluster Tightness on Short Text Clustering Performance*
Catherine Finegan-Dollak, Reed Coke, Rui Zhang, Xiangyi Ye and Dragomir Radev

15:50–16:10 *Generative Topic Embedding: a Continuous Representation of Documents*
Shaohua Li, Tat-Seng Chua, Jun Zhu and Chunyan Miao

16:30–16:50 *Detecting Common Discussion Topics Across Culture From News Reader Comments*
Bei Shi, Wai Lam, Lidong Bing and Yinqing Xu

16:50–17:10 *A Discriminative Topic Model using Document Network Structure*
Weiwei Yang, Jordan Boyd-Graber and Philip Resnik

18:00–21:00 ***Poster and dinner session I (includes SRW)***

AraSenTi: Large-Scale Twitter-Specific Arabic Sentiment Lexicons
Nora Al-Twairesh, Hend Al-Khalifa and Abdulmalik AlSalman

Unsupervised Multi-Author Document Decomposition Based on Hidden Markov Model
Khaled Aldebei, Xiangjian He, Wenjing Jia and Jie Yang

Automatic Text Scoring Using Neural Networks
Dimitrios Alikaniotis, Helen Yannakoudakis and Marek Rei

Improved Semantic Parsers For If-Then Statements
I. Beltagy and Chris Quirk

Universal Dependencies for Learner English
Yevgeni Berzak, Jessica Kenney, Carolyn Spadine, Jing Xian Wang, Lucia Lam, Keiko Sophie Mori, Sebastian Garza and Boris Katz

Extracting token-level signals of syntactic processing from fMRI - with an application to PoS induction
Joachim Bingel, Maria Barrett and Anders Søgaard

Bidirectional Recurrent Convolutional Neural Network for Relation Classification
Rui Cai, Xiaodong Zhang and Houfeng Wang

Sentence Rewriting for Semantic Parsing
Bo Chen, Le Sun, Xianpei Han and Bo An

Chinese Zero Pronoun Resolution with Deep Neural Networks
Chen Chen and Vincent Ng

Constrained Multi-Task Learning for Automated Essay Scoring
Ronan Cummins, Meng Zhang and Ted Briscoe

CFO: Conditional Focused Neural Question Answering with Large-scale Knowledge Bases
Zihang Dai, Lei Li and Wei Xu

Verbs Taking Clausal and Non-Finite Arguments as Signals of Modality – Revisiting the Issue of Meaning Grounded in Syntax
Judith Eckle-Kohler

Tree-to-Sequence Attentional Neural Machine Translation
Akiko Eriguchi, Kazuma Hashimoto and Yoshimasa Tsuruoka

Coordination Annotation Extension in the Penn Tree Bank
Jessica Ficler and Yoav Goldberg

Analyzing Biases in Human Perception of User Age and Gender from Text
Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar and Daniel Preoţiuc-Pietro

Modeling Social Norms Evolution for Personalized Sentiment Classification
Lin Gong, Mohammad Al Boni and Hongning Wang

Modeling Concept Dependencies in a Scientific Corpus
Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan and Gully Burns

Normalized Log-Linear Interpolation of Backoff Language Models is Efficient
Kenneth Heafield, Chase Geigle, Sean Massung and Lane Schwartz

How well do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation
Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin and Wei-Ying Ma

Embeddings for Word Sense Disambiguation: An Evaluation Study
Ignacio Iacobacci, Mohammad Taher Pilehvar and Roberto Navigli

Understanding Discourse on Work and Job-Related Well-Being in Public Social Media
Tong Liu, Christopher Homan, Cecilia Ovesdotter Alm, Megan Lytle, Ann Marie White and Henry Kautz

Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models
Minh-Thang Luong and Christopher D. Manning

End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF
Xuezhe Ma and Eduard Hovy

Off-topic Response Detection for Spontaneous Spoken English Assessment
Andrey Malinin, Rogier van Dalen, Kate Knill, Yu Wang and Mark Gales

Synthesizing Compound Words for Machine Translation
Austin Matthews, Eva Schlinger, Alon Lavie and Chris Dyer

Harnessing Cognitive Features for Sarcasm Detection
Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey and Pushpak Bhattacharyya

End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures
Makoto Miwa and Mohit Bansal

A short proof that O_2 is an MCFL
Mark-Jan Nederhof

Context-aware Argumentative Relation Mining
Huy Nguyen and Diane Litman

Leveraging Inflection Tables for Stemming and Lemmatization.
Garrett Nicolai and Grzegorz Kondrak

Scaling a Natural Language Generation System
Jonathan Pfeil and Soumya Ray

ALTO: Active Learning with Topic Overviews for Speeding Label Induction and Document Labeling
Forough Poursabzi-Sangdeh, Jordan Boyd-Graber, Leah Findlater and Kevin Seppi

Predicting the Rise and Fall of Scientific Topics from Trends in their Rhetorical Framing
Vinodkumar Prabhakaran, William L. Hamilton, Dan McFarland and Dan Jurafsky

Compositional Sequence Labeling Models for Error Detection in Learner Writing
Marek Rei and Helen Yannakoudakis

Neural Semantic Role Labeling with Dependency Path Embeddings
Michael Roth and Mirella Lapata

Prediction of Prospective User Engagement with Intelligent Assistants
Shumpei Sano, Nobuhiro Kaji and Manabu Sassano

Resolving References to Objects in Photographs using the Words-As-Classifiers Model
David Schlangen, Sina Zarrieß and Casey Kennington

RBPB: Regularization-Based Pattern Balancing Method for Event Extraction
Lei Sha, Jing Liu, Chin-Yew Lin, Sujian Li, Baobao Chang and Zhifang Sui

Neural Network-Based Model for Japanese Predicate Argument Structure Analysis
Tomohide Shibata, Daisuke Kawahara and Sadao Kurohashi

Addressing Limited Data for Textual Entailment Across Domains
Chaitanya Shivade, Preethi Raghavan and Siddharth Patwardhan

Annotating and Predicting Non-Restrictive Noun Phrase Modifications
Gabriel Stanovsky and Ido Dagan

Bilingual Segmented Topic Model
Akihiro Tamura and Eiichiro Sumita

Learning Semantically and Additively Compositional Distributional Representations
Ran Tian, Naoaki Okazaki and Kentaro Inui

Inner Attention based Recurrent Neural Networks for Answer Selection
Bingning Wang, Kang Liu and Jun Zhao

Relation Classification via Multi-Level Attention CNNs
Linlin Wang, Zhu Cao, Gerard de Melo and Zhiyuan Liu

Knowledge Base Completion via Coupled Path Ranking
Quan Wang, Jing Liu, Yuanfei Luo, Bin Wang and Chin-Yew Lin

Larger-Context Language Modelling with Recurrent Neural Network
Tian Wang and Kyunghyun Cho

The Creation and Analysis of a Website Privacy Policy Corpus
Shomir Wilson, Florian Schaub, Aswarth Abhilash Dara, Frederick Liu, Sushain Cherivirala, Pedro Giovanni Leon, Mads Schaarup Andersen, Sebastian Zimmeck, Kanthashree Mysore Sathyendra, N. Cameron Russell, Thomas B. Norton, Eduard Hovy, Joel Reidenberg and Norman Sadeh

Sequence-based Structured Prediction for Semantic Parsing
Chunyang Xiao, Marc Dymetman and Claire Gardent

Learning Word Meta-Embeddings
Wenpeng Yin and Hinrich Schütze

Towards Constructing Sports News from Live Text Commentary
Jianmin Zhang, Jin-ge Yao and Xiaojun Wan

A Continuous Space Rule Selection Model for Syntax-based Statistical Machine Translation
Jingyi Zhang, Masao Utiyama, Eiichro Sumita, Graham Neubig and Satoshi Nakamura

Probabilistic Graph-based Dependency Parsing with Convolutional Neural Network
Zhisong Zhang, Hai Zhao and Lianhui Qin

A Search-Based Dynamic Reranking Model for Dependency Parsing
Hao Zhou, Yue Zhang, Shujian Huang, Junsheng Zhou, Xin-Yu Dai and Jiajun Chen

Cross-Lingual Sentiment Classification with Bilingual Document Representation Learning
Xinjie Zhou, Xiaojun Wan and Jianguo Xiao

Segment-Level Sequence Modeling using Gated Recursive Semi-Markov Conditional Random Fields
Jingwei Zhuo, Yong Cao, Jun Zhu, Bo Zhang and Zaiqing Nie

Tuesday, August 9, 2016

9:00–10:10 ***Invited talk II: Mark Steedman***

10:10–10:40 ***Coffee break***

Session 4A: Relations and knowledge bases

10:40–11:00 *Identifying Causal Relations Using Parallel Wikipedia Articles*
Christopher Hidey and Kathy McKeown

11:00–11:20 *Compositional Learning of Embeddings for Relation Paths in Knowledge Base and Text*
Kristina Toutanova, Victoria Lin, Wen-tau Yih, Hoifung Poon and Chris Quirk

11:20–11:40 *Commonsense Knowledge Base Completion*
Xiang Li, Aynaz Taheri, Lifu Tu and Kevin Gimpel

Session 4B: Semantic parsing II

10:40–11:00 *Simpler Context-Dependent Logical Forms via Model Projections*
Reginald Long, Panupong Pasupat and Percy Liang

11:20–11:40 *A Fast Unified Model for Parsing and Sentence Understanding*
Samuel R. Bowman, Jon Gauthier, Abhinav Rastogi, Raghav Gupta, Christopher D. Manning and Christopher Potts

Session 4C: Word meaning II

11:20–11:40 *Investigating Language Universal and Specific Properties in Word Embeddings*
Peng Qian, Xipeng Qiu and Xuanjing Huang

11:40–12:00 *Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change*
William L. Hamilton, Jure Leskovec and Dan Jurafsky

Session 4D: Tasks and datasets

10:40–11:00 *Beyond Plain Spatial Knowledge: Determining Where Entities Are and Are Not Located, and For How Long*
Alakananda Vempala and Eduardo Blanco

11:00–11:20 *LexSemTm: A Semantic Dataset Based on All-words Unsupervised Sense Distribution Learning*
Andrew Bennett, Timothy Baldwin, Jey Han Lau, Diana McCarthy and Francis Bond

11:20–11:40 *The LAMBADA dataset: Word prediction requiring a broad discourse context*
Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda and Raquel Fernandez

11:40–12:00 *WikiReading: A Novel Large-scale Language Understanding Task over Wikipedia*
Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey and David Berthelot

Session 4E: Parsing IV

11:00–11:20 *Optimizing Spectral Learning for Parsing*
Shashi Narayan and Shay B. Cohen

11:20–11:40 *Stack-propagation: Improved Representation Learning for Syntax*
Yuan Zhang and David Weiss

Session 4F: Document analysis

10:40–11:00 *Inferring Perceived Demographics from User Emotional Tone and User-Environment Emotional Contrast*
Svitlana Volkova and Yoram Bachrach

11:00–11:20 *Prototype Synthesis for Model Laws*
Matthew Burgess, Eugenia Giraudy and Eytan Adar

11:20–11:40 *Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM*
Ivan Habernal and Iryna Gurevych

11:40–12:00 *Discovery of Treatments from Text Corpora*
Christian Fong and Justin Grimmer

12:00–13:40 ***Lunch break***

Session 5A: Deep learning (short papers)

Session 5B: Semantics and generation (short papers)

Session 5C: Machine translation II (short papers)

Tuesday, August 9, 2016 (continued)

Session 5D: Text classification (short papers)

Session 5E: Potpourri I (short papers)

15:00–15:30 ***Coffee break***

Session 6A: Machine learning

15:30–15:50 *Learning Structured Predictors from Bandit Feedback for Interactive NLP*
Artem Sokolov, Julia Kreutzer, Christopher Lo and Stefan Riezler

16:10–16:30 *Deep Reinforcement Learning with a Natural Language Action Space*
Ji He, Jianshu Chen, Xiaodong He, Jianfeng Gao, Lihong Li, Li Deng and Mari Ostendorf

16:30–16:50 *Incorporating Copying Mechanism in Sequence-to-Sequence Learning*
Jiatao Gu, Zhengdong Lu, Hang Li and Victor O.K. Li

16:50–17:10 *Cross-domain Text Classification with Multiple Domains and Disparate Label Sets*
Himanshu Sharad Bhatt, Manjira Sinha and Shourya Roy

Session 6B: Word vectors II

15:50–16:10 *Morphological Smoothing and Extrapolation of Word Embeddings*
Ryan Cotterell, Hinrich Schütze and Jason Eisner

16:10–16:30 *Cross-lingual Models of Word Embeddings: An Empirical Comparison*
Shyam Upadhyay, Manaal Faruqui, Chris Dyer and Dan Roth

16:30–16:50 *Take and Took, Gaggle and Goose, Book and Read: Evaluating the Utility of Vector Differences for Lexical Relation Learning*
Ekaterina Vylomova, Laura Rimell, Trevor Cohn and Timothy Baldwin

Tuesday, August 9, 2016 (continued)

Session 6C: Machine translation III

15:30–15:50 *Minimum Risk Training for Neural Machine Translation*
Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun and Yang Liu

16:10–16:30 *A Character-level Decoder without Explicit Segmentation for Neural Machine Translation*
Junyoung Chung, Kyunghyun Cho and Yoshua Bengio

16:30–16:50 *Target-Side Context for Discriminative Models in Statistical Machine Translation*
Aleš Tamchyna, Alexander Fraser, Ondřej Bojar and Marcin Junczys-Dowmunt

16:50–17:10 *Neural Machine Translation of Rare Words with Subword Units*
Rico Sennrich, Barry Haddow and Alexandra Birch

Session 6D: Discourse

15:30–15:50 *Implicit Discourse Relation Detection via a Deep Architecture with Gated Relevance Network*
Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu and Xuanjing Huang

15:50–16:10 *Model Architectures for Quotation Detection*
Christian Scheible, Roman Klinger and Sebastian Padó

16:10–16:30 *Speech Act Modeling of Written Asynchronous Conversations with Task-Specific Embeddings and Conditional Structured Models*
Shafiq Joty and Enamul Hoque

16:30–16:50 *Situation entity types: automatic classification of clause-level aspect*
Annemarie Friedrich, Alexis Palmer and Manfred Pinkal

Session 6E: Language and vision

15:30–15:50 *Learning Prototypical Event Structure from Photo Albums*
Antoine Bosselut, Jianfu Chen, David Warren, Hannaneh Hajishirzi and Yejin Choi

15:50–16:10 *Cross-Lingual Image Caption Generation*
Takashi Miyazaki and Nobuyuki Shimizu

16:10–16:30 *Learning Concept Taxonomies from Multi-modal Data*
Hao Zhang, Zhiting Hu, Yuntian Deng, Mrinmaya Sachan, Zhicheng Yan and Eric Xing

16:30–16:50 *Generating Natural Questions About an Image*
Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He and Lucy Vanderwende

16:50–17:10 *Physical Causality of Action Verbs in Grounded Language Understanding*
Qiaozi Gao, Malcolm Doering, Shaohua Yang and Joyce Chai

Session 6F: Summarization

16:30–16:50 *Optimizing an Approximation of ROUGE - a Problem-Reduction Approach to Extractive Multi-Document Summarization*
Maxime Peyrard and Judith Eckle-Kohler

Session 6G: Learner language

15:30–15:50 *Phrase Structure Annotation and Parsing for Learner English*
Ryo Nagata and Keisuke Sakaguchi

15:50–16:10 *A Trainable Spaced Repetition Model for Language Learning*
Burr Settles and Brendan Meeder

16:30–16:50 *User Modeling in Language Learning with Macaronic Texts*
Adithya Renduchintala, Rebecca Knowles, Philipp Koehn and Jason Eisner

16:50–17:10 *On the Similarities Between Native, Non-native and Translated Texts*
Ella Rabinovich, Sergiu Nisioi, Noam Ordan and Shuly Wintner

Tuesday, August 9, 2016 (continued)

17:30–19:00 ***Poster and dinner session II***

Learning Text Pair Similarity with Context-sensitive Autoencoders
Hadi Amiri, Philip Resnik, Jordan Boyd-Graber and Hal Daumé III

Linguistic Benchmarks of Online News Article Quality
Ioannis Arapakis, Filipa Peleja, Barla Berkant and Joao Magalhaes

Alleviating Poor Context with Background Knowledge for Named Entity Disambiguation
Ander Barrena, Aitor Soroa and Eneko Agirre

Mining Paraphrasal Typed Templates from a Plain Text Corpus
Or Biran, Terra Blevins and Kathleen McKeown

How to Train Dependency Parsers with Inexact Search for Joint Sentence Boundary Detection and Parsing of Entire Documents
Anders Björkelund, Agnieszka Faleńska, Wolfgang Seeker and Jonas Kuhn

MUTT: Metric Unit TesTing for Language Generation Tasks
William Boag, Renan Campos, Kate Saenko and Anna Rumshisky

N-gram language models for massively parallel devices
Nikolay Bogoychev and Adam Lopez

Cross-Lingual Morphological Tagging for Low-Resource Languages
Jan Buys and Jan A. Botha

Semi-Supervised Learning for Neural Machine Translation
Yong Cheng, Wei Xu, Zhongjun He, Wei He, Hua Wu, Maosong Sun and Yang Liu

Strategies for Training Large Vocabulary Neural Language Models
Wenlin Chen, David Grangier and Michael Auli

Predicting the Compositionality of Nominal Compounds: Giving Word Embeddings a Hard Time
Silvio Cordeiro, Carlos Ramisch, Marco Idiart and Aline Villavicencio

Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints
Greg Durrett, Taylor Berg-Kirkpatrick and Dan Klein

Set-Theoretic Alignment for Comparable Corpora
Thierry Etchegoyhen and Andoni Azpeitia

Jointly Learning to Embed and Predict with Multiple Languages
Daniel C. Ferreira, André F. T. Martins and Mariana S. C. Almeida

Supersense Embeddings: A Unified Model for Supersense Interpretation, Prediction, and Utilization
Lucie Flekova and Iryna Gurevych

Efficient techniques for parsing with tree automata
Jonas Groschwitz, Alexander Koller and Mark Johnson

A Vector Space for Distributional Semantics for Entailment
James Henderson and Diana Popa

Hidden Softmax Sequence Model for Dialogue Structure Analysis
Zhiyang He, Xien Liu, Ping Lv and Ji Wu

Summarizing Source Code using a Neural Attention Model
Srinivasan Iyer, Ioannis Konstas, Alvin Cheung and Luke Zettlemoyer

Continuous Profile Models in ASL Syntactic Facial Expression Synthesis
Hernisa Kacorri and Matt Huenerfauth

Evaluating Sentiment Analysis in the Context of Securities Trading
Siavash Kazemian, Shunan Zhao and Gerald Penn

Edge-Linear First-Order Dependency Parsing with Undirected Minimum Spanning Tree Inference
Effi Levi, Roi Reichart and Ari Rappoport

Topic Extraction from Microblog Posts Using Conversation Structures
Jing Li, Ming Liao, Wei Gao, Yulan He and Kam-Fai Wong

Neural Relation Extraction with Selective Attention over Instances
Yankai Lin, Shiqi Shen, Zhiyuan Liu, Huanbo Luan and Maosong Sun

Leveraging FrameNet to Improve Automatic Event Detection
Shulin Liu, Yubo Chen, Shizhu He, Kang Liu and Jun Zhao

Learning To Use Formulas To Solve Simple Arithmetic Problems
Arindam Mitra and Chitta Baral

Unravelling Names of Fictional Characters
Katerina Papantoniou and Stasinos Konstantopoulos

Most "babies" are "little" and most "problems" are "huge": Compositional Entailment in Adjective-Nouns
Ellie Pavlick and Chris Callison-Burch

Modeling Stance in Student Essays
Isaac Persing and Vincent Ng

A New Psychometric-inspired Evaluation Metric for Chinese Word Segmentation
Peng Qian, Xipeng Qiu and Xuanjing Huang

Temporal Anchoring of Events for the TimeBank Corpus
Nils Reimers, Nazanin Dehghani and Iryna Gurevych

Grammatical Error Correction: Machine Translation and Classifiers
Alla Rozovskaya and Dan Roth

Recurrent neural network models for disease name recognition using domain invariant features
Sunil Sahu and Ashish Anand

Domain Adaptation for Authorship Attribution: Improved Structural Correspondence Learning
Upendra Sapkota, Thamar Solorio, Manuel Montes and Steven Bethard

A Corpus-Based Analysis of Canonical Word Order of Japanese Double Object Constructions
Ryohei Sasano and Manabu Okumura

Knowledge-Based Semantic Embedding for Machine Translation
Chen Shi, Shujie Liu, Shuo Ren, Shi Feng, Mu Li, Ming Zhou, Xu Sun and Houfeng Wang

One for All: Towards Language Independent Named Entity Linking
Avirup Sil and Radu Florian

On Approximately Searching for Similar Word Embeddings
Kohei Sugawara, Hayato Kobayashi and Masajiro Iwasaki

Composing Distributed Representations of Relational Patterns
Sho Takase, Naoaki Okazaki and Kentaro Inui

The More Antecedents, the Merrier: Resolving Multi-Antecedent Anaphors
Hardik Vala, Andrew Piper and Derek Ruths

Automatic Labeling of Topic Models Using Text Summaries
Xiaojun Wan and Tianming Wang

Graph-based Dependency Parsing with Bidirectional LSTM
Wenhui Wang and Baobao Chang

TransG : A Generative Model for Knowledge Graph Embedding
Han Xiao, Minlie Huang and Xiaoyan Zhu

Question Answering on Freebase via Relation Extraction and Textual Evidence
Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang and Dongyan Zhao

Vector-space topic models for detecting Alzheimer's disease
Maria Yancheva and Frank Rudzicz

Chinese Couplet Generation with Neural Network Structures
Rui Yan, Cheng-Te Li, Xiaohua Hu and Ming Zhang

20:00 ***Social event***

Wednesday, August 10, 2016

9:00–9:40 ***President's talk***

9:40–10:10 ***Coffee break***

Session 7A: Outstanding papers I

10:10–10:30 *A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task*
Danqi Chen, Jason Bolton and Christopher D. Manning

10:30–10:50 *Learning Language Games through Interaction*
Sida I. Wang, Percy Liang and Christopher D. Manning

10:50–11:10 *Finding Non-Arbitrary Form-Meaning Systematicity Using String-Metric Learning for Kernel Regression*
E.Dario Gutierrez, Roger Levy and Benjamin Bergen

11:10–11:30 *Improving Hypernymy Detection with an Integrated Path-based and Distributional Method*
Vered Shwartz, Yoav Goldberg and Ido Dagan

Session 7B: Outstanding papers II

10:10–10:30 *Multimodal Pivots for Image Caption Translation*
Julian Hitschler, Shigehiko Schamoni and Stefan Riezler

10:30–10:50 *Harnessing Deep Neural Networks with Logic Rules*
Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy and Eric Xing

10:50–11:10 *Case and Cause in Icelandic: Reconstructing Causal Networks of Cascaded Language Changes*
Fermin Moscoso del Prado Martin and Christian Brendel

11:10–11:30 *On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems*
Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen and Steve Young

Wednesday, August 10, 2016 (continued)

11:30–11:50 *Globally Normalized Transition-Based Neural Networks*
Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov and Michael Collins

11:50–13:30 ***Lunch break***

13:30–15:00 ***ACL business meeting (open to all)***

15:00–15:30 ***Coffee break***

Session 8A: Question answering II (short papers)

Session 8B: Word vectors III (short papers)

Session 8C: Topics and discourse (short papers)

Session 8D: Syntax and morphology (short papers)

Session 8E: Potpourri II (short papers)

17:00–19:00 ***Awards and closing session***

Noise reduction and targeted exploration in imitation learning for Abstract Meaning Representation parsing

James Goodman* **Andreas Vlachos†** **Jason Naradowsky***
* Computer Science Department, University College London
james@janigo.co.uk, jason.narad@gmail.com
† Department of Computer Science, University of Sheffield
a.vlachos@sheffield.ac.uk

Abstract

Semantic parsers map natural language statements into meaning representations, and must abstract over syntactic phenomena, resolve anaphora, and identify word senses to eliminate ambiguous interpretations. Abstract meaning representation (AMR) is a recent example of one such semantic formalism which, similar to a dependency parse, utilizes a graph to represent relationships between concepts (Banarescu et al., 2013). As with dependency parsing, transition-based approaches are a common approach to this problem. However, when trained in the traditional manner these systems are susceptible to the accumulation of errors when they find undesirable states during greedy decoding. Imitation learning algorithms have been shown to help these systems recover from such errors. To effectively use these methods for AMR parsing we find it highly beneficial to introduce two novel extensions: noise reduction and targeted exploration. The former mitigates the noise in the feature representation, a result of the complexity of the task. The latter targets the exploration steps of imitation learning towards areas which are likely to provide the most information in the context of a large action-space. We achieve state-of-the art results, and improve upon standard transition-based parsing by 4.7 F_1 points.

1 Introduction

Meaning representation languages and systems have been devised for specific domains, such as ATIS for air-travel bookings (Dahl et al., 1994) and database queries (Zelle and Mooney, 1996; Liang et al., 2013). Such machine-interpretable representations enable many applications relying on natural language understanding. The ambition of Abstract Meaning Representation (AMR) is that it is domain-independent and useful in a variety of applications (Banarescu et al., 2013).

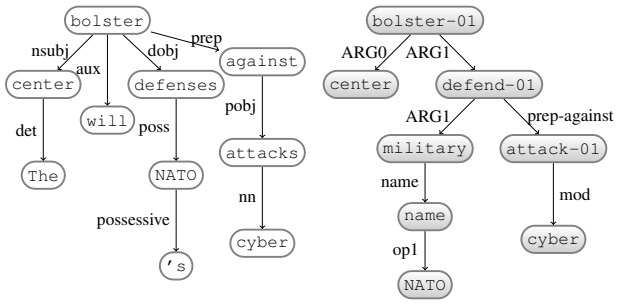


Figure 1: Dependency (left) and AMR graph (right) for: "The center will bolster NATO's defenses against cyber-attacks.'

The first AMR parser by Flanigan et al. (2014) used graph-based inference to find a highest-scoring maximum spanning connected acyclic graph. Later work by Wang et al. (2015b) was inspired by the similarity between the dependency parse of a sentence and its semantic AMR graph (Figure 1). Wang et al. (2015b) start from the dependency parse and learn a transition-based parser that converts it incrementally into an AMR graph using greedy decoding. An advantage of this approach is that the initial stage of dependency parsing is well-studied and trained using larger corpora than that for which AMR annotations exist.

Greedy decoding, where the parser builds the parse while maintaining only the best hypothesis at each step, has a well-documented disadvantage: error propagation (McDonald and Nivre, 2007). When the parser encounters states during parsing that are unlike those found during training, it is more likely to make mistakes, leading to states which are increasingly more foreign and causing errors to accumulate.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1–11,
Berlin, Germany, August 7-12, 2016.

One way to ameliorate this problem is to employ imitation learning algorithms for structured prediction. Algorithms such as SEARN (Daumé III et al., 2009), DAGGER (Ross et al., 2011), and LOLS (Chang et al., 2015) address the problem of error propagation by iteratively adjusting the training data to increasingly expose the model to training instances it is likely to encounter during test. Such algorithms have been shown to improve performance in a variety of tasks including information extraction(Vlachos and Craven, 2011), dependency parsing (Goldberg and Nivre, 2013), and feature selection (He et al., 2013). In this work we build on the transition-based parsing approach of Wang et al. (2015b) and explore the applicability of different imitation algorithms to AMR parsing, which has a more complex output space than those considered previously.

The complexity of AMR parsing affects transition-based methods that rely on features to represent structure, since these often cannot capture the information necessary to predict the correct transition according to the gold standard. In other words, the features defined are not sufficient to "explain" why different actions should preferred by the model. Such instances become noise during training, resulting in lower accuracy. To address this issue, we show that the α-bound Khardon and Wachman (2007), which drops consistently misclassified training instances, provides a simple and effective way of reducing noise and raising performance in perceptron-style classification training, and does so reliably across a range of parameter settings. This noise reduction is essential for imitation learning to gain traction in this task, and we gain 1.8 points of F_1-Score using the DAGGER imitation learning algorithm.

DAGGER relies on an externally specified expert (oracle) to define the correct action in each state; this defines a simple 0-1 loss function for each action. Other imitation learning algorithms (such as LOLS, SEARN) and the variant of DAGGER proposed by Vlachos and Clark (2014) (henceforth V-DAGGER) can leverage a task level loss function that does not decompose over the actions taken to construct the AMR graph. However these require extra computations to roll-out to an end-state AMR graph for each possible action *not* taken. The large action-space of our transition system makes these algorithms computationally infeasible, and roll-outs to an end-state for many of the possible actions will provide little additional information. Hence we modify the algorithms to target this exploration to actions where the classifier being trained is uncertain of the correct response, or disagrees with the expert. This provides a further gain of 2.7 F_1 points.

This paper extends imitation learning to structured prediction tasks more complex than previously attempted. In the process, we review and compare recently proposed algorithms and show how their components can be recombined and adjusted to construct a variant appropriate to the task in hand. Hence we invest some effort reviewing these algorithms and their common elements.

Overall, we obtain a final F-Score of 0.70 on the newswire corpus of LDC2013E117 (Knight et al., 2014). This is identical to the score obtained by Wang et al. (2015a), the highest so far published. Our gain of 4.5 F_1 points from imitation learning over standard transition-based parsing is orthogonal to that of Wang et al. (2015a) from additional trained analysers, including co-reference and semantic role labellers, incorporated in the feature set. We further test on five other corpora of AMR graphs, including weblog domains, and show a consistent improvement in all cases with the application of imitation learning using DAGGER and the targeted V-DAGGER we propose here.

2 Transition-based AMR parsing

AMR parsing is an example of the wider family of structured prediction problems, in which we seek a mapping from an input $\mathbf{x} \in \mathcal{X}$ to a structured output $\mathbf{y} \in \mathcal{Y}$. Here $\mathbf{x}$ is the dependency tree, and $\mathbf{y}$ the AMR graph; both are graphs and we notationally replace $\mathbf{x}$ with s_1 and $\mathbf{y}$ with s_T, with $s_{1\ldots T} \in \mathcal{S}$. s_i are the intermediate graph configurations (states) that the system transitions through.

A transition-based parser starts with an input s_1, and selects an action $a_1 \in \mathcal{A}$, using a classifier. a_i converts s_i into s_{i+1}, i.e. $s_{i+1} = a_i(s_i)$. We term the set of states and actions $\langle s_1, a_1, \ldots a_{T-1}, s_T \rangle$ a trajectory of length T. The classifier $\hat{\pi}$ is trained to predict a_i from s_i, with $\hat{\pi}(s) = \arg\max_{a \in \mathcal{A}} \mathbf{w_a} \cdot \Phi(s)$, assuming a linear classifier and a feature function $\Phi(s)$.

We require an expert, π^*, that can indicate what actions should be taken on each s_i to reach the target (gold) end state. In problems like POS-tagging these are directly inferable from gold, as the number of actions (T) equals the number of

Action Name	Param.	Pre-conditions	Outcome of action
NextEdge	l_r	β non-empty	Set label of edge (σ_0, β_0) to l_r. Pop β_0.
NextNode	l_c	β empty	Set concept of node σ_0 to l_c. Pop σ_0, and initialise β.
Swap		β non-empty	Make β_0 parent of σ_0 (reverse edge) and its sub-graph. Pop β_0 and insert β_0 as σ_1.
ReplaceHead		β non-empty	Pop σ_0 and delete it from the graph. Parents of σ_0 become parents of β_0. Other children of σ_0 become children of β_0. Insert β_0 at the head of σ and re-initialise β.
Reattach	κ	β non-empty	Pop β_0 and delete edge (σ_0, β_0). Attach β_0 as a child of κ. If κ has already been popped from σ then re-insert it as σ_1.
DeleteNode		β empty; leaf σ_0	Pop σ_0 and delete it from the graph.
Insert	l_c		Insert a new node δ with AMR concept l_c as the parent of σ_0, and insert δ into σ.
InsertBelow			Insert a new node δ with AMR concept l_c as a child of σ_0.

Table 1: Action Space for the transition-based graph parsing algorithm

Algorithm 1: Greedy transition-based parsing

Data: policy π, start state s_1
Result: terminal state s_T

$s_{current} \leftarrow s_1$;
while $s_{current}$ *not terminal* **do**
  $a_{next} \leftarrow \pi(s_{current})$
  $s_{current} \leftarrow a_{next}(s_{current})$
$s_T \leftarrow s_{current}$

tokens with a 1:1 correspondence between them. In dependency parsing and AMR parsing this is not straightforward and dedicated transition systems are devised.

Given a labeled training dataset $\mathcal{D}$, algorithm 1 is first used to generate a trajectory for each of the inputs ($d \in D$) with $\pi = \pi^*$, the expert from which we wish to generalise. The data produced from all expert trajectories (i.e. $\langle s_{i,d}, a_{i,d} \rangle$ for all $i \in 1 \ldots T$ and all $d \in 1 \ldots D$), are used to train the classifier $\hat{\pi}$, the learned classifier, using standard supervised learning techniques. Algorithm 1 is reused to apply $\hat{\pi}$ to unseen data. Our transition system (defining $\mathcal{A}$, $\mathcal{S}$), and feature sets are based on Wang et al. (2015b), and are not the main focus of this paper. We introduce the key concepts here, with more details in the supplemental material.

We initialise the state with the stack of the nodes in the dependency tree, root node at the bottom. This stack is termed σ. A second stack, β is initialised with all children of the top node in σ. The state at any time is described by σ, β, and the current graph (which starts as the dependency tree with one node per token). At any stage before termination some of the nodes will be labelled with words from the sentence, and others with AMR concepts. Each action manipulates the top nodes in each stack, σ_0 and β_0. We reach a terminal state when σ is empty. The objective function to maximise is the Smatch score (Cai and Knight, 2013), which calculates an F_1-Score between the predicted and gold-target AMR graphs.

Table 1 summarises the actions in $\mathcal{A}$. NextNode and NextEdge form the core action set, labelling nodes and edges respectively without changing the graph structure. Swap, Reattach and ReplaceHead change graph structure, keeping it a tree. We permit a Reattach action to use parameter κ equal to any node within six edges from σ_0, excluding any that would disconnect the graph or create a cycle.

The Insert/InsertBelow actions insert a new node as a parent/child of σ_0. These actions are not used in Wang et al. (2015b), but Insert is very similar to the Infer action of Wang et al. (2015a). We do not use the Reentrance action of Wang et al. (2015b), as we found it not to add any benefit. This means that the output AMR is always a tree.

Our transition system has two characteristics which provide a particular challenge: given a sentence, the trajectory length T is theoretically unbounded; and $|\mathcal{A}|$ can be of the order 10^3 to 10^4. Commonly used transition-based systems have a fixed trajectory length T, which often arises naturally from the nature of the problem. In PoS-tagging each token requires a single action, and in syntactic parsing the total size of the graph is limited to the number of tokens in the input. The lack of a bound in T here is due to Insert actions that can grow the the graph, potentially *ad infinitum*, and actions like Reattach, which can move a sub-graph repeatedly back-and-forth. The action space size is due to the size of the AMR vocabulary, which for relations (edge-labels) is restricted to about 100 possible values, but for concepts (node-labels) is almost as broad as an En-

Algorithm 2: Generic Imitation Learning

Data: data D, expert π^*, Loss function $F(s)$
Result: learned classifier C, trained policy $\hat{\pi}$

1 Initialise C_0; **for** $n = 1$ **to** N **do**
2 Initialise $E_n = \phi$;
3 $\pi_{Rollin} = RollInPolicy(\pi^*, C_{0...n-1}, n)$;
4 $\pi_{Rollout} = RollOutPolicy(\pi^*, C_{0...n-1}, n)$;
5 **for** $d \in D$ **do**
6 Predict trajectory $\hat{s}_{1:T}$ with π_{Rollin};
7 **for** $\hat{s}_t \in \hat{s}_{1:T}$ **do**
8 **foreach** $a_t^j \in Explore(\hat{s}_t, \pi^*, \pi_{Rollin})$ **do**
9 $\Phi_t^j = \Phi(d, a_t^j, \hat{s}_{1:t})$;
10 Predict $\hat{s}'_{t+1:T}$ with $\pi_{Rollout}$;
11 $L_t^j = F(\hat{s}'_T)$;
12 **foreach** j **do**
13 $ActionCost_t^j = L_t^j - \min_k L_t^k$
14 Add $(\Phi_t, ActionCost_t)$ to E_n;
15 $\hat{\pi}_n, C_n = Train(C_{1...n-1}, E_1 \dots E_n)$;

glish dictionary. The large action space and unbounded T also make beam search difficult to apply since it relies on a fixed length T with commensurability of actions at the same index on different search trajectories.

3 Imitation Learning for Structured Prediction

Imitation learning originated in robotics, training a robot to follow the actions of a human expert (Schaal, 1999; Silver et al., 2008). The robot moves from state to state via actions, generating a trajectory in the same manner as the transition-based parser of Algorithm 1.

In the imitation learning literature, the learning of a policy $\hat{\pi}$ from just the expert generated trajectories is termed "exact imitation".As discussed, it is prone to error propagation, which arises because the implicit assumption of i.i.d. inputs (s_i) during training does not hold. The states in any trajectory are dependent on previous states, and on the policy used. A number of imitation learning algorithms have been proposed to mitigate error propagation, and share a common structure shown in Algorithm 2. Table 2 highlights some key differences between them.

The general algorithm firstly applies a policy π_{RollIn} (usually the expert, π^*, to start) to the data instances to generate a set of 'RollIn' trajectories in line 6 (we adopt the terminology of 'RollIn' and 'RollOut' trajectories from Chang et al. (2015)). Secondly a number of 'what if' scenarios are considered, in which a different action a_t^j is taken from a given s_t instead of the actual a_t in the RollIn trajectory (line 8). Each of these exploratory actions generates a RollOut trajectory (line 10) to a terminal state, for which a loss (L) is calculated using a loss function, $F(s_T^j)$, defined on the terminal states. For a number of different exploratory actions taken from a state s_t on a RollIn trajectory, the action cost (or relative loss) of each is calculated (line 13). Finally the generated $\langle s_t, a_t^j, ActionCost_t^j \rangle$ data are used to train a classifier, using any cost-sensitive classification (CSC) method (line 15). New π_{RollIn} and $\pi_{RollOut}$ are generated, and the process repeated over a number of iterations. In general the starting expert policy is progressively removed in each iteration, so that the training data moves closer and closer to the distribution encountered by just the trained classifier. This is required to reduce error propagation. For a general imitation learning algorithm we need to specify:

- the policy to generate the RollIn trajectory (the *RollInPolicy*)
- the policy to generate RollOut trajectories, including rules for interpolation of learned and expert policies (the *RollOutPolicy*)
- which one-step deviations to explore with a RollOut (the *Explore* function)
- how RollOut data are used in the classification learning algorithm to generate $\hat{\pi}_i$. (within the *Train* function)

Exact Imitation can be considered a single iteration of this algorithm, with π_{RollIn} equal to the expert policy, and a 0-1 binary loss for F (0 loss for $\pi^*(s_t)$, the expert action, and a loss of 1 for any other action); all one-step deviations from the expert trajectory are considered without explicit RollOut to a terminal state.

In **SEARN** (Daumé III et al., 2009), one of the first imitation learning algorithms in this framework, the π_{RollIn} and $\pi_{RollOut}$ policies are identical within each iteration, and are a stochastic blend of the expert and all classifiers trained in previous iterations. The $Explore$ function considers every possible one-step deviation from the RollIn trajectories, with a full RollOut to a terminal state. The

Algorithm	$\hat{\pi}$	RollIn	RollOut	Explore	Train
Exact Imitation	Deterministic	Expert only	None. 0/1 expert loss	All 1-step	E_1 only
SEARN	Stochastic	Mixture	Mixture, step-level stochastic	All 1-step	E_n only
LOLS	Deterministic	Learned only	Mixture, trajectory-level stoch.	All 1-step	$E_1 \dots E_n$
SCB-LOLS	Deterministic	Learned only	Mixture, trajectory-level stoch.	Random	$E_1 \dots E_n$
SMILE	Stochastic	Mixture	None. 0/1 expert loss	All 1-step	$E_1 \dots E_n$
DAGGER	Deterministic	Mixture	None. 0/1 expert loss	All 1-step	$E_1 \dots E_n$
V-DAGGER	Deterministic	Mixture	Mixture, step-level stochastic	All 1-step	$E_1 \dots E_n$
AGGREVATE	Deterministic	Learned only	Expert only	Random	$E_1 \dots E_n$

Table 2: Comparison of selected aspects of Imitation Learning algorithms.

$Train$ function uses only the training data from the most recent iteration (E_n) to train C_n.

LOLS extends this work to provide a deterministic learned policy (Chang et al., 2015), with $\hat{\pi}_n = C_n$. At each iteration $\hat{\pi}_n$ is trained on *all* previously gathered data $E_{1\dots n}$; π_{RollIn} uses the latest classifier $\hat{\pi}_{n-1}$, and each RollOut uses the same policy for all actions in the trajectory; either π^* with probability β, or $\hat{\pi}_{n-1}$ otherwise. Both LOLS and SEARN use an exhaustive search of alternative actions as an $Explore$ function. Chang et al. (2015) consider Structured Contextual Bandits (SCB) as a partial information case, the SCB modification of LOLS permits only one cost function call per RollIn (received from the external environment), so exhaustive RollOut exploration at each step is not possible. SCB-LOLS $Explore$ picks a single step $t \in \{1 \dots T\}$ at *random* at which to make a *random* single-step deviation.

Another strand of work uses only the expert policy when calculating the action cost. Ross and Bagnell (2010) introduce **SMILE**, and later **DAGGER** (Ross et al., 2011). These do not RollOut as such, but as in exact imitation consider all one-step deviations from the RollIn trajectory and obtain a $0/1$ action cost for each by asking the expert what it would do in that state. At the nth iteration the training trajectories are generated from an interpolation of π^* and $\hat{\pi}_{n-1}$, with the latter progressively increasing in importance; π^* is used with probability (1-δ)$^{n-1}$ for some decay rate δ. $\hat{\pi}_n$ is trained using all $E_{1\dots n}$. Ross et al. (2011) discuss and reject calculating an action cost by completing a RollOut from each one-step deviation to a terminal state. Three reasons given are:

1. Lack of real-world applicability, for example in robotic control.
2. Lack of knowledge of the final loss function, if we just have the expert's actions.
3. Time spent calculating RollOuts and calling the expert.

Ross and Bagnell (2014) *do* incorporate RollOuts to calculate an action cost in their **AGGREVATE** algorithm. These RollOuts use the expert policy only, and allow a cost-sensitive classifier to be trained that can learn that some mistakes are more serious than others. As with DAGGER, the trained policy cannot become better than the expert.

V-DAGGER is the variant proposed by Vlachos and Clark (2014) in a semantic parsing task. It is the same as DAGGER, but with RollOuts using the same policy as RollIn. For both V-DAGGER and SEARN, the stochasticity of the RollOut means that a number of independent samples are taken for each one-step deviation to reduce the variance of the action cost, and noise in the training data. This noise reduction comes at the expense of the time needed to compute additional RollOuts.

4 Adapting imitation learning to AMR

Algorithms with full RollOuts have particular value in the absence of an optimal (or near-optimal) expert able to pick the best action from any state. If we have a suitable loss function, then the benefit of RollOuts may become worth the computation expended on them. For AMR parsing we have both a loss function in Smatch, and the ability to generate arbitrary RollOuts.

We therefore use a heuristic expert. This reduces the computational cost at the expense of not always predicting the best action. An expert needs an alignment between gold AMR nodes and tokens in the parse-tree or sentence to determine the actions to convert to one from the other. These alignments are not provided in the gold AMR, and our expert uses the AMR node to token alignments of JAMR (Flanigan et al., 2014). These alignments are not trained, but generated using regex and string matching rules. However, trajectories are in the range 50-200 actions for most training sentences, which combined with the size of $|\mathcal{A}|$ makes an exhaustive search of all one-step deviations expensive. Compare this to unlabeled shift-

reduce parsers with 4 actions, or POS tagging with $|\mathcal{A}| \sim 30$.

4.1 Targeted exploration

To reduce this cost we note that exploring RollOuts for all possible alternative actions can be uninformative when the learned and expert policies agree on an action and none of the other actions score highly with the learned policy. Extending this insight we modify the $Explore$ function in Algorithm 2 to only consider the expert action, plus all actions scored by the current learned policy that are within a threshold τ of the score for the best rated action. In the first iteration, when there is no current learned policy, we pick a number of actions (usually 10) at random for exploration. Both SCB-LOLS and AGGREVATE use partial exploration, but select the step $t \in 1 \dots T$, and the action a_t at random. Here we optimise computational resources by directing the search to areas for which the trained policy is least sure of the optimal action, or disagrees with the expert.

Using imitation learning to address error propagation of transition-based parsing provides theoretical benefit from ensuring the distribution of s_t, a_t in the training data is consistent with the distribution on unseen test data. Using RollOuts that mix expert and learned policies additionally permits the learned policy to exceed the performance of a poor expert. Incorporating targeted exploration strategies in the $Explore$ function makes this computationally feasible.

4.2 Noise Reduction

Different samples for a RollOut trajectory using V-DAGGER or SEARN can give very different terminal states s_T (the final AMR graph) from the same starting s_t and a_t due to the step-level stochasticity. The resultant high variance in the reward signal hinders effective learning. Daumé III et al. (2009) have a similar problem, and note that an approximate cost function outperforms single Monte Carlo sampling, "likely due to the noise induced following a single sample".

To control noise we use the α-bound discussed by Khardon and Wachman (2007). This excludes a training example (i.e. an individual tuple s_i, a_i) from future training once it has been misclassified α times in training. We find that this simple idea avoids the need for multiple RollOut samples.

An attraction of LOLS is that it randomly selects either expert or learned policy for each RollOut, and then applies this consistently to the whole trajectory. Using LOLS should reduce noise without increasing the sample size. Unfortunately the unbounded T of our transition system leads to problems if we drop the expert from the RollIn or RollOut policy mix too quickly, with many trajectories never terminating. Ultimately $\hat{\pi}$ learns to stop doing this, but even with targeted exploration training time is prohibitive and our LOLS experiments failed to provide results. We find that V-DAGGER with an α-bound works as a good compromise, keeping the expert involved in RollIn, and speeding up learning overall.

Another approach we try is a form of *focused costing* (Vlachos and Craven, 2011). Instead of using the learned policy for $\beta\%$ of steps in the RollOut, we use it for the first b steps, and then revert to the expert. This has several potential advantages: the heuristic expert is faster than scoring all possible actions; it focuses the impact of the exploratory step on immediate actions/effects so that mistakes $\hat{\pi}$ makes on a distant part of the graph do not affect the action cost; it reduces noise for the same reason. We increase b in each iteration so that the expert is asymptotically removed from RollOuts, a function otherwise supported by the decay parameter, δ.

4.3 Transition System adaptations

Applying imitation learning to a transition system with unbounded T can and does cause problems in early iterations, with RollIn or RollOut trajectories failing to complete while the learned policy, $\hat{\pi}$, is still relatively poor. To ensure every trajectory completes we add action constraints to the system. These avoid the most pathological scenarios, such as disallowing a Reattach of a previously Reattached sub-graph. These constraints are only needed in the first few iterations until $\hat{\pi}$ learns, via the action costs, to avoid these scenarios. They are listed in the Supplemental Material. As a final failsafe we insert a hard-stop on any trajectory once $T > 300$.

To address the size of $|A|$, we only consider a subset of AMR concepts when labelling a node. Wang et al. (2015b) use all concepts that occur in the training data in the same sentence as the lemma of the node, leading to hundreds or thousands of possible actions from some states. We use the smaller set of concepts that were assigned by the expert to the lemma of the current node any-

	Exact Imitation			Imitation Learning				
Experiment	No α	α=1	α-Gain	No α	α=1	IL Gain (α)	IL Gain (No α)	Total Gain
AROW, C=10	65.5	66.8	1.3	65.5	67.4	0.6	0.0	1.9
AROW, C=100	66.4	66.6	0.2	66.4	67.7	1.1	0.0	1.3
AROW, C=1000	66.4	67.0	0.6	66.5	68.2	1.2	0.1	1.8
PA, C=100	66.7	66.5	-0.2	67.2	68.7	2.2	0.5	2.0
Perceptron	65.5	65.3	-0.2	66.6	68.6	3.3	1.1	3.1

Table 3: DAGGER with α-bound. All figures are F-Scores on the validation set. 5 iterations of classifier training take place after each DAgger iteration. A decay rate (δ) for π^* of 0.3 was used.

where in the training data. We obtain these assignments from an initial application of the expert to the full training data.

We add actions to use the actual word or lemma of the current node to increase generalisation, plus an action to append '-01' to 'verbify' an unseen word. This is similar to the work of Werling et al. (2015) in word to AMR concept mapping, and is useful since 38% of the test AMR concepts do not exist in the training data (Flanigan et al., 2014).

Full details of the heuristics of the expert policy, features used and pre-processing are in Supplemental Material. All code is available at `https://github.com/hopshackle/dagger-AMR`.

4.4 Naïve Smatch as Loss Function

Smatch (Cai and Knight, 2013) uses heuristics to control the combinatorial explosion of possible mappings between the input and output graphs, but is still too computationally expensive to be calculated for every RollOut during training. We retain Smatch for reporting all final results, but use 'Naïve Smatch' as an approximation during training. This skips the combinatorial mapping of nodes between predicted and target AMR graphs. Instead, for each graph we compile a list of:

- Node labels, e.g. `name`
- Node-Edge-Node label concatenations, e.g. `leave-01:ARG0:room`
- Node-Edge label concatenations, e.g. `leave-01:ARG0`, `ARG0:room`

The loss is the number of entries that appear in only one of the lists. We do not convert to an F_1 score, as retaining the absolute number of mistakes is proportional to the size of the graph.

The flexibility of the transition system means multiple different actions from a given state s_i can lead, via different RollOut trajectories, to the same target s_T. This can result in many actions having the best action cost, reducing the signal in the training data and giving poor learning. To encourage short trajectories we break these ties with a penalty of $T/5$ to Naïve Smatch. Multiple routes of the same length still exist, and are preferred equally. Note that the ordering of the stack of dependency tree nodes in the transition system means we start at leaf nodes and move up the tree. This prevents sub-components of the output AMR graph being produced in an arbitrary order.

5 Experiments

The main dataset used is the newswire (proxy) section of LDC2014T12 (Knight et al., 2014). The data from years 1995-2006 form the training data, with 2007 as the validation set and 2008 as the test set. The data split is the same as that used by Flanigan et al. (2014) and Wang et al. (2015b). [1]

We first assess the impact of noise reduction using the alpha bound, and report these experiments without Rollouts (i.e. using DAGGER) to isolate the effect of noise reduction. Table 3 summarises results using exact imitation and DAGGER with the α-bound set to discard a training instance after one misclassification. This is the most extreme setting, and the one that gave best results. We try AROW (Crammer et al., 2013), Passive-Aggressive (PA) (Crammer et al., 2006), and perceptron (Collins, 2002) classifiers, with averaging in all cases. We see a benefit from the α-bound for exact imitation only with AROW, which is more noise-sensitive than PA or the simple perceptron. With DAGGER there is a benefit for all classifiers. In all cases the α-bound and DAGGER are synergistic; without the α-bound imitation learning works less well, if at all. α=1 was the optimal setting, with lesser benefit observed for larger values.

We now turn our attention to targeted exploration and focused costing, for which we use V-DAGGER as explained in section 4. For all V-

[1] Formally Flanigan et al. (2014; Wang et al. (2015b) use the pre-release version of this dataset (LDC2013E117). Werling et al. (2015) conducted comparative tests on the two versions, and found only a very minor changes of 0.1 to 0.2 points of F-score when using the final release.

Authors	Algorithmic Approach	R	P	F
Flanigan et al. (2014)	Concept identification with semi-markov model followed by optimisation of constrained graph that contains all of these.	0.52	0.66	0.58
Werling et al. (2015)	As Flanigan et al. (2014), with enhanced concept identification	0.59	0.66	0.62
Wang et al. (2015b)	Single stage using transition-based parsing algorithm	0.62	0.64	0.63
Pust et al. (2015)	Single stage System-Based Machine Translation	-	-	0.66
Peng et al. (2015)	Hyperedge replacement grammar	0.57	0.59	0.58
Artzi et al. (2015)	Combinatory Categorial Grammar induction	0.66	0.67	0.66
Wang et al. (2015a)	Extensions to action space and features in Wang et al. (2015b)	0.69	0.71	0.70
This work	Imitation Learning with transition-based parsing	0.68	0.73	0.70

Table 4: Comparison of previous work on the AMR task. R, P and F are Recall, Precision and F-Score.

DAGGER experiments we use AROW with regularisation parameter C=1000, and δ=0.3.

Figure 2 shows results by iteration of reducing the number of RollOuts explored. Only the expert action, plus actions that score close to the best-scoring action (defined by the threshold) are used for RollOuts. Using the action cost information from RollOuts does surpass simple DAGGER, and unsurprisingly more exploration is better.

Figure 3 shows the same data, but by total computational time spent[2]. This adjusts the picture, as small amounts of exploration give a faster benefit, albeit not always reaching the same peak performance. As a baseline, three iterations of V-DAGGER without targeted exploration (threshold = ∞) takes 9600 minutes on the same hardware to give an F-Score of 0.652 on the validation set.

Figure 4 shows the improvement using focused costing. The 'n/m' setting sets b, the number of initial actions taken by $\hat{\pi}$ in a RollOut to n, and then increases this by m at each iteration. We gain an increase of 2.9 points from 0.682 to 0.711. In all the settings tried, focused costing improves the results, and requires progressive removal of the expert to achieve the best score.

We use the classifier from the Focused Costing 5/5 run to achieve an F-Score on the held-out test set of 0.70, equal to the best published result so far (Wang et al., 2015a). Our gain of 4.7 points from imitation learning over standard transition-based parsing is orthogonal to that of Wang et al. (2015a) using exact imitation with additional trained analysers; they experience a gain of 2 points from using a Charniak parser (Charniak and Johnson, 2005) trained on the full OntoNotes corpus instead of the Stanford parser used here and in Wang et al. (2015b), and a further gain of 2 points from a semantic role labeller. Table 4 lists previous AMR work on the same dataset.

[2]experiments were run on 8-core Google Cloud n1-highmem-8 machines.

	Validation F-Score			Test F-Score	
Dataset	EI	D	V-D	V-D	Rao et al
proxy	0.670	0.686	0.704	0.70	0.61
dfa	0.495	0.532	0.546	0.50	0.44
bolt	0.456	0.468	0.524	0.52	0.46
xinhua	0.598	0.623	0.683	0.62	0.52
lpp	0.540	0.546	0.564	0.55	0.52

Table 5: Comparison of Exact Imitation (EI), DAGGER (D), V-DAGGER (V-D) on all components of the LDC2014T12 corpus.

Using DAGGER with this system we obtained an F-Score of 0.60 in the Semeval 2016 task on AMR parsing, one standard deviation above the mean of all entries. (Goodman et al., 2016)

Finally we test on all components of the LDC2014T12 corpus as shown in Table 5, which include both newswire and weblog data, as well as the freely available AMRs for *The Little Prince*, (lpp)[3]. For each we use exact imitation, DAGGER, and V-DAGGER on the train/validation/splits specified in the corpus. In all cases, imitation learning without RollOuts (DAGGER) improves on exact imitation, and incorporating RollOuts (V-DAGGER) provides an additional benefit. Rao et al. (2015) use SEARN on the same datasets, but with a very different transition system. We show their results for comparison.

Our expert achieves a Smatch F-Score of 0.94 on the training data. This explains why DAGGER, which assumes a good expert, is effective. Introducing RollOuts provides additional theoretical benefits from a non-decomposable loss function that can take into account longer-term impacts of an action. This provides much more information than the 0/1 binary action cost in DAGGER, and we can use Naïve Smatch as an approximation to our actual objective function during training. This informational benefit comes at the cost of increased noise and computational expense, which we control with targeted exploration and focused

[3]http://amr.isi.edu/download.html

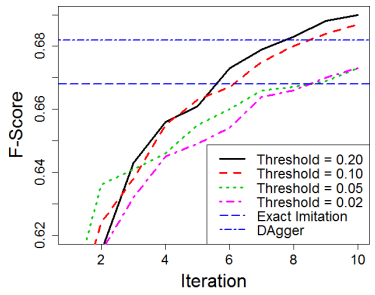


Figure 2: Targeted exploration with V-DAGGER by iteration.

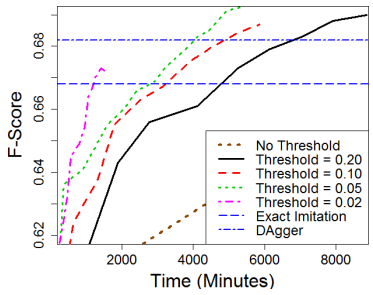


Figure 3: Targeted exploration with V-DAGGER by time.

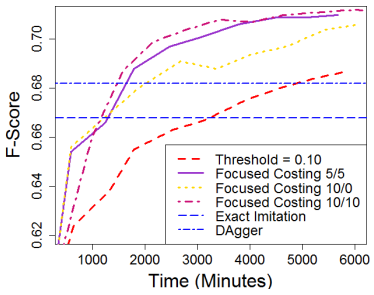


Figure 4: Focused costing with V-DAGGER. All runs use threshold of 0.10.

costing. We gain 2.7 points in F-Score, at the cost of 80-100x more computation. In problems with a less good expert, the gain from exploration could be much greater. Similarly, if designing an expert for a task is time-consuming, then it may be a better investment to rely on exploration with a poor expert to achieve the same result.

6 Related Work

Other strategies have been used to mitigate the error propagation problem in transition-based parsing. A common approach is to use beam search through state-space for each action choice to find a better approximation of the long-term score of the action, e.g. Zhang and Clark (2008). Goldberg and Elhadad (2010) remove the determinism of the sequence of actions to create *easy-first* parsers, which postpone uncertain, error-prone decisions until more information is available. This contrasts with working inflexibly left-to-right along a sentence, or bottom-to-top up a tree.

Goldberg and Nivre (2012) introduce *dynamic* experts that are *complete* in that they will respond from any state, not just those on the perfect trajectory assuming no earlier mistakes; any expert used with an imitation learning algorithm needs to be *complete* in this sense. Their algorithm takes exploratory steps off the expert trajectory to augment the training data collected in a fashion very similar to DAGGER.

Honnibal et al. (2013) use a *non-monotonic* parser that allows actions that are inconsistent with previous actions. When such an action is taken it amends the results of previous actions to ensure post-hoc consistency. Our parser is non-monotonic, and we have the same problem encountered by Honnibal et al. (2013) with many different actions from a state s_i able to reach the target s_T, following different "paths up the mountain". This leads to poor learning. To resolve this with fixed T they break ties with a monotonic parser, so that actions that do not require later correction are scored higher in the training data. In our variable T environment, adding a penalty to the size of T is sufficient (section 4.4).

Vlachos and Clark (2014) use V-DAGGER to give a benefit of 4.8 points of F-Score in a domain-specific semantic parsing problem similar to AMR. Their expert is sub-optimal, with no information on alignment between words in the input sentence, and nodes in the target graph. The parser learns to link words in the input to one of the 35 node types, with the 'expert' policy aligning completely at random. This is infeasible with AMR parsing due to the much larger vocabulary.

7 Conclusions

Imitation learning provides a total benefit of 4.5 points with our AMR transition-based parser over exact imitation. This is a more complex task than many previous applications of imitation learning, and we found that noise reduction was an essential pre-requisite. Using a simple 0/1 binary action cost using a heuristic expert provided a benefit of 1.8, with the remaining 2.7 points coming from RollOuts with targeted exploration, focused costing and a non-decomposable loss function that was a better approximation to our objective.

We have considered imitation learning algorithms as a toolbox that can be tailored to fit the characteristics of the task. An unbounded T meant that the LOLS RollIn was not ideal, but this could be modified to slow the loss of influence of the expert policy. We anticipate the approaches that we have found useful in the case of AMR to reduce the impact of noise, efficiently support large action spaces with targeted exploration, and cope with unbounded trajectories in the transition system will be of relevance to other structured prediction tasks.

Acknowledgments

Andreas Vlachos is supported by the EPSRC grant Diligent (EP/M005429/1) and Jason Naradowsky by a Google Focused Research award. We would also like to thank our anonymous reviewers for many comments that helped improve this paper.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal, September. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria, August. Association for Computational Linguistics.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *ACL (2)*, pages 748–752.

Kai-wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015. Learning to search better than your teacher. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2058–2066.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2013. Adaptive regularization of weight vectors. *Mach Learn*, 91:155–187.

Deborah A Dahl, Madeleine Bates, Michael Brown, William Fisher, Kate Hunicke-Smith, David Pallett, Christine Pao, Alexander Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *Proceedings of the workshop on Human Language Technology*, pages 43–48. Association for Computational Linguistics.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Jeffrey Flanigan, Sam Thomson, Jaime Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1426–1436. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *COLING*, pages 959–976.

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

James Goodman, Andreas Vlachos, and Jason Naradowsky. 2016. Ucl+sheffield at semeval-2016 task 8: Imitation learning for amr parsing with an alpha-bound. In *Proceedings of the 10th International Workshop on Semantic Evaluation*.

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *Empirical Methods in Natural Language Processing*.

Matthew Honnibal, Yoav Goldberg, and Mark Johnson. 2013. A non-monotonic arc-eager transition system for dependency parsing. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 163–172. Citeseer.

Roni Khardon and Gabriel Wachman. 2007. Noise tolerant variants of the perceptron algorithm. *The journal of machine learning research*, 8:227–248.

Kevin Knight, Laura Baranescu, Claire Bonial, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, and Nathan Schneider. 2014. Abstract meaning representation (amr) annotation release 1.0. *Linguistic Data Consortium Catalog. LDC2014T12*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Ryan T McDonald and Joakim Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP-CoNLL*, pages 122–131.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. *CoNLL 2015*, page 32.

Michael Pust, Ulf Hermjakob, Kevin Knight, Daniel Marcu, and Jonathan May. 2015. Using syntax-based machine translation to parse english into abstract meaning representation. *arXiv preprint arXiv:1504.06665*.

Sudha Rao, Yogarshi Vyas, Hal Daume III, and Philip Resnik. 2015. Parser for abstract meaning representation using learning to search. *arXiv preprint arXiv:1510.07586*.

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *13th International Conference on Artificial Intelligence and Statistics*, pages 661–668.

Stephane Ross and J Andrew Bagnell. 2014. Reinforcement and imitation learning via interactive no-regret learning. *arXiv preprint arXiv:1406.5979*.

Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *14th International Conference on Artificial Intelligence and Statistics*, volume 15, pages 627–635.

Stefan Schaal. 1999. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242.

David Silver, James Bagnell, and Anthony Stentz. 2008. High performance outdoor navigation from overhead data using imitation learning. *Robotics: Science and Systems IV, Zurich, Switzerland*.

Andreas Vlachos and Stephen Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics*, 2:547–559.

Andreas Vlachos and Mark Craven. 2011. Search-based structured prediction applied to biomedical event extraction. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 49–57. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 857–862, Beijing, China, July. Association for Computational Linguistics.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015b. A transition-based algorithm for amr parsing. *North American Association for Computational Linguistics, Denver, Colorado*.

Keenon Werling, Gabor Angeli, and Christopher D. Manning. 2015. Robust subgraph generation improves abstract meaning representation parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 982–991, Beijing, China, July. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055.

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

Data Recombination for Neural Semantic Parsing

Robin Jia
Computer Science Department
Stanford University
robinjia@stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

Abstract

Modeling crisp logical regularities is crucial in semantic parsing, making it difficult for neural models with no task-specific prior knowledge to achieve good results. In this paper, we introduce data recombination, a novel framework for injecting such prior knowledge into a model. From the training data, we induce a high-precision synchronous context-free grammar, which captures important conditional independence properties commonly found in semantic parsing. We then train a sequence-to-sequence recurrent network (RNN) model with a novel attention-based copying mechanism on datapoints sampled from this grammar, thereby teaching the model about these structural properties. Data recombination improves the accuracy of our RNN model on three semantic parsing datasets, leading to new state-of-the-art performance on the standard GeoQuery dataset for models with comparable supervision.

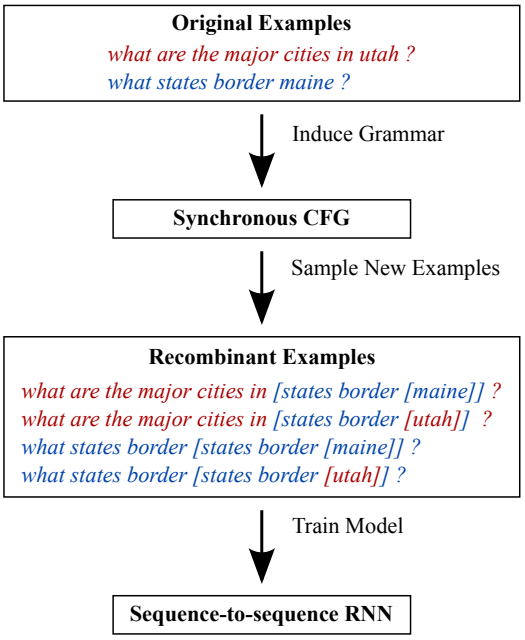


Figure 1: An overview of our system. Given a dataset, we induce a high-precision synchronous context-free grammar. We then sample from this grammar to generate new "recombinant" examples, which we use to train a sequence-to-sequence RNN.

1 Introduction

Semantic parsing—the precise translation of natural language utterances into logical forms—has many applications, including question answering (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Liang et al., 2011; Berant et al., 2013), instruction following (Artzi and Zettlemoyer, 2013b), and regular expression generation (Kushman and Barzilay, 2013). Modern semantic parsers (Artzi and Zettlemoyer, 2013a; Berant et al., 2013) are complex pieces of software, requiring hand-crafted features, lexicons, and grammars.

Meanwhile, recurrent neural networks (RNNs) have made swift inroads into many structured prediction tasks in NLP, including machine translation (Sutskever et al., 2014; Bahdanau et al., 2014) and syntactic parsing (Vinyals et al., 2015b; Dyer et al., 2015). Because RNNs make very few domain-specific assumptions, they have the potential to succeed at a wide variety of tasks with minimal feature engineering. However, this flexibility also puts RNNs at a disadvantage compared to standard semantic parsers, which can generalize naturally by leveraging their built-in awareness of logical compositionality.

In this paper, we introduce data recombination, a generic framework for declaratively inject-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 12–22,
Berlin, Germany, August 7-12, 2016.

GEO
x: "*what is the population of iowa ?*"
y:
```
_answer ( NV , (
  _population ( NV , V1 ) , _const (
    V0 , _stateid ( iowa ) ) ) )
```
ATIS
x: "*can you list all flights from chicago to milwaukee*"
y:
```
( _lambda $0 e ( _and
  ( _flight $0 )
  ( _from $0 chicago :  _ci )
  ( _to $0 milwaukee :  _ci ) ) )
```
Overnight
x: "*when is the weekly standup*"
y:
```
( call listValue ( call
    getProperty meeting.weekly_standup
    ( string start_time ) ) )
```

Figure 2: One example from each of our domains. We tokenize logical forms as shown, thereby casting semantic parsing as a sequence-to-sequence task.

ing prior knowledge into a domain-general structured prediction model. In data recombination, prior knowledge about a task is used to build a high-precision generative model that expands the empirical distribution by allowing fragments of different examples to be combined in particular ways. Samples from this generative model are then used to train a domain-general model. In the case of semantic parsing, we construct a generative model by inducing a synchronous context-free grammar (SCFG), creating new examples such as those shown in Figure 1; our domain-general model is a sequence-to-sequence RNN with a novel attention-based copying mechanism. Data recombination boosts the accuracy of our RNN model on three semantic parsing datasets. On the GEO dataset, data recombination improves test accuracy by 4.3 percentage points over our baseline RNN, leading to new state-of-the-art results for models that do not use a seed lexicon for predicates.

2 Problem statement

We cast semantic parsing as a sequence-to-sequence task. The input utterance x is a sequence of words $x_1, \ldots, x_m \in \mathcal{V}^{(\text{in})}$, the input vocabulary; similarly, the output logical form y is a sequence of tokens $y_1, \ldots, y_n \in \mathcal{V}^{(\text{out})}$, the output vocabulary. A linear sequence of tokens might appear to lose the hierarchical structure of a logical form, but there is precedent for this choice: Vinyals et al. (2015b) showed that an RNN can reliably predict tree-structured outputs in a linear fashion.

We evaluate our system on three existing semantic parsing datasets. Figure 2 shows sample input-output pairs from each of these datasets.

- **GeoQuery** (GEO) contains natural language questions about US geography paired with corresponding Prolog database queries. We use the standard split of 600 training examples and 280 test examples introduced by Zettlemoyer and Collins (2005). We preprocess the logical forms to De Brujin index notation to standardize variable naming.
- **ATIS** (ATIS) contains natural language queries for a flights database paired with corresponding database queries written in lambda calculus. We train on 4473 examples and evaluate on the 448 test examples used by Zettlemoyer and Collins (2007).
- **Overnight** (OVERNIGHT) contains logical forms paired with natural language paraphrases across eight varied subdomains. Wang et al. (2015) constructed the dataset by generating all possible logical forms up to some depth threshold, then getting multiple natural language paraphrases for each logical form from workers on Amazon Mechanical Turk. We evaluate on the same train/test splits as Wang et al. (2015).

In this paper, we only explore learning from logical forms. In the last few years, there has an emergence of semantic parsers learned from denotations (Clarke et al., 2010; Liang et al., 2011; Berant et al., 2013; Artzi and Zettlemoyer, 2013b). While our system cannot directly learn from denotations, it could be used to rerank candidate derivations generated by one of these other systems.

3 Sequence-to-sequence RNN Model

Our sequence-to-sequence RNN model is based on existing attention-based neural machine translation models (Bahdanau et al., 2014; Luong et al., 2015a), but also includes a novel attention-based copying mechanism. Similar copying mechanisms have been explored in parallel by Gu et al. (2016) and Gulcehre et al. (2016).

3.1 Basic Model

Encoder. The encoder converts the input sequence $x_1, \ldots, x_m$ into a sequence of *context-*

sensitive embeddings $b_1, \ldots, b_m$ using a bidirectional RNN (Bahdanau et al., 2014). First, a word embedding function $\phi^{(\text{in})}$ maps each word x_i to a fixed-dimensional vector. These vectors are fed as input to two RNNs: a forward RNN and a backward RNN. The forward RNN starts with an initial hidden state h_0^{F}, and generates a sequence of hidden states $h_1^{\text{F}}, \ldots, h_m^{\text{F}}$ by repeatedly applying the recurrence

$$h_i^{\text{F}} = \text{LSTM}(\phi^{(\text{in})}(x_i), h_{i-1}^{\text{F}}). \quad (1)$$

The recurrence takes the form of an LSTM (Hochreiter and Schmidhuber, 1997). The backward RNN similarly generates hidden states $h_m^{\text{B}}, \ldots, h_1^{\text{B}}$ by processing the input sequence in reverse order. Finally, for each input position i, we define the context-sensitive embedding b_i to be the concatenation of h_i^{F} and h_i^{B}

Decoder. The decoder is an attention-based model (Bahdanau et al., 2014; Luong et al., 2015a) that generates the output sequence $y_1, \ldots, y_n$ one token at a time. At each time step j, it writes y_j based on the current hidden state s_j, then updates the hidden state to s_{j+1} based on s_j and y_j. Formally, the decoder is defined by the following equations:

$$s_1 = \tanh(W^{(s)}[h_m^{\text{F}}, h_1^{\text{B}}]). \quad (2)$$

$$e_{ji} = s_j^\top W^{(a)} b_i. \quad (3)$$

$$\alpha_{ji} = \frac{\exp(e_{ji})}{\sum_{i'=1}^{m} \exp(e_{ji'})}. \quad (4)$$

$$c_j = \sum_{i=1}^{m} \alpha_{ji} b_i. \quad (5)$$

$$P(y_j = w \mid x, y_{1:j-1}) \propto \exp(U_w[s_j, c_j]). \quad (6)$$

$$s_{j+1} = \text{LSTM}([\phi^{(\text{out})}(y_j), c_j], s_j). \quad (7)$$

When not specified, i ranges over $\{1, \ldots, m\}$ and j ranges over $\{1, \ldots, n\}$. Intuitively, the α_{ji}'s define a probability distribution over the input words, describing what words in the input the decoder is focusing on at time j. They are computed from the unnormalized attention scores e_{ji}. The matrices $W^{(s)}$, $W^{(a)}$, and U, as well as the embedding function $\phi^{(\text{out})}$, are parameters of the model.

3.2 Attention-based Copying

In the basic model of the previous section, the next output word y_j is chosen via a simple softmax over all words in the output vocabulary. However, this model has difficulty generalizing to the long tail of entity names commonly found in semantic parsing datasets. Conveniently, entity names in the input often correspond directly to tokens in the output (e.g., "*iowa*" becomes `iowa` in Figure 2).[1]

To capture this intuition, we introduce a new attention-based copying mechanism. At each time step j, the decoder generates one of two types of actions. As before, it can write any word in the output vocabulary. In addition, it can copy any input word x_i directly to the output, where the probability with which we copy x_i is determined by the attention score on x_i. Formally, we define a latent action a_j that is either $\texttt{Write}[w]$ for some $w \in \mathcal{V}^{(\text{out})}$ or $\texttt{Copy}[i]$ for some $i \in \{1, \ldots, m\}$. We then have

$$P(a_j = \texttt{Write}[w] \mid x, y_{1:j-1}) \propto \exp(U_w[s_j, c_j]), \quad (8)$$

$$P(a_j = \texttt{Copy}[i] \mid x, y_{1:j-1}) \propto \exp(e_{ji}). \quad (9)$$

The decoder chooses a_j with a softmax over all these possible actions; y_j is then a deterministic function of a_j and x. During training, we maximize the log-likelihood of y, marginalizing out a.

Attention-based copying can be seen as a combination of a standard softmax output layer of an attention-based model (Bahdanau et al., 2014) and a Pointer Network (Vinyals et al., 2015a); in a Pointer Network, the only way to generate output is to copy a symbol from the input.

4 Data Recombination

4.1 Motivation

The main contribution of this paper is a novel data recombination framework that injects important prior knowledge into our oblivious sequence-to-sequence RNN. In this framework, we induce a high-precision generative model from the training data, then sample from it to generate new training examples. The process of inducing this generative model can leverage any available prior knowledge, which is transmitted through the generated examples to the RNN model. A key advantage of our two-stage approach is that it allows us to declare desired properties of the task which might be hard to capture in the model architecture.

[1] On GEO and ATIS, we make a point not to rely on orthography for non-entities such as "*state*" to `_state`, since this leverages information not available to previous models (Zettlemoyer and Collins, 2005) and is much less language-independent.

Examples
("*what states border texas ?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(texas))))`)
("*what is the highest mountain in ohio ?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV), const(V0, stateid(ohio)))))`)

Rules created by ABSENTITIES
ROOT → ⟨ "*what states border* STATEID *?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(`STATEID`))))`⟩
STATEID → ⟨ "*texas*", `texas` ⟩
ROOT → ⟨ "*what is the highest mountain in* STATEID *?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV),`
`const(V0, stateid(`STATEID`)))))`⟩
STATEID → ⟨"*ohio*", `ohio`⟩

Rules created by ABSWHOLEPHRASES
ROOT → ⟨ "*what states border* STATE *?*", `answer(NV, (state(V0), next_to(V0, NV),` STATE `))`⟩
STATE → ⟨ "*states border texas*", `state(V0), next_to(V0, NV), const(V0, stateid(texas))`⟩
ROOT → ⟨ "*what is the highest mountain in* STATE *?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV),` STATE `)))`⟩

Rules created by CONCAT-2
ROOT → ⟨SENT$_1$ `</s>` SENT$_2$, SENT$_1$ `</s>` SENT$_2$⟩
SENT → ⟨ "*what states border texas ?*",
`answer(NV, (state(V0), next_to(V0, NV), const(V0, stateid(texas))))` ⟩
SENT → ⟨ "*what is the highest mountain in ohio ?*",
`answer(NV, highest(V0, (mountain(V0), loc(V0, NV), const(V0, stateid(ohio)))))` ⟩

Figure 3: Various grammar induction strategies illustrated on GEO. Each strategy converts the rules of an input grammar into rules of an output grammar. This figure shows the base case where the input grammar has rules ROOT → $\langle x, y \rangle$ for each (x, y) pair in the training dataset.

Our approach generalizes data augmentation, which is commonly employed to inject prior knowledge into a model. Data augmentation techniques focus on modeling invariances—transformations like translating an image or adding noise that alter the inputs x, but do not change the output y. These techniques have proven effective in areas like computer vision (Krizhevsky et al., 2012) and speech recognition (Jaitly and Hinton, 2013).

In semantic parsing, however, we would like to capture more than just invariance properties. Consider an example with the utterance "*what states border texas ?*". Given this example, it should be easy to generalize to questions where "*texas*" is replaced by the name of any other state: simply replace the mention of Texas in the logical form with the name of the new state. Underlying this phenomenon is a strong conditional independence principle: the meaning of the rest of the sentence is independent of the name of the state in question. Standard data augmentation is not sufficient to model such phenomena: instead of holding y fixed, we would like to apply simultaneous transformations to x and y such that the new x still maps to the new y. Data recombination addresses this need.

4.2 General Setting

In the general setting of data recombination, we start with a training set $\mathcal{D}$ of (x, y) pairs, which defines the empirical distribution $\hat{p}(x, y)$. We then fit a generative model $\tilde{p}(x, y)$ to $\hat{p}$ which generalizes beyond the support of $\hat{p}$, for example by splicing together fragments of different examples. We refer to examples in the support of $\tilde{p}$ as *recombinant* examples. Finally, to train our actual model $p_\theta(y \mid x)$, we maximize the expected value of $\log p_\theta(y \mid x)$, where (x, y) is drawn from $\tilde{p}$.

4.3 SCFGs for Semantic Parsing

For semantic parsing, we induce a synchronous context-free grammar (SCFG) to serve as the backbone of our generative model $\tilde{p}$. An SCFG consists of a set of production rules $X \to \langle \alpha, \beta \rangle$, where X is a category (non-terminal), and α and β are sequences of terminal and non-terminal symbols. Any non-terminal symbols in α must be aligned to the same non-terminal symbol in β, and vice versa. Therefore, an SCFG defines a set of joint derivations of aligned pairs of strings. In our case, we use an SCFG to represent joint deriva-

tions of utterances x and logical forms y (which for us is just a sequence of tokens). After we induce an SCFG G from $\mathcal{D}$, the corresponding generative model $\tilde{p}(x, y)$ is the distribution over pairs (x, y) defined by sampling from G, where we choose production rules to apply uniformly at random.

It is instructive to compare our SCFG-based data recombination with WASP (Wong and Mooney, 2006; Wong and Mooney, 2007), which uses an SCFG as the actual semantic parsing model. The grammar induced by WASP must have good coverage in order to generalize to new inputs at test time. WASP also requires the implementation of an efficient algorithm for computing the conditional probability $p(y \mid x)$. In contrast, our SCFG is only used to convey prior knowledge about conditional independence structure, so it only needs to have high precision; our RNN model is responsible for boosting recall over the entire input space. We also only need to forward sample from the SCFG, which is considerably easier to implement than conditional inference.

Below, we examine various strategies for inducing a grammar G from a dataset $\mathcal{D}$. We first encode $\mathcal{D}$ as an initial grammar with rules ROOT $\rightarrow \langle x, y \rangle$ for each $(x, y) \in \mathcal{D}$. Next, we will define each grammar induction strategy as a mapping from an input grammar G_{in} to a new grammar G_{out}. This formulation allows us to compose grammar induction strategies (Section 4.3.4).

4.3.1 Abstracting Entities

Our first grammar induction strategy, ABSENTITIES, simply abstracts entities with their types. We assume that each entity e (e.g., `texas`) has a corresponding type $e.t$ (e.g., `state`), which we infer based on the presence of certain predicates in the logical form (e.g. `stateid`). For each grammar rule $X \rightarrow \langle \alpha, \beta \rangle$ in G_{in}, where α contains a token (e.g., "*texas*") that string matches an entity (e.g., `texas`) in β, we add two rules to G_{out}: (i) a rule where both occurrences are replaced with the type of the entity (e.g., `state`), and (ii) a new rule that maps the type to the entity (e.g., STATEID $\rightarrow \langle$"*texas*", `texas`$\rangle$; we reserve the category name STATE for the next section). Thus, G_{out} generates recombinant examples that fuse most of one example with an entity found in a second example. A concrete example from the GEO domain is given in Figure 3.

4.3.2 Abstracting Whole Phrases

Our second grammar induction strategy, ABSWHOLEPHRASES, abstracts both entities and whole phrases with their types. For each grammar rule $X \rightarrow \langle \alpha, \beta \rangle$ in G_{in}, we add up to two rules to G_{out}. First, if α contains tokens that string match to an entity in β, we replace both occurrences with the type of the entity, similarly to rule (i) from ABSENTITIES. Second, if we can infer that the entire expression β evaluates to a set of a particular type (e.g. `state`) we create a rule that maps the type to $\langle \alpha, \beta \rangle$. In practice, we also use some simple rules to strip question identifiers from α, so that the resulting examples are more natural. Again, refer to Figure 3 for a concrete example.

This strategy works because of a more general conditional independence property: the meaning of any semantically coherent phrase is conditionally independent of the rest of the sentence, the cornerstone of compositional semantics. Note that this assumption is not always correct in general: for example, phenomena like anaphora that involve long-range context dependence violate this assumption. However, this property holds in most existing semantic parsing datasets.

4.3.3 Concatenation

The final grammar induction strategy is a surprisingly simple approach we tried that turns out to work. For any $k \geq 2$, we define the CONCAT-k strategy, which creates two types of rules. First, we create a single rule that has ROOT going to a sequence of k SENT's. Then, for each root-level rule ROOT $\rightarrow \langle \alpha, \beta \rangle$ in G_{in}, we add the rule SENT $\rightarrow \langle \alpha, \beta \rangle$ to G_{out}. See Figure 3 for an example.

Unlike ABSENTITIES and ABSWHOLEPHRASES, concatenation is very general, and can be applied to any sequence transduction problem. Of course, it also does not introduce additional information about compositionality or independence properties present in semantic parsing. However, it does generate harder examples for the attention-based RNN, since the model must learn to attend to the correct parts of the now-longer input sequence. Related work has shown that training a model on more difficult examples can improve generalization, the most canonical case being dropout (Hinton et al., 2012; Wager et al., 2013).

```
function TRAIN(dataset D, number of epochs T,
      number of examples to sample n)
    Induce grammar G from D
    Initialize RNN parameters θ randomly
    for each iteration t = 1, ..., T do
        Compute current learning rate η_t
        Initialize current dataset D_t to D
        for i = 1, ..., n do
            Sample new example (x', y') from G
            Add (x', y') to D_t
        end for
        Shuffle D_t
        for each example (x, y) in D_t do
            θ ← θ + η_t ∇ log p_θ(y | x)
        end for
    end for
end function
```

Figure 4: The training procedure with data recombination. We first induce an SCFG, then sample new recombinant examples from it at each epoch.

4.3.4 Composition

We note that grammar induction strategies can be composed, yielding more complex grammars. Given any two grammar induction strategies f_1 and f_2, the composition $f_1 \circ f_2$ is the grammar induction strategy that takes in G_{in} and returns $f_1(f_2(G_{\text{in}}))$. For the strategies we have defined, we can perform this operation symbolically on the grammar rules, without having to sample from the intermediate grammar $f_2(G_{\text{in}})$.

5 Experiments

We evaluate our system on three domains: GEO, ATIS, and OVERNIGHT. For ATIS, we report logical form exact match accuracy. For GEO and OVERNIGHT, we determine correctness based on denotation match, as in Liang et al. (2011) and Wang et al. (2015), respectively.

5.1 Choice of Grammar Induction Strategy

We note that not all grammar induction strategies make sense for all domains. In particular, we only apply ABSWHOLEPHRASES to GEO and OVERNIGHT. We do not apply ABSWHOLEPHRASES to ATIS, as the dataset has little nesting structure.

5.2 Implementation Details

We tokenize logical forms in a domain-specific manner, based on the syntax of the formal language being used. On GEO and ATIS, we disallow copying of predicate names to ensure a fair comparison to previous work, as string matching between input words and predicate names is not commonly used. We prevent copying by prepending underscores to predicate tokens; see Figure 2 for examples.

On ATIS alone, when doing attention-based copying and data recombination, we leverage an external lexicon that maps natural language phrases (e.g., "*kennedy airport*") to entities (e.g., `jfk:ap`). When we copy a word that is part of a phrase in the lexicon, we write the entity associated with that lexicon entry. When performing data recombination, we identify entity alignments based on matching phrases and entities from the lexicon.

We run all experiments with 200 hidden units and 100-dimensional word vectors. We initialize all parameters uniformly at random within the interval $[-0.1, 0.1]$. We maximize the log-likelihood of the correct logical form using stochastic gradient descent. We train the model for a total of 30 epochs with an initial learning rate of 0.1, and halve the learning rate every 5 epochs, starting after epoch 15. We replace word vectors for words that occur only once in the training set with a universal `<unk>` word vector. Our model is implemented in Theano (Bergstra et al., 2010).

When performing data recombination, we sample a new round of recombinant examples from our grammar at each epoch. We add these examples to the original training dataset, randomly shuffle all examples, and train the model for the epoch. Figure 4 gives pseudocode for this training procedure. One important hyperparameter is how many examples to sample at each epoch: we found that a good rule of thumb is to sample as many recombinant examples as there are examples in the training dataset, so that half of the examples the model sees at each epoch are recombinant.

At test time, we use beam search with beam size 5. We automatically balance missing right parentheses by adding them at the end. On GEO and OVERNIGHT, we then pick the highest-scoring logical form that does not yield an executor error when the corresponding denotation is computed. On ATIS, we just pick the top prediction on the beam.

5.3 Impact of the Copying Mechanism

First, we measure the contribution of the attention-based copying mechanism to the model's overall

	GEO	ATIS	OVERNIGHT
No Copying	74.6	69.9	76.7
With Copying	85.0	76.3	75.8

Table 1: Test accuracy on GEO, ATIS, and OVERNIGHT, both with and without copying. On OVERNIGHT, we average across all eight domains.

	GEO	ATIS
Previous Work		
Zettlemoyer and Collins (2007)		**84.6**
Kwiatkowski et al. (2010)	88.9	
Liang et al. (2011)[2]	91.1	
Kwiatkowski et al. (2011)	88.6	82.8
Poon (2013)		83.5
Zhao and Huang (2015)	88.9	84.2
Our Model		
No Recombination	85.0	76.3
ABSENTITIES	85.4	79.9
ABSWHOLEPHRASES	87.5	
CONCAT-2	84.6	79.0
CONCAT-3		77.5
AWP + AE	88.9	
AE + C2		78.8
AWP + AE + C2	**89.3**	
AE + C3		83.3

Table 2: Test accuracy using different data recombination strategies on GEO and ATIS. AE is ABSENTITIES, AWP is ABSWHOLEPHRASES, C2 is CONCAT-2, and C3 is CONCAT-3.

performance. On each task, we train and evaluate two models: one with the copying mechanism, and one without. Training is done without data recombination. The results are shown in Table 1.

On GEO and ATIS, the copying mechanism helps significantly: it improves test accuracy by 10.4 percentage points on GEO and 6.4 points on ATIS. However, on OVERNIGHT, adding the copying mechanism actually makes our model perform slightly worse. This result is somewhat expected, as the OVERNIGHT dataset contains a very small number of distinct entities. It is also notable that both systems surpass the previous best system on OVERNIGHT by a wide margin.

We choose to use the copying mechanism in all subsequent experiments, as it has a large advantage in realistic settings where there are many distinct entities in the world. The concurrent work of Gu et al. (2016) and Gulcehre et al. (2016), both of whom propose similar copying mechanisms, provides additional evidence for the utility of copying on a wide range of NLP tasks.

5.4 Main Results

For our main results, we train our model with a variety of data recombination strategies on all three datasets. These results are summarized in Tables 2 and 3. We compare our system to the baseline of not using any data recombination, as well as to state-of-the-art systems on all three datasets.

We find that data recombination consistently improves accuracy across the three domains we evaluated on, and that the strongest results come from composing multiple strategies. Combining ABSWHOLEPHRASES, ABSENTITIES, and CONCAT-2 yields a 4.3 percentage point improvement over the baseline without data recombination on GEO, and an average of 1.7 percentage points on OVERNIGHT. In fact, on GEO, we achieve test accuracy of 89.3%, which surpasses the previous state-of-the-art, excluding Liang et al. (2011), which used a seed lexicon for predicates. On ATIS, we experiment with concatenating more than 2 examples, to make up for the fact that we cannot apply ABSWHOLEPHRASES, which generates longer examples. We obtain a test accuracy of 83.3 with ABSENTITIES composed with CONCAT-3, which beats the baseline by 7 percentage points and is competitive with the state-of-the-art.

Data recombination without copying. For completeness, we also investigated the effects of data recombination on the model without attention-based copying. We found that recombination helped significantly on GEO and ATIS, but hurt the model slightly on OVERNIGHT. On GEO, the best data recombination strategy yielded test accuracy of 82.9%, for a gain of 8.3 percentage points over the baseline with no copying and no recombination; on ATIS, data recombination gives test accuracies as high as 74.6%, a 4.7 point gain over the same baseline. However, no data recombination strategy improved average test accuracy on OVERNIGHT; the best one resulted in a 0.3 percentage point decrease in test accuracy. We hypothesize that data recombination helps less on OVERNIGHT in general because the space of possible logical forms is very limited, making it more like a large multiclass classification task. Therefore, it is less important for the model to learn good compositional representations that generalize to new logical forms at test time.

[2]The method of Liang et al. (2011) is not comparable to ours, as they as they used a seed lexicon mapping words to predicates. We explicitly avoid using such prior knowledge in our system.

	BASKETBALL	BLOCKS	CALENDAR	HOUSING	PUBLICATIONS	RECIPES	RESTAURANTS	SOCIAL	Avg.
Previous Work									
Wang et al. (2015)	46.3	41.9	74.4	54.0	59.0	70.8	75.9	48.2	58.8
Our Model									
No Recombination	85.2	58.1	78.0	71.4	76.4	79.6	76.2	81.4	75.8
ABSENTITIES	86.7	60.2	78.0	65.6	73.9	77.3	79.5	81.3	75.3
ABSWHOLEPHRASES	86.7	55.9	79.2	69.8	76.4	77.8	80.7	80.9	75.9
CONCAT-2	84.7	**60.7**	75.6	69.8	74.5	80.1	79.5	80.8	75.7
AWP + AE	85.2	54.1	78.6	67.2	73.9	79.6	**81.9**	**82.1**	75.3
AWP + AE + C2	**87.5**	60.2	**81.0**	**72.5**	**78.3**	**81.0**	79.5	79.6	**77.5**

Table 3: Test accuracy using different data recombination strategies on the OVERNIGHT tasks.

Depth-2 (same length)
x: "*rel:12 of rel:17 of ent:14*"
y: `( _rel:12 ( _rel:17 _ent:14 ) )`
Depth-4 (longer)
x: "*rel:23 of rel:36 of rel:38 of rel:10 of ent:05*"
y: `( _rel:23 ( _rel:36 ( _rel:38 ( _rel:10 _ent:05 ) ) ) )`

Figure 5: A sample of our artificial data.

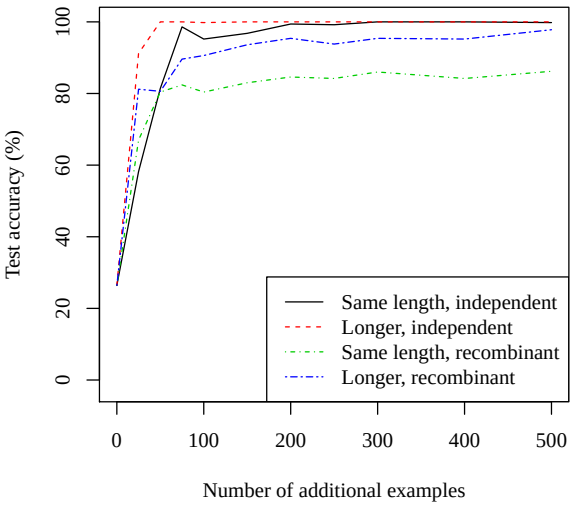


Figure 6: The results of our artificial data experiments. We see that the model learns more from longer examples than from same-length examples.

5.5 Effect of Longer Examples

Interestingly, strategies like ABSWHOLEPHRASES and CONCAT-2 help the model even though the resulting recombinant examples are generally not in the support of the test distribution. In particular, these recombinant examples are on average longer than those in the actual dataset, which makes them harder for the attention-based model. Indeed, for every domain, our best accuracy numbers involved some form of concatenation, and often involved ABSWHOLEPHRASES as well. In comparison, applying ABSENTITIES alone, which generates examples of the same length as those in the original dataset, was generally less effective.

We conducted additional experiments on artificial data to investigate the importance of adding longer, harder examples. We experimented with adding new examples via data recombination, as well as adding new independent examples (e.g. to simulate the acquisition of more training data). We constructed a simple world containing a set of entities and a set of binary relations. For any n, we can generate a set of depth-n examples, which involve the composition of n relations applied to a single entity. Example data points are shown in Figure 5. We train our model on various datasets, then test it on a set of 500 randomly chosen depth-2 examples. The model always has access to a small seed training set of 100 depth-2 examples. We then add one of four types of examples to the training set:

- **Same length, independent**: New randomly chosen depth-2 examples.[3]
- **Longer, independent**: Randomly chosen depth-4 examples.
- **Same length, recombinant**: Depth-2 examples sampled from the grammar induced by applying ABSENTITIES to the seed dataset.
- **Longer, recombinant**: Depth-4 examples sampled from the grammar induced by applying ABSWHOLEPHRASES followed by ABSENTITIES to the seed dataset.

To maintain consistency between the independent and recombinant experiments, we fix the recombinant examples across all epochs, instead of resampling at every epoch. In Figure 6, we plot accuracy on the test set versus the number of additional examples added of each of these four types. As

[3]Technically, these are not completely independent, as we sample these new examples without replacement. The same applies to the longer "independent" examples.

expected, independent examples are more helpful than the recombinant ones, but both help the model improve considerably. In addition, we see that even though the test dataset only has short examples, adding longer examples helps the model more than adding shorter ones, in both the independent and recombinant cases. These results underscore the importance training on longer, harder examples.

6 Discussion

In this paper, we have presented a novel framework we term data recombination, in which we generate new training examples from a high-precision generative model induced from the original training dataset. We have demonstrated its effectiveness in improving the accuracy of a sequence-to-sequence RNN model on three semantic parsing datasets, using a synchronous context-free grammar as our generative model.

There has been growing interest in applying neural networks to semantic parsing and related tasks. Dong and Lapata (2016) concurrently developed an attention-based RNN model for semantic parsing, although they did not use data recombination. Grefenstette et al. (2014) proposed a non-recurrent neural model for semantic parsing, though they did not run experiments. Mei et al. (2016) use an RNN model to perform a related task of instruction following.

Our proposed attention-based copying mechanism bears a strong resemblance to two models that were developed independently by other groups. Gu et al. (2016) apply a very similar copying mechanism to text summarization and single-turn dialogue generation. Gulcehre et al. (2016) propose a model that decides at each step whether to write from a "shortlist" vocabulary or copy from the input, and report improvements on machine translation and text summarization. Another piece of related work is Luong et al. (2015b), who train a neural machine translation system to copy rare words, relying on an external system to generate alignments.

Prior work has explored using paraphrasing for data augmentation on NLP tasks. Zhang et al. (2015) augment their data by swapping out words for synonyms from WordNet. Wang and Yang (2015) use a similar strategy, but identify similar words and phrases based on cosine distance between vector space embeddings. Unlike our data recombination strategies, these techniques only change inputs x, while keeping the labels y fixed. Additionally, these paraphrasing-based transformations can be described in terms of grammar induction, so they can be incorporated into our framework.

In data recombination, data generated by a high-precision generative model is used to train a second, domain-general model. Generative oversampling (Liu et al., 2007) learns a generative model in a multiclass classification setting, then uses it to generate additional examples from rare classes in order to combat label imbalance. Uptraining (Petrov et al., 2010) uses data labeled by an accurate but slow model to train a computationally cheaper second model. Vinyals et al. (2015b) generate a large dataset of constituency parse trees by taking sentences that multiple existing systems parse in the same way, and train a neural model on this dataset.

Some of our induced grammars generate examples that are not in the test distribution, but nonetheless aid in generalization. Related work has also explored the idea of training on altered or out-of-domain data, often interpreting it as a form of regularization. Dropout training has been shown to be a form of adaptive regularization (Hinton et al., 2012; Wager et al., 2013). Guu et al. (2015) showed that encouraging a knowledge base completion model to handle longer path queries acts as a form of structural regularization.

Language is a blend of crisp regularities and soft relationships. Our work takes RNNs, which excel at modeling soft phenomena, and uses a highly structured tool—synchronous context free grammars—to infuse them with an understanding of crisp structure. We believe this paradigm for simultaneously modeling the soft and hard aspects of language should have broader applicability beyond semantic parsing.

Acknowledgments This work was supported by the NSF Graduate Research Fellowship under Grant No. DGE-114747, and the DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF-15-1-0462.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x50757a37779b485f89012e4ba03b6f4f/`.

References

Y. Artzi and L. Zettlemoyer. 2013a. UW SPF: The University of Washington semantic parsing framework. *arXiv preprint arXiv:1311.3011*.

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

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Python for Scientific Computing Conference*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

L. Dong and M. Lapata. 2016. Language to logical form with neural attention. In *Association for Computational Linguistics (ACL)*.

C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Association for Computational Linguistics (ACL)*.

E. Grefenstette, P. Blunsom, N. de Freitas, and K. M. Hermann. 2014. A deep architecture for semantic parsing. In *ACL Workshop on Semantic Parsing*, pages 22–27.

J. Gu, Z. Lu, H. Li, and V. O. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics (ACL)*.

C. Gulcehre, S. Ahn, R. Nallapati, B. Zhou, and Y. Bengio. 2016. Pointing the unknown words. In *Association for Computational Linguistics (ACL)*.

K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In *Empirical Methods in Natural Language Processing (EMNLP)*.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

S. Hochreiter and J. Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

N. Jaitly and G. E. Hinton. 2013. Vocal tract length perturbation (vtlp) improves speech recognition. In *International Conference on Machine Learning (ICML)*.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105.

N. Kushman and R. Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 826–836.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

A. Liu, J. Ghosh, and C. Martin. 2007. Generative oversampling for mining imbalanced datasets. In *International Conference on Data Mining (DMIN)*.

M. Luong, H. Pham, and C. D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

M. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Association for Computational Linguistics (ACL)*, pages 11–19.

H. Mei, M. Bansal, and M. R. Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

S. Petrov, P. Chang, M. Ringgaard, and H. Alshawi. 2010. Uptraining for accurate deterministic question parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*.

H. Poon. 2013. Grounded unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112.

O. Vinyals, M. Fortunato, and N. Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2674–2682.

O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2755–2763.

S. Wager, S. I. Wang, and P. Liang. 2013. Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems (NIPS)*.

W. Y. Wang and D. Yang. 2015. That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *North American Association for Computational Linguistics (NAACL)*, pages 439–446.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

X. Zhang, J. Zhao, and Y. LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS)*.

K. Zhao and L. Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *North American Association for Computational Linguistics (NAACL)*.

Inferring Logical Forms From Denotations

Panupong Pasupat
Computer Science Department
Stanford University
ppasupat@cs.stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

Abstract

A core problem in learning semantic parsers from denotations is picking out consistent logical forms—those that yield the correct denotation—from a combinatorially large space. To control the search space, previous work relied on restricted set of rules, which limits expressivity. In this paper, we consider a much more expressive class of logical forms, and show how to use dynamic programming to efficiently represent the complete set of consistent logical forms. Expressivity also introduces many more spurious logical forms which are consistent with the correct denotation but do not represent the meaning of the utterance. To address this, we generate fictitious worlds and use crowdsourced denotations on these worlds to filter out spurious logical forms. On the WIKITABLEQUESTIONS dataset, we increase the coverage of answerable questions from 53.5% to 76%, and the additional crowdsourced supervision lets us rule out 92.1% of spurious logical forms.

1 Introduction

Consider the task of learning to answer complex natural language questions (e.g., "*Where did the last 1st place finish occur?*") using only question-answer pairs as supervision (Clarke et al., 2010; Liang et al., 2011; Berant et al., 2013; Artzi and Zettlemoyer, 2013). Semantic parsers map the question into a logical form (e.g., $\mathbf{R}[\texttt{Venue}].\texttt{argmax}(\texttt{Position.1st}, \texttt{Index})$) that can be executed on a knowledge source to obtain the answer (denotation). Logical forms are very expressive since they can be recursively composed, but this very expressivity makes it more difficult to search over the space of logical forms. Previous work sidesteps this obstacle by restricting the set of possible logical form compositions, but this is limiting. For instance, for the system in Pasupat and Liang (2015), in only 53.5% of the examples was the correct logical form even in the set of generated logical forms.

The goal of this paper is to solve two main challenges that prevent us from generating more expressive logical forms. The first challenge is computational: the number of logical forms grows exponentially as their size increases. Directly enumerating over all logical forms becomes infeasible, and pruning techniques such as beam search can inadvertently prune out correct logical forms.

The second challenge is the large increase in *spurious* logical forms—those that do not reflect the semantics of the question but coincidentally execute to the correct denotation. For example, while logical forms $z_1, \ldots, z_5$ in Figure 1 are all *consistent* (they execute to the correct answer y), the logical forms z_4 and z_5 are spurious and would give incorrect answers if the table were to change.

We address these two challenges by solving two interconnected tasks. The first task, which addresses the computational challenge, is to enumerate the set Z of all consistent logical forms given a question x, a knowledge source w ("world"), and the target denotation y (Section 4). Observing that the space of possible denotations grows much more slowly than the space of logical forms, we perform *dynamic programming on denotations* (DPD) to make search feasible. Our method is guaranteed to find all consistent logical forms up to some bounded size.

Given the set Z of consistent logical forms, the second task is to filter out spurious logical forms from Z (Section 5). Using the property that spurious logical forms ultimately give a wrong answer when the data in the world w changes, we create

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 23–32,
Berlin, Germany, August 7-12, 2016.

Year	Venue	Position	Event	Time
2001	Hungary	2nd	400m	47.12
2003	Finland	1st	400m	46.69
2005	Germany	11th	400m	46.62
2007	Thailand	1st	relay	182.05
2008	China	7th	relay	180.32

x: "*Where did the last 1st place finish occur?*"
y: Thailand

Consistent

Correct

z_1: $\mathbf{R}$[Venue].argmax(Position.1st, Index)
Among rows with Position = 1st, pick the one with maximum index, then return the Venue of that row.

z_2: $\mathbf{R}$[Venue].Index.max($\mathbf{R}$[Index].Position.1st)
Find the maximum index of rows with Position = 1st, then return the Venue of the row with that index.

z_3: $\mathbf{R}$[Venue].argmax(Position.Number.*1*, $\mathbf{R}[\lambda x.\mathbf{R}$[Date].$\mathbf{R}$[Year].$x]$)
Among rows with Position number 1, pick one with latest date in the Year column and return the Venue.

Spurious

z_4: $\mathbf{R}$[Venue].argmax(Position.Number.*1*, $\mathbf{R}[\lambda x.\mathbf{R}$[Number].$\mathbf{R}$[Time].$x]$)
Among rows with Position number 1, pick the one with maximum Time number. Return the Venue.

z_5: $\mathbf{R}$[Venue].Year.Number.($\mathbf{R}$[Number].$\mathbf{R}$[Year].argmax(Type.Row, Index) $-$ *1*)
Subtract 1 from the Year in the last row, then return the Venue of the row with that Year.

Inconsistent

$\tilde{z}$: $\mathbf{R}$[Venue].argmin(Position.1st, Index)
Among rows with Position = 1st, pick the one with minimum index, then return the Venue. (= Finland)

Figure 1: Six logical forms generated from the question x. The first five are *consistent*: they execute to the correct answer y. Of those, *correct* logical forms z_1, z_2, and z_3 are different ways to represent the semantics of x, while *spurious* logical forms z_4 and z_5 get the right answer y for the wrong reasons.

fictitious worlds to test the denotations of the logical forms in Z. We use crowdsourcing to annotate the correct denotations on a subset of the generated worlds. To reduce the amount of annotation needed, we choose the subset that maximizes the expected information gain. The pruned set of logical forms would provide a stronger supervision signal for training a semantic parser.

We test our methods on the WIKITABLEQUESTIONS dataset of complex questions on Wikipedia tables. We define a simple, general set of deduction rules (Section 3), and use DPD to confirm that the rules generate a correct logical form in

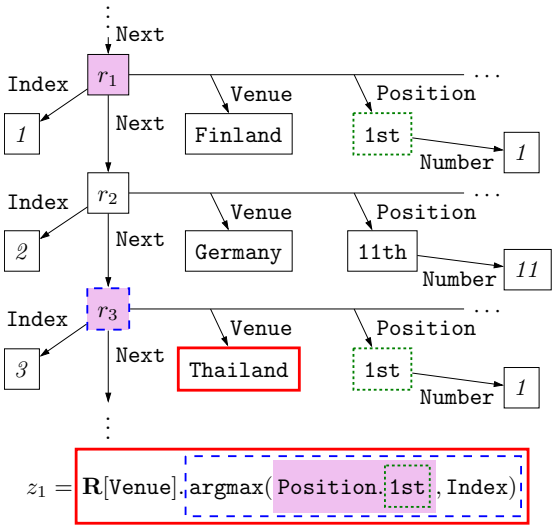


Figure 2: The table in Figure 1 is converted into a graph. The recursive execution of logical form z_1 is shown via the different colors and styles.

76% of the examples, up from the 53.5% in Pasupat and Liang (2015). Moreover, unlike beam search, DPD is guaranteed to find all consistent logical forms up to a bounded size. Finally, by using annotated data on fictitious worlds, we are able to prune out 92.1% of the spurious logical forms.

2 Setup

The overarching motivation of this work is allowing people to ask questions involving computation on semi-structured knowledge sources such as tables from the Web. This section introduces how the knowledge source is represented, how the computation is carried out using *logical forms*, and our task of inferring correct logical forms.

Worlds. We use the term *world* to refer to a collection of entities and relations between entities. One way to represent a world w is as a directed graph with nodes for entities and directed edges for relations. (For example, a world about geography would contain a node `Europe` with an edge `Contains` to another node `Germany`.)

In this paper, we use data tables from the Web as knowledge sources, such as the one in Figure 1. We follow the construction in Pasupat and Liang (2015) for converting a table into a directed graph (see Figure 2). Rows and cells become nodes (e.g., r_0 = first row and `Finland`) while columns become labeled directed edges between them (e.g., `Venue` maps r_1 to `Finland`). The graph is augmented with additional edges `Next` (from each

row to the next) and Index (from each row to its index number). In addition, we add normalization edges to cell nodes, including Number (from the cell to the first number in the cell), Num2 (the second number), Date (interpretation as a date), and Part (each list item if the cell represents a list). For example, a cell with content "*3-4*" has a Number edge to the integer *3*, a Num2 edge to *4*, and a Date edge to *XX-03-04*.

Logical forms. We can perform computation on a world w using a *logical form* z, a small program that can be executed on the world, resulting in a *denotation* $[\![z]\!]_w$.

We use lambda DCS (Liang, 2013) as the language of logical forms. As a demonstration, we will use z_1 in Figure 2 as an example. The smallest units of lambda DCS are entities (e.g., 1st) and relations (e.g., Position). Larger logical forms can be constructed using logical operations, and the denotation of the new logical form can be computed from denotations of its constituents. For example, applying the *join* operation on Position and 1st gives Position.1st, whose denotation is the set of entities with relation Position pointing to 1st. With the world in Figure 2, the denotation is $[\![\texttt{Position.1st}]\!]_w = \{r_1, r_3\}$, which corresponds to the 2nd and 4th rows in the table. The partial logical form Position.1st is then used to construct argmax(Position.1st, Index), the denotation of which can be computed by mapping the entities in $[\![\texttt{Position.1st}]\!]_w = \{r_1, r_3\}$ using the relation Index ($\{r_0 : 0, r_1 : 1, \dots\}$), and then picking the one with the largest mapped value (r_3, which is mapped to *3*). The resulting logical form is finally combined with $\mathbf{R}[\texttt{Venue}]$ with another *join* operation. The relation $\mathbf{R}[\texttt{Venue}]$ is the *reverse* of Venue, which corresponds to traversing Venue edges in the reverse direction.

Semantic parsing. A semantic parser maps a natural language utterance x (e.g., "*Where did the last 1st place finish occur?*") into a logical form z. With denotations as supervision, a semantic parser is trained to put high probability on z's that are *consistent*—logical forms that execute to the correct denotation y (e.g., Thailand). When the space of logical forms is large, searching for consistent logical forms z can become a challenge.

As illustrated in Figure 1, consistent logical forms can be divided into two groups: *correct* logical forms represent valid ways for computing the answer, while *spurious* logical forms accidentally get the right answer for the wrong reasons (e.g., z_4 picks the row with the maximum time but gets the correct answer anyway).

Tasks. Denote by Z and Z_c the sets of all consistent and correct logical forms, respectively. The first task is to efficiently compute Z given an utterance x, a world w, and the correct denotation y (Section 4). With the set Z, the second task is to infer Z_c by pruning spurious logical forms from Z (Section 5).

3 Deduction rules

The space of logical forms given an utterance x and a world w is defined recursively by a set of *deduction rules* (Table 1). In this setting, each constructed logical form belongs to a *category* (*Set*, *Rel*, or *Map*). These categories are used for type checking in a similar fashion to categories in syntactic parsing. Each deduction rule specifies the categories of the arguments, category of the resulting logical form, and how the logical form is constructed from the arguments.

Deduction rules are divided into base rules and compositional rules. A base rule follows one of the following templates:

$$\textit{TokenSpan}[span] \rightarrow c\,[f(span)] \quad (1)$$

$$\emptyset \rightarrow c\,[f()] \quad (2)$$

A rule of Template 1 is triggered by a span of tokens from x (e.g., to construct z_1 in Figure 2 from x in Figure 1, Rule B1 from Table 1 constructs 1st of category *Set* from the phrase "*1st*"). Meanwhile, a rule of Template 2 generates a logical form without any trigger (e.g., Rule B5 generates Position of category *Rel* from the graph edge Position without a specific trigger in x).

Compositional rules then construct larger logical forms from smaller ones:

$$c_1\,[z_1] + c_2\,[z_2] \rightarrow c\,[g(z_1, z_2)] \quad (3)$$

$$c_1\,[z_1] \rightarrow c\,[g(z_1)] \quad (4)$$

A rule of Template 3 combines partial logical forms z_1 and z_2 of categories c_1 and c_2 into $g(z_1, z_2)$ of category c (e.g., Rule C1 uses 1st of category *Set* and Position of category *Rel* to construct Position.1st of category *Set*). Template 4 works similarly.

Most rules construct logical forms without requiring a trigger from the utterance x. This is

	Rule	Semantics
	Base Rules	
B1	*TokenSpan* → *Set*	fuzzymatch($span$)
	(entity fuzzily matching the text: "*chinese*" → `China`)	
B2	*TokenSpan* → *Set*	val($span$)
	(interpreted value: "*march 2015*" → *2015-03-XX*)	
B3	$\emptyset$ → *Set*	`Type.Row`
	(the set of all rows)	
B4	$\emptyset$ → *Set*	$c \in$ ClosedClass
	(any entity from a column with few unique entities)	
	(e.g., `400m` or `relay` from the Event column)	
B5	$\emptyset$ → *Rel*	$r \in$ GraphEdges
	(any relation in the graph: `Venue`, `Next`, `Num2`, ...)	
B6	$\emptyset$ → *Rel*	`!=` | `<` | `<=` | `>` | `>=`
	Compositional Rules	
C1	*Set* + *Rel* → *Set*	$z_2.z_1 \mid \mathbf{R}[z_2].z_1$
	($\mathbf{R}[z]$ is the reverse of z; i.e., flip the arrow direction)	
C2	*Set* → *Set*	$a(z_1)$
	($a \in \{\texttt{count}, \texttt{max}, \texttt{min}, \texttt{sum}, \texttt{avg}\}$)	
C3	*Set* + *Set* → *Set*	$z_1 \sqcap z_2 \mid z_1 \sqcup z_2 \mid z_1 - z_2$
	(subtraction is only allowed on numbers)	
	Compositional Rules with Maps	
	Initialization	
M1	*Set* → *Map*	(z_1, x) (identity map)
	Operations on Map	
M2	*Map* + *Rel* → *Map*	$(u_1, z_2.b_1) \mid (u_1, \mathbf{R}[z_2].b_1)$
M3	*Map* → *Map*	$(u_1, a(b_1))$
	($a \in \{\texttt{count}, \texttt{max}, \texttt{min}, \texttt{sum}, \texttt{avg}\}$)	
M4	*Map* + *Set* → *Map*	$(u_1, b_1 \sqcap z_2) \mid \ldots$
M5	*Map* + *Map* → *Map*	$(u_1, b_1 \sqcap b_2) \mid \ldots$
	(Allowed only when $u_1 = u_2$)	
	(Rules M4 and M5 are repeated for $\sqcup$ and $-$)	
	Finalization	
M6	*Map* → *Set*	$\texttt{argmin}(u_1, \mathbf{R}[\lambda x.b_1])$ | $\texttt{argmax}(u_1, \mathbf{R}[\lambda x.b_1])$

Table 1: Deduction rules define the space of logical forms by specifying how partial logical forms are constructed. The logical form of the i-th argument is denoted by z_i (or (u_i, b_i) if the argument is a *Map*). The set of final logical forms contains any logical form with category *Set*.

crucial for generating implicit relations (e.g., generating `Year` from "*what's the venue in 2000?*" without a trigger "*year*"), and generating operations without a lexicon (e.g., generating `argmax` from "*where's the longest competition*"). However, the downside is that the space of possible logical forms becomes very large.

The *Map* category. The technique in this paper requires execution of partial logical forms. This poses a challenge for `argmin` and `argmax` operations, which take a set and a binary relation as arguments. The binary could be a complex function (e.g., in z_3 from Figure 1). While it is possible to build the binary independently from the set, executing a complex binary is sometimes impossible (e.g., the denotation of $\lambda x.\texttt{count}(x)$ is impossible to write explicitly without knowledge of x).

We address this challenge with the *Map* category. A *Map* is a pair (u, b) of a finite set u (unary) and a binary relation b. The denotation of (u, b) is $([\![u]\!]_w, [\![b]\!]'_w)$ where the binary $[\![b]\!]'_w$ is $[\![b]\!]_w$ with the domain restricted to the set $[\![u]\!]_w$. For example, consider the construction of $\texttt{argmax}(\texttt{Position.1st}, \texttt{Index})$. After constructing `Position.1st` with denotation $\{r_1, r_3\}$, Rule M1 initializes $(\texttt{Position.1st}, x)$ with denotation $(\{r_1, r_3\}, \{r_1 : \{r_1\}, r_3 : \{r_3\}\})$. Rule M2 is then applied to generate $(\texttt{Position.1st}, \mathbf{R}[\texttt{Index}].x)$ with denotation $(\{r_1, r_3\}, \{r_1 : \{1\}, r_3 : \{3\}\})$. Finally, Rule M6 converts the *Map* into the desired `argmax` logical form with denotation $\{r_3\}$.

Generality of deduction rules. Using domain knowledge, previous work restricted the space of logical forms by manually defining the categories c or the semantic functions f and g to fit the domain. For example, the category *Set* might be divided into *Records*, *Values*, and *Atomic* when the knowledge source is a table (Pasupat and Liang, 2015). Another example is when a compositional rule g (e.g., $\texttt{sum}(z_1)$) must be triggered by some phrase in a lexicon (e.g., words like "*total*" that align to `sum` in the training data). Such restrictions make search more tractable but greatly limit the scope of questions that can be answered.

Here, we have increased the coverage of logical forms by making the deduction rules simple and general, essentially following the syntax of lambda DCS. The base rules only generates entities that approximately match the utterance, but all possible relations, and all possible further combinations.

Beam search. Given the deduction rules, an utterance x and a world w, we would like to generate all derived logical forms Z. We first present the floating parser (Pasupat and Liang, 2015), which uses beam search to generate $Z_\text{b} \subseteq Z$, a usually incomplete subset. Intuitively, the algorithm first constructs base logical forms based on spans of the utterance, and then builds larger logical forms of increasing size in a "floating" fashion—without requiring a trigger from the utterance.

Formally, partial logical forms with category c and size s are stored in a *cell* (c, s). The algorithm first generates base logical forms from base deduction rules and store them in cells $(c, 0)$ (e.g., the cell $(\textit{Set}, 0)$ contains `1st`, `Type.Row`, and so on). Then for each size $s = 1, \ldots, s_\text{max}$, we populate

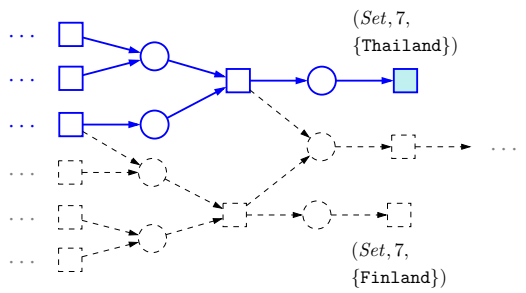


Figure 3: The first pass of DPD constructs cells (c, s, d) (square nodes) using denotationally invariant semantic functions (circle nodes). The second pass enumerates all logical forms along paths that lead to the correct denotation y (solid lines).

the cells (c, s) by applying compositional rules on partial logical forms with size less than s. For instance, when $s = 2$, we can apply Rule C1 on logical forms `Number`.*1* from cell $(\textit{Set}, s_1 = 1)$ and `Position` from cell $(\textit{Rel}, s_2 = 0)$ to create `Position.Number`.*1* in cell $(\textit{Set}, s_0 + s_1 + 1 = 2)$. After populating each cell (c, s), the list of logical forms in the cell is pruned based on the model scores to a fixed beam size in order to control the search space. Finally, the set Z_{b} is formed by collecting logical forms from all cells $(\textit{Set}, s)$ for $s = 1, \ldots, s_{\max}$.

Due to the generality of our deduction rules, the number of logical forms grows quickly as the size s increases. As such, partial logical forms that are essential for building the desired logical forms might fall off the beam early on. In the next section, we present a new search method that compresses the search space using denotations.

4 Dynamic programming on denotations

Our first step toward finding all correct logical forms is to represent all consistent logical forms (those that execute to the correct denotation). Formally, given x, w, and y, we wish to generate the set Z of all logical forms z such that $[\![z]\!]_w = y$.

As mentioned in the previous section, beam search does not recover the full set Z due to pruning. Our key observation is that while the number of logical forms explodes, the number of *distinct denotations* of those logical forms is much more controlled, as multiple logical forms can share the same denotation. So instead of directly enumerating logical forms, we use *dynamic programming on denotations* (DPD), which is inspired by similar methods from program induction (Lau et al., 2003; Liang et al., 2010; Gulwani, 2011).

The main idea of DPD is to collapse logical forms with the same denotation together. Instead of using cells (c, s) as in beam search, we perform dynamic programming using cells (c, s, d) where d is a denotation. For instance, the logical form `Position.Number`.*1* will now be stored in cell $(\textit{Set}, 2, \{r_1, r_3\})$.

For DPD to work, each deduction rule must have a *denotationally invariant* semantic function g, meaning that the denotation of the resulting logical form $g(z_1, z_2)$ only depends on the denotations of z_1 and z_2:

$$[\![z_1]\!]_w = [\![z_1']\!]_w \ \wedge \ [\![z_2]\!]_w = [\![z_2']\!]_w \\ \Rightarrow [\![g(z_1, z_2)]\!]_w = [\![g(z_1', z_2')]\!]_w$$

All of our deduction rules in Table 1 are denotationally invariant, but a rule that, for instance, returns the argument with the larger logical form size would not be. Applying a denotationally invariant deduction rule on any pair of logical forms from (c_1, s_1, d_1) and (c_2, s_2, d_2) always results in a logical form with the same denotation d in the same cell $(c, s_1 + s_2 + 1, d)$.[1] (For example, the cell $(\textit{Set}, 4, \{r_3\})$ contains $z_1 :=$ `argmax(Position.1st, Index)` and $z_1' :=$ `argmin(Event.Relay, Index)`. Combining each of these with `Venue` using Rule C1 gives $\mathbf{R}[\texttt{Venue}].z_1$ and $\mathbf{R}[\texttt{Venue}].z_1'$, which belong to the same cell $(\textit{Set}, 5, \{\texttt{Thailand}\})$).

Algorithm. DPD proceeds in two forward passes. The first pass finds the possible combinations of cells (c, s, d) that lead to the correct denotation y, while the second pass enumerates the logical forms in the cells found in the first pass. Figure 3 illustrates the DPD algorithm.

In the first pass, we are only concerned about finding relevant cell combinations and not the actual logical forms. Therefore, any logical form that belongs to a cell could be used as an argument of a deduction rule to generate further logical forms. Thus, we keep at most one logical form per cell; subsequent logical forms that are generated for that cell are discarded.

After populating all cells up to size $s_{\max}$, we list all cells $(\textit{Set}, s, y)$ with the correct denotation y, and then note all possible rule combinations $(\text{cell}_1, \text{rule})$ or $(\text{cell}_1, \text{cell}_2, \text{rule})$ that lead to those

[1] Semantic functions f with one argument work similarly.

final cells, including the combinations that yielded discarded logical forms.

The second pass retrieves the actual logical forms that yield the correct denotation. To do this, we simply populate the cells (c, s, d) with all logical forms, using only rule combinations that lead to final cells. This elimination of irrelevant rule combinations effectively reduces the search space. (In Section 6.2, we empirically show that the number of cells considered is reduced by 98.7%.)

The parsing chart is represented as a hypergraph as in Figure 3. After eliminating unused rule combinations, each of the remaining hyperpaths from base predicates to the target denotation corresponds to a single logical form. making the remaining parsing chart a compact implicit representation of all consistent logical forms. This representation is guaranteed to cover all possible logical forms under the size limit $s_{\max}$ that can be constructed by the deduction rules.

In our experiments, we apply DPD on the deduction rules in Table 1 and explicitly enumerate the logical forms produced by the second pass. For efficiency, we prune logical forms that are clearly redundant (e.g., applying `max` on a set of size 1). We also restrict a few rules that might otherwise create too many denotations. For example, we restricted the union operation ($\sqcup$) except unions of two entities (e.g., we allow `Germany` $\sqcup$ `Finland` but not `Venue.Hungary` $\sqcup \ldots$), subtraction when building a *Map*, and `count` on a set of size 1.[2]

5 Fictitious worlds

After finding the set Z of all consistent logical forms, we want to filter out spurious logical forms. To do so, we observe that semantically correct logical forms should also give the correct denotation in worlds w' other than than w. In contrast, spurious logical forms will fail to produce the correct denotation on some other world.

Generating fictitious worlds. With the observation above, we generate *fictitious worlds* $w_1, w_2, \ldots$, where each world w_i is a slight alteration of w. As we will be executing logical forms $z \in Z$ on w_i, we should ensure that all entities and relations in $z \in Z$ appear in the fictitious world w_i (e.g., z_1 in Figure 1 would be meaningless if the entity `1st` does not appear in w_i). To this end, we impose that all predicates present in the original world w should also be present in w_i as well.

[2]While we technically can apply `count` on sets of size 1, the number of spurious logical forms explodes as there are too many sets of size 1 generated.

Year	Venue	Position	Event	Time
2001	Finland	7th	relay	46.62
2003	Germany	1st	400m	180.32
2005	China	1st	relay	47.12
2007	Hungary	7th	relay	182.05

Figure 4: From the example in Figure 1, we generate a table for the fictitious world w_1.

	w	w_1	w_2	$\cdots$	
z_1	Thailand	China	Finland	$\cdots$	q_1
z_2	Thailand	China	Finland	$\cdots$	
z_3	Thailand	China	Finland	$\cdots$	
z_4	Thailand	Germany	China	$\cdots$	q_2
z_5	Thailand	China	China	$\cdots$	q_3
z_6	Thailand	China	China	$\cdots$	
$\vdots$	$\vdots$	$\vdots$	$\vdots$		

Figure 5: We execute consistent logical forms $z_i \in Z$ on fictitious worlds to get denotation tuples. Logical forms with the same denotation tuple are grouped into the same equivalence class q_j.

In our case where the world w comes from a data table t, we construct w_i from a new table t_i as follows: we go through each column of t and resample the cells in that column. The cells are sampled using random draws without replacement if the original cells are all distinct, and with replacement otherwise. Sorted columns are kept sorted. To ensure that predicates in w exist in w_i, we use the same set of table columns and enforce that any entity fuzzily matching a span in the question x must be present in t_i (e.g., for the example in Figure 1, the generated t_i must contain "1st"). Figure 4 shows an example fictitious table generated from the table in Figure 1.

Fictitious worlds are similar to test suites for computer programs. However, unlike manually designed test suites, we do not yet know the correct answer for each fictitious world or whether a world is helpful for filtering out spurious logical forms. The next subsections introduce our method for choosing a subset of useful fictitious worlds to be annotated.

Equivalence classes. Let $W = (w_1, \ldots, w_k)$ be the list of all possible fictitious worlds. For each $z \in Z$, we define the denotation tuple $[\![z]\!]_W = ([\![z]\!]_{w_1}, \ldots, [\![z]\!]_{w_k})$. We observe that some logical forms produce the same denotation across all

fictitious worlds. This may be due to an algebraic equivalence in logical forms (e.g., z_1 and z_2 in Figure 1) or due to the constraints in the construction of fictitious worlds (e.g., z_1 and z_3 in Figure 1 are equivalent as long as the Year column is sorted). We group logical forms into equivalence classes based on their denotation tuples, as illustrated in Figure 5. When the question is unambiguous, we expect at most one equivalence class to contain correct logical forms.

Annotation. To pin down the correct equivalence class, we acquire the correct answers to the question x on some subset $W' = (w'_1, \ldots, w'_\ell) \subseteq W$ of ℓ fictitious worlds, as it is impractical to obtain annotations on all fictitious worlds in W. We compile equivalence classes that agree with the annotations into a set Z_c of correct logical forms.

We want to choose W' that gives us the most information about the correct equivalence class as possible. This is analogous to standard practices in active learning (Settles, 2010).[3] Let $\mathcal{Q}$ be the set of all equivalence classes q, and let $[\![q]\!]_{W'}$ be the denotation tuple computed by executing an arbitrary $z \in q$ on W'. The subset W' divides $\mathcal{Q}$ into partitions $F_t = \{q \in \mathcal{Q} : [\![q]\!]_{W'} = t\}$ based on the denotation tuples t (e.g., from Figure 5, if W' contains just w_2, then q_2 and q_3 will be in the same partition $F_{(\text{China})}$). The annotation t^*, which is also a denotation tuple, will mark one of these partitions F_{t^*} as correct. Thus, to prune out many spurious equivalence classes, the partitions should be as numerous and as small as possible.

More formally, we choose a subset W' that maximizes the expected information gain (or equivalently, the reduction in entropy) about the correct equivalence class given the annotation. With random variables $Q \in \mathcal{Q}$ representing the correct equivalence class and $T^*_{W'}$ for the annotation on worlds W', we seek to find $\arg\min_{W'} \mathrm{H}(Q \mid T^*_{W'})$. Assuming a uniform prior on Q ($p(q) = 1/|\mathcal{Q}|$) and accurate annotation ($p(t^* \mid q) = \mathbf{I}[q \in F_{t^*}]$):

$$\begin{aligned}\mathrm{H}(Q \mid T^*_{W'}) &= \sum_{q,t} p(q,t) \log \frac{p(t)}{p(q,t)} \\ &= \frac{1}{|\mathcal{Q}|} \sum_t |F_t| \log |F_t|. \qquad (*)\end{aligned}$$

[3]The difference is that we are obtaining partial information about an individual example rather than partial information about the parameters.

We exhaustively search for W' that minimizes (*). The objective value follows our intuition since $\sum_t |F_t| \log |F_t|$ is small when the terms $|F_t|$ are small and numerous.

In our experiments, we approximate the full set W of fictitious worlds by generating $k = 30$ worlds to compute equivalence classes. We choose a subset of $\ell = 5$ worlds to be annotated.

6 Experiments

For the experiments, we use the training portion of the WIKITABLEQUESTIONS dataset (Pasupat and Liang, 2015), which consists of 14,152 questions on 1,679 Wikipedia tables gathered by crowd workers. Answering these complex questions requires different types of operations. The same operation can be phrased in different ways (e.g., "*best*", "*top ranking*", or "*lowest ranking number*") and the interpretation of some phrases depend on the context (e.g., "*number of*" could be a table lookup or a count operation). The lexical content of the questions is also quite diverse: even excluding numbers and symbols, the 14,152 training examples contain 9,671 unique words, only 10% of which appear more than 10 times.

We attempted to manually annotate the first 300 examples with lambda DCS logical forms. We successfully constructed correct logical forms for 84% of these examples, which is a good number considering the questions were created by humans who could use the table however they wanted. The remaining 16% reflect limitations in our setup—for example, non-canonical table layouts, answers appearing in running text or images, and common sense reasoning (e.g., knowing that "Quarterfinal" is better than "Round of 16").

6.1 Generality of deduction rules

We compare our set of deduction rules with the one given in Pasupat and Liang (2015) (henceforth PL15). PL15 reported generating the annotated logical form in 53.5% of the first 200 examples. With our more general deduction rules, we use DPD to verify that the rules are able to generate the annotated logical form in 76% of the first 300 examples, within the logical form size limit $s_{\max}$ of 7. This is 90.5% of the examples that were successfully annotated. Figure 6 shows some examples of logical forms we cover that PL15 could not. Since DPD is guaranteed to find all consistent logical forms, we can be sure that the logical

"*which opponent has the most wins*" $z = \texttt{argmax}(\mathbf{R}[\texttt{Opponent}].\texttt{Type.Row},$ $\mathbf{R}[\lambda x.\texttt{count}(\texttt{Opponent}.x \sqcap \texttt{Result.Lost}])$
"*how long did ian armstrong serve?*" $z = \mathbf{R}[\texttt{Num2}].\mathbf{R}[\texttt{Term}].\texttt{Member.IanArmstrong}$ $- \mathbf{R}[\texttt{Number}].\mathbf{R}[\texttt{Term}].\texttt{Member.IanArmstrong}$
"*which players came in a place before lukas bauer?*" $z = \mathbf{R}[\texttt{Name}].\texttt{Index.<}.\mathbf{R}[\texttt{Index}].\texttt{Name.LukasBauer}$
"*which players played the same position as ardo kreek?*" $z = \mathbf{R}[\texttt{Player}].\texttt{Position}.\mathbf{R}[\texttt{Position}].\texttt{Player.Ardo}$ $\sqcap \texttt{!=.Ardo}$

Figure 6: Several example logical forms our system can generated that are not covered by the deduction rules from the previous work PL15.

forms not covered are due to limitations of the deduction rules. Indeed, the remaining examples either have logical forms with size larger than 7 or require other operations such as addition, union of arbitrary sets, etc.

6.2 Dynamic programming on denotations

Search space. To demonstrate the savings gained by collapsing logical forms with the same denotation, we track the growth of the number of unique logical forms and denotations as the logical form size increases. The plot in Figure 7 shows that the space of logical forms explodes much more quickly than the space of denotations.

The use of denotations also saves us from considering a significant amount of irrelevant partial logical forms. On average over 14,152 training examples, DPD generates approximately 25,000 consistent logical forms. The first pass of DPD generates $\approx$ 153,000 cells (c, s, d), while the second pass generates only $\approx$ 2,000 cells resulting from $\approx$ 8,000 rule combinations, resulting in a 98.7% reduction in the number of cells that have to be considered.

Comparison with beam search. We compare DPD to beam search on the ability to generate (but not rank) the annotated logical forms. We consider two settings: when the beam search parameters are uninitialized (i.e., the beams are pruned randomly), and when the parameters are trained using the system from PL15 (i.e., the beams are pruned based on model scores). The plot in Figure 8 shows that DPD generates more annotated logical forms (76%) compared to beam search (53.7%), even when beam search is guided heuristically by learned parameters. Note that DPD is an exact algorithm and does not require a heuristic.

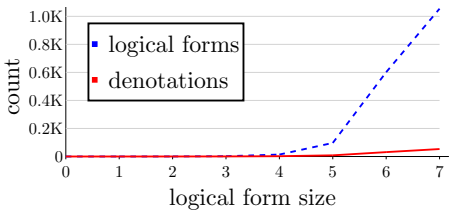


Figure 7: The median of the number of logical forms (dashed) and denotations (solid) as the formula size increases. The space of logical forms grows much faster than the space of denotations.

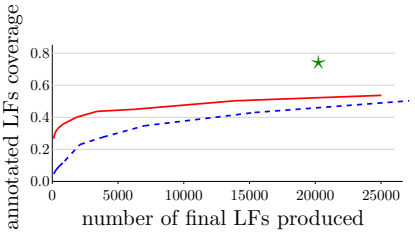


Figure 8: The number of annotated logical forms that can be generated by beam search, both uninitialized (dashed) and initialized (solid), increases with the number of candidates generated (controlled by beam size), but lacks behind DPD (star).

6.3 Fictitious worlds

We now explore how fictitious worlds divide the set of logical forms into equivalence classes, and how the annotated denotations on the chosen worlds help us prune spurious logical forms.

Equivalence classes. Using 30 fictitious worlds per example, we produce an average of 1,237 equivalence classes. One possible concern with using a limited number of fictitious worlds is that we may fail to distinguish some pairs of non-equivalent logical forms. We verify the equivalence classes against the ones computed using 300 fictitious worlds. We found that only 5% of the logical forms are split from the original equivalence classes.

Ideal Annotation. After computing equivalence classes, we choose a subset W' of 5 fictitious worlds to be annotated based on the information-theoretic objective. For each of the 252 examples with an annotated logical form z^*, we use the denotation tuple $t^* = [\![z^*]\!]_{W'}$ as the annotated answers on the chosen fictitious worlds. We are able to rule out 98.7% of the spurious equivalence classes and 98.3% of spurious logical forms. Furthermore, we are able to filter down to just one equivalence class in 32.7% of the examples, and

at most three equivalence classes in 51.3% of the examples. If we choose 5 fictitious worlds randomly instead of maximizing information gain, then the above statistics are 22.6% and 36.5%, respectively. When more than one equivalence classes remain, usually only one class is a dominant class with many equivalent logical forms, while other classes are small and contain logical forms with unusual patterns (e.g., z_5 in Figure 1).

The average size of the correct equivalence class is $\approx$ 3,000 with the standard deviation of $\approx$ 8,000. Because we have an expressive logical language, there are fundamentally many equivalent ways of computing the same quantity.

Crowdsourced Annotation. Data from crowdsourcing is more susceptible to errors. From the 252 annotated examples, we use 177 examples where at least two crowd workers agree on the answer of the original world w. When the crowdsourced data is used to rule out spurious logical forms, the entire set Z of consistent logical forms is pruned out in 11.3% of the examples, and the correct equivalent class is removed in 9% of the examples. These issues are due to annotation errors, inconsistent data (e.g., having date of death before birth date), and different interpretations of the question on the fictitious worlds. For the remaining examples, we are able to prune out 92.1% of spurious logical forms (or 92.6% of spurious equivalence classes).

To prevent the entire Z from being pruned, we can relax our assumption and keep logical forms z that disagree with the annotation in at most 1 fictitious world. The number of times Z is pruned out is reduced to 3%, but the number of spurious logical forms pruned also decreases to 78%.

7 Related Work and Discussion

This work evolved from a long tradition of learning executable semantic parsers, initially from annotated logical forms (Zelle and Mooney, 1996; Kate et al., 2005; Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010), but more recently from denotations (Clarke et al., 2010; Liang et al., 2011; Berant et al., 2013; Kwiatkowski et al., 2013; Pasupat and Liang, 2015). A central challenge in learning from denotations is finding consistent logical forms (those that execute to a given denotation).

As Kwiatkowski et al. (2013) and Berant and Liang (2014) both noted, a chief difficulty with executable semantic parsing is the "schema mismatch"—words in the utterance do not map cleanly onto the predicates in the logical form. This mismatch is especially pronounced in the WIKITABLEQUESTIONS of Pasupat and Liang (2015). In the second example of Figure 6, "*how long*" is realized by a logical form that computes a difference between two dates. The ramification of this mismatch is that finding consistent logical forms cannot solely proceed from the language side. This paper is about using annotated denotations to drive the search over logical forms.

This takes us into the realm of program induction, where the goal is to infer a program (logical form) from input-output pairs (for us, world-denotation pairs). Here, previous work has also leveraged the idea of dynamic programming on denotations (Lau et al., 2003; Liang et al., 2010; Gulwani, 2011), though for more constrained spaces of programs. Continuing the program analogy, generating fictitious worlds is similar in spirit to fuzz testing for generating new test cases (Miller et al., 1990), but the goal there is coverage in a single program rather than identifying the correct (equivalence class of) programs. This connection can potentially improve the flow of ideas between the two fields.

Finally, the effectiveness of dynamic programming on denotations relies on having a manageable set of denotations. For more complex logical forms and larger knowledge graphs, there are many possible angles worth exploring: performing abstract interpretation to collapse denotations into equivalence classes (Cousot and Cousot, 1977), relaxing the notion of getting the correct denotation (Steinhardt and Liang, 2015), or working in a continuous space and relying on gradient descent (Guu et al., 2015; Neelakantan et al., 2016; Yin et al., 2016; Reed and de Freitas, 2016). This paper, by virtue of exact dynamic programming, sets the standard.

Acknowledgments. We gratefully acknowledge the support of the Google Natural Language Understanding Focused Program. In addition, we would like to thank anonymous reviewers for their helpful comments.

Reproducibility. Code and experiments for this paper are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x47cc64d9c8ba4a878807c7c35bb22a42/`.

References

Y. Artzi and L. Zettlemoyer. 2013. UW SPF: The University of Washington semantic parsing framework. *arXiv preprint arXiv:1311.3011*.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Association for Computational Linguistics (ACL)*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

P. Cousot and R. Cousot. 1977. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Principles of Programming Languages (POPL)*, pages 238–252.

S. Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM SIGPLAN Notices*, 46(1):317–330.

K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1062–1068.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Lau, S. Wolfman, P. Domingos, and D. S. Weld. 2003. Programming by demonstration using version space algebra. *Machine Learning*, 53:111–156.

P. Liang, M. I. Jordan, and D. Klein. 2010. Learning programs: A hierarchical Bayesian approach. In *International Conference on Machine Learning (ICML)*, pages 639–646.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv*.

B. P. Miller, L. Fredriksen, and B. So. 1990. An empirical study of the reliability of UNIX utilities. *Communications of the ACM*, 33(12):32–44.

A. Neelakantan, Q. V. Le, and I. Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *International Conference on Learning Representations (ICLR)*.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

S. Reed and N. de Freitas. 2016. Neural programmer-interpreters. In *International Conference on Learning Representations (ICLR)*.

B. Settles. 2010. Active learning literature survey. Technical report, University of Wisconsin, Madison.

J. Steinhardt and P. Liang. 2015. Learning with relaxed supervision. In *Advances in Neural Information Processing Systems (NIPS)*.

P. Yin, Z. Lu, H. Li, and B. Kao. 2016. Neural enquirer: Learning to query tables with natural language. *arXiv*.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

Language to Logical Form with Neural Attention

Li Dong and **Mirella Lapata**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
li.dong@ed.ac.uk, mlap@inf.ed.ac.uk

Abstract

Semantic parsing aims at mapping natural language to machine interpretable meaning representations. Traditional approaches rely on high-quality lexicons, manually-built templates, and linguistic features which are either domain- or representation-specific. In this paper we present a general method based on an attention-enhanced encoder-decoder model. We encode input utterances into vector representations, and generate their logical forms by conditioning the output sequences or trees on the encoding vectors. Experimental results on four datasets show that our approach performs competitively without using hand-engineered features and is easy to adapt across domains and meaning representations.

1 Introduction

Semantic parsing is the task of translating text to a formal meaning representation such as logical forms or structured queries. There has recently been a surge of interest in developing machine learning methods for semantic parsing (see the references in Section 2), due in part to the existence of corpora containing utterances annotated with formal meaning representations. Figure 1 shows an example of a question (left hand-side) and its annotated logical form (right hand-side), taken from JOBS (Tang and Mooney, 2001), a well-known semantic parsing benchmark. In order to predict the correct logical form for a given utterance, most previous systems rely on predefined templates and manually designed features, which often render the parsing model domain- or representation-specific. In this work, we aim to use a simple yet effective method to bridge the gap between natural language and logical form with minimal domain knowledge.

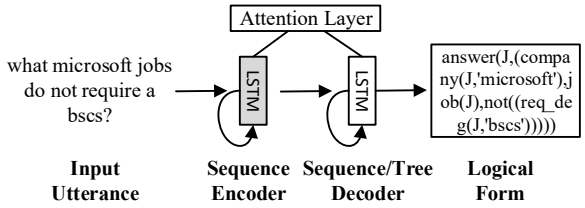


Figure 1: Input utterances and their logical forms are encoded and decoded with neural networks. An attention layer is used to learn soft alignments.

Encoder-decoder architectures based on recurrent neural networks have been successfully applied to a variety of NLP tasks ranging from syntactic parsing (Vinyals et al., 2015a), to machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), and image description generation (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015b). As shown in Figure 1, we adapt the general encoder-decoder paradigm to the semantic parsing task. Our model learns from natural language descriptions paired with meaning representations; it encodes sentences and decodes logical forms using recurrent neural networks with long short-term memory (LSTM) units. We present two model variants, the first one treats semantic parsing as a vanilla sequence transduction task, whereas our second model is equipped with a hierarchical tree decoder which explicitly captures the compositional structure of logical forms. We also introduce an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015b) allowing the model to learn soft alignments between natural language and logical forms and present an argument identification step to handle rare mentions of entities and numbers.

Evaluation results demonstrate that compared to previous methods our model achieves similar or better performance across datasets and meaning representations, despite using no hand-engineered domain- or representation-specific features.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 33–43,
Berlin, Germany, August 7-12, 2016.

2 Related Work

Our work synthesizes two strands of research, namely semantic parsing and the encoder-decoder architecture with neural networks.

The problem of learning semantic parsers has received significant attention, dating back to Woods (1973). Many approaches learn from sentences paired with logical forms following various modeling strategies. Examples include the use of parsing models (Miller et al., 1996; Ge and Mooney, 2005; Lu et al., 2008; Zhao and Huang, 2015), inductive logic programming (Zelle and Mooney, 1996; Tang and Mooney, 2000; Thompson and Mooney, 2003), probabilistic automata (He and Young, 2006), string/tree-to-tree transformation rules (Kate et al., 2005), classifiers based on string kernels (Kate and Mooney, 2006), machine translation (Wong and Mooney, 2006; Wong and Mooney, 2007; Andreas et al., 2013), and combinatory categorial grammar induction techniques (Zettlemoyer and Collins, 2005; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2010; Kwiatkowski et al., 2011). Other work learns semantic parsers without relying on logical-from annotations, e.g., from sentences paired with conversational logs (Artzi and Zettlemoyer, 2011), system demonstrations (Chen and Mooney, 2011; Goldwasser and Roth, 2011; Artzi and Zettlemoyer, 2013), question-answer pairs (Clarke et al., 2010; Liang et al., 2013), and distant supervision (Krishnamurthy and Mitchell, 2012; Cai and Yates, 2013; Reddy et al., 2014).

Our model learns from natural language descriptions paired with meaning representations. Most previous systems rely on high-quality lexicons, manually-built templates, and features which are either domain- or representation-specific. We instead present a general method that can be easily adapted to different domains and meaning representations. We adopt the general encoder-decoder framework based on neural networks which has been recently repurposed for various NLP tasks such as syntactic parsing (Vinyals et al., 2015a), machine translation (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014), image description generation (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015b), question answering (Hermann et al., 2015), and summarization (Rush et al., 2015).

Mei et al. (2016) use a sequence-to-sequence model to map navigational instructions to actions. Our model works on more well-defined meaning representations (such as Prolog and lambda calculus) and is conceptually simpler; it does not employ bidirectionality or multi-level alignments. Grefenstette et al. (2014) propose a different architecture for semantic parsing based on the combination of two neural network models. The first model learns shared representations from pairs of questions and their translations into knowledge base queries, whereas the second model generates the queries conditioned on the learned representations. However, they do not report empirical evaluation results.

3 Problem Formulation

Our aim is to learn a model which maps natural language input $q = x_1 \cdots x_{|q|}$ to a logical form representation of its meaning $a = y_1 \cdots y_{|a|}$. The conditional probability $p(a|q)$ is decomposed as:

$$p(a|q) = \prod_{t=1}^{|a|} p(y_t|y_{<t}, q) \tag{1}$$

where $y_{<t} = y_1 \cdots y_{t-1}$.

Our method consists of an **encoder** which encodes natural language input q into a vector representation and a **decoder** which learns to generate $y_1, \cdots, y_{|a|}$ conditioned on the encoding vector. In the following we describe two models varying in the way in which $p(a|q)$ is computed.

3.1 Sequence-to-Sequence Model

This model regards both input q and output a as sequences. As shown in Figure 2, the encoder and decoder are two different L-layer recurrent neural networks with long short-term memory (LSTM) units which recursively process tokens one by one. The first $|q|$ time steps belong to the encoder, while the following $|a|$ time steps belong to the decoder. Let $\mathbf{h}_t^l \in \mathbb{R}^n$ denote the hidden vector at time step t and layer l. $\mathbf{h}_t^l$ is then computed by:

$$\mathbf{h}_t^l = \text{LSTM}\left(\mathbf{h}_{t-1}^l, \mathbf{h}_t^{l-1}\right) \tag{2}$$

where LSTM refers to the LSTM function being used. In our experiments we follow the architecture described in Zaremba et al. (2015), however other types of gated activation functions are possible (e.g., Cho et al. (2014)). For the encoder, $\mathbf{h}_t^0 = \mathbf{W}_q \mathbf{e}(x_t)$ is the word vector of the current input token, with $\mathbf{W}_q \in \mathbb{R}^{n \times |V_q|}$ being a parameter matrix, and $\mathbf{e}(\cdot)$ the index of the corresponding

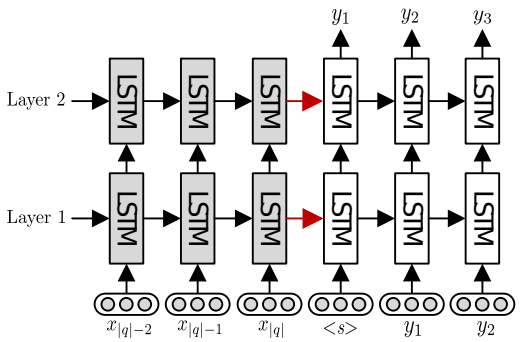


Figure 2: Sequence-to-sequence (SEQ2SEQ) model with two-layer recurrent neural networks.

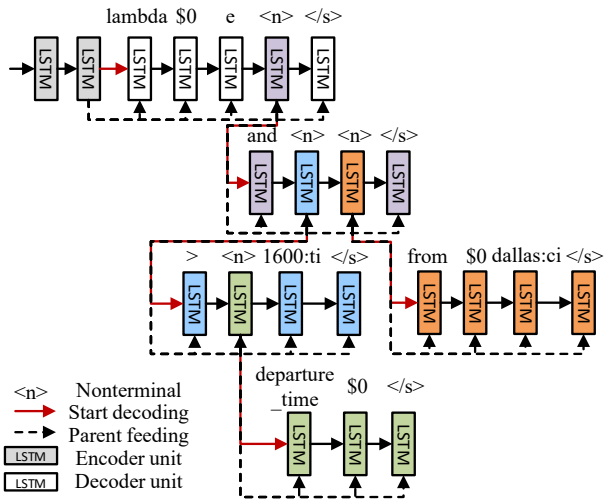


Figure 3: Sequence-to-tree (SEQ2TREE) model with a hierarchical tree decoder.

token. For the decoder, $\mathbf{h}_t^0 = \mathbf{W}_a \mathbf{e}(y_{t-1})$ is the word vector of the previous predicted word, where $\mathbf{W}_a \in \mathbb{R}^{n \times |V_a|}$. Notice that the encoder and decoder have different LSTM parameters.

Once the tokens of the input sequence $x_1, \cdots, x_{|q|}$ are encoded into vectors, they are used to initialize the hidden states of the first time step in the decoder. Next, the hidden vector of the topmost LSTM $\mathbf{h}_t^L$ in the decoder is used to predict the t-th output token as:

$$p\left(y_t | y_{<t}, q\right) = \text{softmax}\left(\mathbf{W}_o \mathbf{h}_t^L\right)^\intercal \mathbf{e}\left(y_t\right) \quad (3)$$

where $\mathbf{W}_o \in \mathbb{R}^{|V_a| \times n}$ is a parameter matrix, and $\mathbf{e}\left(y_t\right) \in \{0,1\}^{|V_a|}$ a one-hot vector for computing y_t's probability from the predicted distribution.

We augment every sequence with a "start-of-sequence" <*s*> and "end-of-sequence" </*s*> token. The generation process terminates once </*s*> is predicted. The conditional probability of generating the whole sequence $p\left(a|q\right)$ is then obtained using Equation (1).

3.2 Sequence-to-Tree Model

The SEQ2SEQ model has a potential drawback in that it ignores the hierarchical structure of logical forms. As a result, it needs to memorize various pieces of auxiliary information (e.g., bracket pairs) to generate well-formed output. In the following we present a hierarchical tree decoder which is more faithful to the compositional nature of meaning representations. A schematic description of the model is shown in Figure 3.

The present model shares the same encoder with the sequence-to-sequence model described in Section 3.1 (essentially it learns to encode input q as vectors). However, its decoder is fundamentally different as it generates logical forms in a top-down manner. In order to represent tree structure, we define a "nonterminal" <*n*> token which indicates subtrees. As shown in Figure 3, we preprocess the logical form "*lambda $0 e (and (>(departure_time $0) 1600:ti) (from $0 dallas:ci))*" to a tree by replacing tokens between pairs of brackets with nonterminals. Special tokens <*s*> and <*(*> denote the beginning of a sequence and nonterminal sequence, respectively (omitted from Figure 3 due to lack of space). Token </*s*> represents the end of sequence.

After encoding input q, the hierarchical tree decoder uses recurrent neural networks to generate tokens at depth 1 of the subtree corresponding to parts of logical form a. If the predicted token is <*n*>, we decode the sequence by conditioning on the nonterminal's hidden vector. This process terminates when no more nonterminals are emitted. In other words, a sequence decoder is used to hierarchically generate the tree structure.

In contrast to the sequence decoder described in Section 3.1, the current hidden state does not only depend on its previous time step. In order to better utilize the parent nonterminal's information, we introduce a *parent-feeding* connection where the hidden vector of the parent nonterminal is concatenated with the inputs and fed into LSTM.

As an example, Figure 4 shows the decoding tree corresponding to the logical form "*A B (C)*", where $y_1 \cdots y_6$ are predicted tokens, and $t_1 \cdots t_6$ denote different time steps. Span "*(C)*" corresponds to a subtree. Decoding in this example has two steps: once input q has been encoded, we first generate $y_1 \cdots y_4$ at depth 1 until token </*s*> is

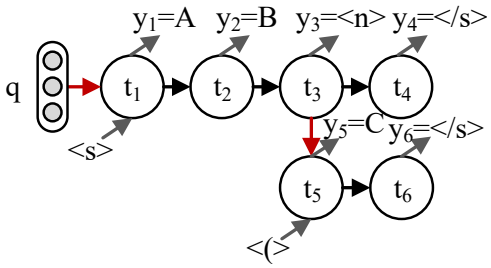


Figure 4: A SEQ2TREE decoding example for the logical form "*A B (C)*".

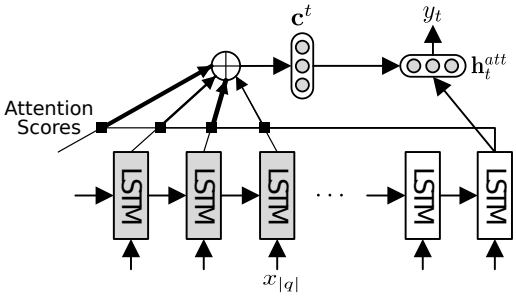


Figure 5: Attention scores are computed by the current hidden vector and all the hidden vectors of encoder. Then, the encoder-side context vector $\mathbf{c}^t$ is obtained in the form of a weighted sum, which is further used to predict y_t.

predicted; next, we generate y_5, y_6 by conditioning on nonterminal t_3's hidden vectors. The probability $p(a|q)$ is the product of these two sequence decoding steps:

$$p(a|q) = p(y_1 y_2 y_3 y_4|q)\, p(y_5 y_6|y_{\leq 3}, q) \quad (4)$$

where Equation (3) is used for the prediction of each output token.

3.3 Attention Mechanism

As shown in Equation (3), the hidden vectors of the input sequence are not directly used in the decoding process. However, it makes intuitively sense to consider relevant information from the input to better predict the current token. Following this idea, various techniques have been proposed to integrate encoder-side information (in the form of a context vector) at each time step of the decoder (Bahdanau et al., 2015; Luong et al., 2015b; Xu et al., 2015).

As shown in Figure 5, in order to find relevant encoder-side context for the current hidden state $\mathbf{h}_t^L$ of decoder, we compute its attention score with the k-th hidden state in the encoder as:

$$s_k^t = \frac{\exp\{\mathbf{h}_k^L \cdot \mathbf{h}_t^L\}}{\sum_{j=1}^{|q|} \exp\{\mathbf{h}_j^L \cdot \mathbf{h}_t^L\}} \quad (5)$$

where $\mathbf{h}_1^L, \cdots, \mathbf{h}_{|q|}^L$ are the top-layer hidden vectors of the encoder. Then, the context vector is the weighted sum of the hidden vectors in the encoder:

$$\mathbf{c}^t = \sum_{k=1}^{|q|} s_k^t \mathbf{h}_k^L \quad (6)$$

In lieu of Equation (3), we further use this context vector which acts as a summary of the encoder to compute the probability of generating y_t as:

$$\mathbf{h}_t^{att} = \tanh\left(\mathbf{W}_1 \mathbf{h}_t^L + \mathbf{W}_2 \mathbf{c}^t\right) \quad (7)$$

$$p(y_t|y_{<t}, q) = \text{softmax}\left(\mathbf{W}_o \mathbf{h}_t^{att}\right)^{\mathsf{T}} \mathbf{e}(y_t) \quad (8)$$

where $\mathbf{W}_o \in \mathbb{R}^{|V_a| \times n}$ and $\mathbf{W}_1, \mathbf{W}_2 \in \mathbb{R}^{n \times n}$ are three parameter matrices, and $\mathbf{e}(y_t)$ is a one-hot vector used to obtain y_t's probability.

3.4 Model Training

Our goal is to maximize the likelihood of the generated logical forms given natural language utterances as input. So the objective function is:

$$\text{minimize} - \sum_{(q,a)\in\mathcal{D}} \log p(a|q) \quad (9)$$

where $\mathcal{D}$ is the set of all natural language-logical form training pairs, and $p(a|q)$ is computed as shown in Equation (1).

The RMSProp algorithm (Tieleman and Hinton, 2012) is employed to solve this non-convex optimization problem. Moreover, dropout is used for regularizing the model (Zaremba et al., 2015). Specifically, dropout operators are used between different LSTM layers and for the hidden layers before the softmax classifiers. This technique can substantially reduce overfitting, especially on datasets of small size.

3.5 Inference

At test time, we predict the logical form for an input utterance q by:

$$\hat{a} = \arg\max_{a'} p\left(a'|q\right) \quad (10)$$

where a' represents a candidate output. However, it is impractical to iterate over all possible results to obtain the optimal prediction. According to Equation (1), we decompose the probability $p(a|q)$ so that we can use greedy search (or beam search) to generate tokens one by one.

Algorithm 1 Decoding for SEQ2TREE

Input: q: Natural language utterance
Output: $\hat{a}$: Decoding result

$\triangleright$ *Push the encoding result to a queue*
$Q.init(\{hid : \mathsf{SeqEnc}(q)\})$
$\triangleright$ *Decode until no more nonterminals*
while $(c \leftarrow Q.pop()) \neq \varnothing$ **do**
    $\triangleright$ *Call sequence decoder*
    $c.child \leftarrow \mathsf{SeqDec}(c.hid)$
    $\triangleright$ *Push new nonterminals to queue*
    for $n \leftarrow$ nonterminal in $c.child$ **do**
        $Q.push(\{hid : \mathsf{HidVec}(n)\})$
$\hat{a} \leftarrow$ convert decoding tree to output sequence

Algorithm 1 describes the decoding process for SEQ2TREE. The time complexity of both decoders is $\mathcal{O}(|a|)$, where $|a|$ is the length of output. The extra computation of SEQ2TREE compared with SEQ2SEQ is to maintain the nonterminal queue, which can be ignored because most of time is spent on matrix operations. We implement the hierarchical tree decoder in a batch mode, so that it can fully utilize GPUs. Specifically, as shown in Algorithm 1, every time we pop multiple nonterminals from the queue and decode these nonterminals in one batch.

3.6 Argument Identification

The majority of semantic parsing datasets have been developed with question-answering in mind. In the typical application setting, natural language questions are mapped into logical forms and executed on a knowledge base to obtain an answer. Due to the nature of the question-answering task, many natural language utterances contain entities or numbers that are often parsed as arguments in the logical form. Some of them are unavoidably rare or do not appear in the training set at all (this is especially true for small-scale datasets). Conventional sequence encoders simply replace rare words with a special unknown word symbol (Luong et al., 2015a; Jean et al., 2015), which would be detrimental for semantic parsing.

We have developed a simple procedure for argument identification. Specifically, we identify entities and numbers in input questions and replace them with their type names and unique IDs. For instance, we pre-process the training example "*jobs with a salary of 40000*" and its logical form "*job(ANS), salary_greater_than(ANS, 40000, year)*" as "*jobs with a salary of* $\underline{num_0}$" and "*job(ANS), salary_greater_than(ANS,* $\underline{num_0}$*, year)*". We use the pre-processed examples as training data. At inference time, we also mask entities and numbers with their types and IDs. Once we obtain the decoding result, a post-processing step recovers all the markers $\underline{type_i}$ to their corresponding logical constants.

4 Experiments

We compare our method against multiple previous systems on four datasets. We describe these datasets below, and present our experimental settings and results. Finally, we conduct model analysis in order to understand what the model learns. The code is available at `https://github.com/donglixp/lang2logic`.

4.1 Datasets

Our model was trained on the following datasets, covering different domains and using different meaning representations. Examples for each domain are shown in Table 1.

JOBS This benchmark dataset contains 640 queries to a database of job listings. Specifically, questions are paired with Prolog-style queries. We used the same training-test split as Zettlemoyer and Collins (2005) which contains 500 training and 140 test instances. Values for the variables company, degree, language, platform, location, job area, and number are identified.

GEO This is a standard semantic parsing benchmark which contains 880 queries to a database of U.S. geography. GEO has 880 instances split into a training set of 680 training examples and 200 test examples (Zettlemoyer and Collins, 2005). We used the same meaning representation based on lambda-calculus as Kwiatkowski et al. (2011). Values for the variables city, state, country, river, and number are identified.

ATIS This dataset has 5,410 queries to a flight booking system. The standard split has 4,480 training instances, 480 development instances, and 450 test instances. Sentences are paired with lambda-calculus expressions. Values for the variables date, time, city, aircraft code, airport, airline, and number are identified.

IFTTT Quirk et al. (2015) created this dataset by extracting a large number of if-this-then-that

Dataset	Length	Example
JOBS	9.80	*what microsoft jobs do not require a bscs?*
	22.90	answer(company(J,'microsoft'),job(J),not((req_deg(J,'bscs'))))
GEO	7.60	*what is the population of the state with the largest area?*
	19.10	(population:i (argmax $0 (state:t $0) (area:i $0)))
ATIS	11.10	*dallas to san francisco leaving after 4 in the afternoon please*
	28.10	(lambda $0 e (and (>(departure_time $0) 1600:ti) (from $0 dallas:ci) (to $0 san_francisco:ci)))
IFTTT	6.95	*Turn on heater when temperature drops below 58 degree*
	21.80	TRIGGER: Weather - Current_temperature_drops_below - ((Temperature (58)) (Degrees_in (f)))
		ACTION: WeMo_Insight_Switch - Turn_on - ((Which_switch? ("")))

Table 1: Examples of natural language descriptions and their meaning representations from four datasets. The average length of input and output sequences is shown in the second column.

recipes from the IFTTT website[1]. Recipes are simple programs with exactly one trigger and one action which users specify on the site. Whenever the conditions of the trigger are satisfied, the action is performed. Actions typically revolve around home security (e.g., "*turn on my lights when I arrive home*"), automation (e.g., "*text me if the door opens*"), well-being (e.g., "*remind me to drink water if I've been at a bar for more than two hours*"), and so on. Triggers and actions are selected from different channels (160 in total) representing various types of services, devices (e.g., Android), and knowledge sources (such as ESPN or Gmail). In the dataset, there are 552 trigger functions from 128 channels, and 229 action functions from 99 channels. We used Quirk et al.'s (2015) original split which contains $77,495$ training, $5,171$ development, and $4,294$ test examples. The IFTTT programs are represented as abstract syntax trees and are paired with natural language descriptions provided by users (see Table 1). Here, numbers and URLs are identified.

4.2 Settings

Natural language sentences were lowercased; misspellings were corrected using a dictionary based on the Wikipedia list of common misspellings. Words were stemmed using NLTK (Bird et al., 2009). For IFTTT, we filtered tokens, channels and functions which appeared less than five times in the training set. For the other datasets, we filtered input words which did not occur at least two times in the training set, but kept all tokens in the logical forms. Plain string matching was employed to identify augments as described in Section 3.6. More sophisticated approaches could be used, however we leave this future work.

Model hyper-parameters were cross-validated on the training set for JOBS and GEO. We used the standard development sets for ATIS and IFTTT. We used the RMSProp algorithm (with batch size set to 20) to update the parameters. The smoothing constant of RMSProp was 0.95. Gradients were clipped at 5 to alleviate the exploding gradient problem (Pascanu et al., 2013). Parameters were randomly initialized from a uniform distribution $\mathcal{U}(-0.08, 0.08)$. A two-layer LSTM was used for IFTTT, while a one-layer LSTM was employed for the other domains. The dropout rate was selected from $\{0.2, 0.3, 0.4, 0.5\}$. Dimensions of hidden vector and word embedding were selected from $\{150, 200, 250\}$. Early stopping was used to determine the number of epochs. Input sentences were reversed before feeding into the encoder (Sutskever et al., 2014). We use greedy search to generate logical forms during inference. Notice that two decoders with shared word embeddings were used to predict triggers and actions for IFTTT, and two softmax classifiers are used to classify channels and functions.

Method	Accuracy
COCKTAIL (Tang and Mooney, 2001)	79.4
PRECISE (Popescu et al., 2003)	88.0
ZC05 (Zettlemoyer and Collins, 2005)	79.3
DCS+L (Liang et al., 2013)	90.7
TISP (Zhao and Huang, 2015)	85.0
SEQ2SEQ	87.1
− attention	77.9
− argument	70.7
SEQ2TREE	90.0
− attention	83.6

Table 2: Evaluation results on JOBS.

4.3 Results

We first discuss the performance of our model on JOBS, GEO, and ATIS, and then examine our results on IFTTT. Tables 2–4 present comparisons against a variety of systems previously described

[1] http://www.ifttt.com

Method	Accuracy
SCISSOR (Ge and Mooney, 2005)	72.3
KRISP (Kate and Mooney, 2006)	71.7
WASP (Wong and Mooney, 2006)	74.8
λ-WASP (Wong and Mooney, 2007)	86.6
LNLZ08 (Lu et al., 2008)	81.8
ZC05 (Zettlemoyer and Collins, 2005)	79.3
ZC07 (Zettlemoyer and Collins, 2007)	86.1
UBL (Kwiatkowski et al., 2010)	87.9
FUBL (Kwiatkowski et al., 2011)	88.6
KCAZ13 (Kwiatkowski et al., 2013)	89.0
DCS+L (Liang et al., 2013)	87.9
TISP (Zhao and Huang, 2015)	88.9
SEQ2SEQ	84.6
− attention	72.9
− argument	68.6
SEQ2TREE	87.1
− attention	76.8

Table 3: Evaluation results on GEO. 10-fold cross-validation is used for the systems shown in the top half of the table. The standard split of ZC05 is used for all other systems.

Method	Accuracy
ZC07 (Zettlemoyer and Collins, 2007)	84.6
UBL (Kwiatkowski et al., 2010)	71.4
FUBL (Kwiatkowski et al., 2011)	82.8
GUSP-FULL (Poon, 2013)	74.8
GUSP++ (Poon, 2013)	83.5
TISP (Zhao and Huang, 2015)	84.2
SEQ2SEQ	84.2
− attention	75.7
− argument	72.3
SEQ2TREE	84.6
− attention	77.5

Table 4: Evaluation results on ATIS.

Method	Channel	+Func	F1
retrieval	28.9	20.2	41.7
phrasal	19.3	11.3	35.3
sync	18.1	10.6	35.1
classifier	48.8	35.2	48.4
posclass	50.0	36.9	49.3
SEQ2SEQ	54.3	39.2	50.1
− attention	54.0	37.9	49.8
− argument	53.9	38.6	49.7
SEQ2TREE	55.2	40.1	50.4
− attention	54.3	38.2	50.0

(a) Omit non-English.

Method	Channel	+Func	F1
retrieval	36.8	25.4	49.0
phrasal	27.8	16.4	39.9
sync	26.7	15.5	37.6
classifier	64.8	47.2	56.5
posclass	67.2	50.4	57.7
SEQ2SEQ	68.8	50.5	60.3
− attention	68.7	48.9	59.5
− argument	68.8	50.4	59.7
SEQ2TREE	69.6	51.4	60.4
− attention	68.7	49.5	60.2

(b) Omit non-English & unintelligible.

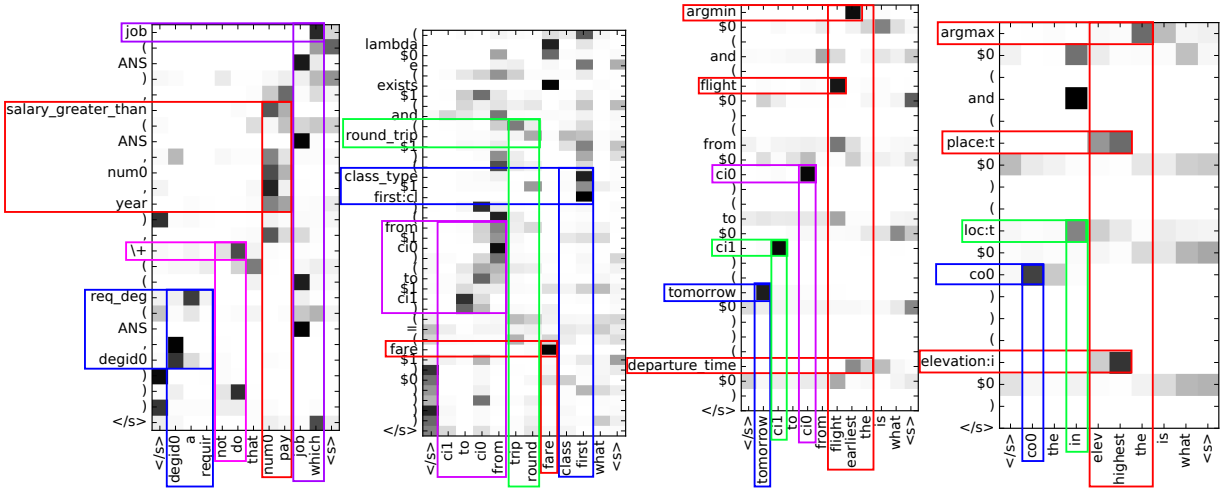
Method	Channel	+Func	F1
retrieval	43.3	32.3	56.2
phrasal	37.2	23.5	45.5
sync	36.5	24.1	42.8
classifier	79.3	66.2	65.0
posclass	81.4	71.0	66.5
SEQ2SEQ	87.8	75.2	73.7
− attention	88.3	73.8	72.9
− argument	86.8	74.9	70.8
SEQ2TREE	89.7	78.4	74.2
− attention	87.6	74.9	73.5

(c) ≥ 3 turkers agree with gold.

Table 5: Evaluation results on IFTTT.

in the literature. We report results with the full models (SEQ2SEQ, SEQ2TREE) and two ablation variants, i.e., without an attention mechanism (−attention) and without argument identification (−argument). We report accuracy which is defined as the proportion of the input sentences that are correctly parsed to their gold standard logical forms. Notice that DCS+L, KCAZ13 and GUSP output answers directly, so accuracy in this setting is defined as the percentage of correct answers.

Overall, SEQ2TREE is superior to SEQ2SEQ. This is to be expected since SEQ2TREE explicitly models compositional structure. On the JOBS and GEO datasets which contain logical forms with nested structures, SEQ2TREE outperforms SEQ2SEQ by 2.9% and 2.5%, respectively. SEQ2TREE achieves better accuracy over SEQ2SEQ on ATIS too, however, the difference is smaller, since ATIS is a simpler domain without complex nested structures. We find that adding attention substantially improves performance on all three datasets. This underlines the importance of utilizing soft alignments between inputs and outputs. We further analyze what the attention layer learns in Figure 6. Moreover, our results show that argument identification is critical for small-scale datasets. For example, about 92% of city names appear less than 4 times in the GEO training set, so it is difficult to learn reliable parameters for these words. In relation to previous work, the proposed models achieve comparable or better performance. Importantly, we use the same framework (SEQ2SEQ or SEQ2TREE) across datasets and meaning representations (Prolog-style logical forms in JOBS and lambda calculus in the other two datasets) without modification. Despite this relatively simple approach, we observe that SEQ2TREE ranks second on JOBS, and is tied for first place with ZC07 on ATIS.



(a) which jobs pay num0 that do not require a degid0

(b) what's first class fare round trip from ci0 to ci1

(c) what is the earliest flight from ci0 to ci1 tomorrow

(d) what is the highest elevation in the co0

Figure 6: Alignments (same color rectangles) produced by the attention mechanism (darker color represents higher attention score). Input sentences are reversed and stemmed. Model output is shown for SEQ2SEQ (a, b) and SEQ2TREE (c, d).

We illustrate examples of alignments produced by SEQ2SEQ in Figures 6a and 6b. Alignments produced by SEQ2TREE are shown in Figures 6c and 6d. Matrices of attention scores are computed using Equation (5) and are represented in grayscale. Aligned input words and logical form predicates are enclosed in (same color) rectangles whose overlapping areas contain the attention scores. Also notice that attention scores are computed by LSTM hidden vectors which encode context information rather than just the words in their current positions. The examples demonstrate that the attention mechanism can successfully model the correspondence between sentences and logical forms, capturing reordering (Figure 6b), many-to-many (Figure 6a), and many-to-one alignments (Figures 6c,d).

For IFTTT, we follow the same evaluation protocol introduced in Quirk et al. (2015). The dataset is extremely noisy and measuring accuracy is problematic since predicted abstract syntax trees (ASTs) almost never exactly match the gold standard. Quirk et al. view an AST as a set of productions and compute balanced F1 instead which we also adopt. The first column in Table 5 shows the percentage of channels selected correctly for both triggers and actions. The second column measures accuracy for *both* channels and functions. The last column shows balanced F1 against the gold tree over all productions in the proposed derivation. We compare our model against posclass, the method introduced in Quirk et al. and several of their baselines. posclass is reminiscent of KRISP (Kate and Mooney, 2006), it learns distributions over productions given input sentences represented as a bag of linguistic features. The retrieval baseline finds the closest description in the training data based on character string-edit-distance and returns the recipe for that training program. The phrasal method uses phrase-based machine translation to generate the recipe, whereas sync extracts synchronous grammar rules from the data, essentially recreating WASP (Wong and Mooney, 2006). Finally, they use a binary classifier to predict whether a production should be present in the derivation tree corresponding to the description.

Quirk et al. (2015) report results on the full test data and smaller subsets after noise filtering, e.g., when non-English and unintelligible descriptions are removed (Tables 5a and 5b). They also ran their system on a high-quality subset of description-program pairs which were found in the gold standard and at least three humans managed to independently reproduce (Table 5c). Across all subsets our models outperforms posclass and related baselines. Again we observe that SEQ2TREE consistently outperforms SEQ2SEQ, albeit with a small margin. Compared to the previous datasets, the attention mechanism and our argument iden-

tification method yield less of an improvement. This may be due to the size of Quirk et al. (2015) and the way it was created – user curated descriptions are often of low quality, and thus align very loosely to their corresponding ASTs.

4.4 Error Analysis

Finally, we inspected the output of our model in order to identify the most common causes of errors which we summarize below.

Under-Mapping The attention model used in our experiments does not take the alignment history into consideration. So, some question words, expecially in longer questions, may be ignored in the decoding process. This is a common problem for encoder-decoder models and can be addressed by explicitly modelling the decoding coverage of the source words (Tu et al., 2016; Cohn et al., 2016). Keeping track of the attention history would help adjust future attention and guide the decoder towards untranslated source words.

Argument Identification Some mentions are incorrectly identified as arguments. For example, the word *may* is sometimes identified as a month when it is simply a modal verb. Moreover, some argument mentions are ambiguous. For instance, *6 o'clock* can be used to express either *6 am* or *6 pm*. We could disambiguate arguments based on contextual information. The execution results of logical forms could also help prune unreasonable arguments.

Rare Words Because the data size of JOBS, GEO, and ATIS is relatively small, some question words are rare in the training set, which makes it hard to estimate reliable parameters for them. One solution would be to learn word embeddings on unannotated text data, and then use these as pretrained vectors for question words.

5 Conclusions

In this paper we presented an encoder-decoder neural network model for mapping natural language descriptions to their meaning representations. We encode natural language utterances into vectors and generate their corresponding logical forms as sequences or trees using recurrent neural networks with long short-term memory units. Experimental results show that enhancing the model with a hierarchical tree decoder and an attention mechanism improves performance across the board. Extensive comparisons with previous methods show that our approach performs competitively, without recourse to domain- or representation-specific features. Directions for future work are many and varied. For example, it would be interesting to learn a model from question-answer pairs without access to target logical forms. Beyond semantic parsing, we would also like to apply our SEQ2TREE model to related structured prediction tasks such as constituency parsing.

Acknowledgments We would like to thank Luke Zettlemoyer and Tom Kwiatkowski for sharing the ATIS dataset. The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st ACL*, pages 47–52, Sofia, Bulgaria.

Yoav Artzi and Luke Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Proceedings of the 2011 EMNLP*, pages 421–432, Edinburgh, United Kingdom.

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

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the ICLR*, San Diego, California.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *2nd Joint Conference on Lexical and Computational Semantics*, pages 328–338, Atlanta, Georgia.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 15th AAAI*, pages 859–865, San Francisco, California.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 EMNLP*, pages 1724–1734, Doha, Qatar.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of CONLL*, pages 18–27, Uppsala, Sweden.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating structural alignment biases into an attentional neural translation model. In *Proceedings of the 2016 NAACL*, San Diego, California.

Ruifang Ge and Raymond J. Mooney. 2005. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of CoNLL*, pages 9–16, Ann Arbor, Michigan.

Dan Goldwasser and Dan Roth. 2011. Learning from natural instructions. In *Proceedings of the 22nd IJCAI*, pages 1794–1800, Barcelona, Spain.

Edward Grefenstette, Phil Blunsom, Nando de Freitas, and Karl Moritz Hermann. 2014. A deep architecture for semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, Atlanta, Georgia.

Yulan He and Steve Young. 2006. Semantic processing using the hidden vector state model. *Speech Communication*, 48(3-4):262–275.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1684–1692. Curran Associates, Inc.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of 53rd ACL and 7th IJCNLP*, pages 1–10, Beijing, China.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 EMNLP*, pages 1700–1709, Seattle, Washington.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of CVPR*, pages 3128–3137, Boston, Massachusetts.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st COLING and 44th ACL*, pages 913–920, Sydney, Australia.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the 20th AAAI*, pages 1062–1068, Pittsburgh, Pennsylvania.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 EMNLP*, pages 754–765, Jeju Island, Korea.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 EMNLP*, pages 1223–1233, Cambridge, Massachusetts.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Proceedings of the 2011 EMNLP*, pages 1512–1523, Edinburgh, United Kingdom.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 EMNLP*, pages 1545–1556, Seattle, Washington.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 EMNLP*, pages 783–792, Honolulu, Hawaii.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015a. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd ACL and 7th IJCNLP*, pages 11–19, Beijing, China.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015b. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 EMNLP*, pages 1412–1421, Lisbon, Portugal.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the 30th AAAI*, Phoenix, Arizona. to appear.

Scott Miller, David Stallard, Robert Bobrow, and Richard Schwartz. 1996. A fully statistical approach to natural language interfaces. In *ACL*, pages 55–61.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th ICML*, pages 1310–1318, Atlanta, Georgia.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st ACL*, pages 933–943, Sofia, Bulgaria.

Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. 2003. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th IUI*, pages 149–157, Miami, Florida.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of 53rd ACL and 7th IJCNLP*, pages 878–888, Beijing, China.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *TACL*, 2(Oct):377–392.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 EMNLP*, pages 379–389, Lisbon, Portugal.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Lappoon R. Tang and Raymond J. Mooney. 2000. Automated construction of database interfaces: Intergrating statistical and relational learning for semantic parsing. In *Proceedings of the 2000 EMNLP*, pages 133–141, Hong Kong, China.

Lappoon R. Tang and Raymond J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th ECML*, pages 466–477, Freiburg, Germany.

Cynthia A. Thomspon and Raymond J. Mooney. 2003. Acquiring word-meaning mappings for natural language interfaces. *Journal of Artifical Intelligence Research*, 18:1–44.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. Technical report.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th ACL*, Berlin, Germany.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015a. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2755–2763. Curran Associates, Inc.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015b. Show and tell: A neural image caption generator. In *Proceedings of CVPR*, pages 3156–3164, Boston, Massachusetts.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the 2006 NAACL*, pages 439–446, New York, New York.

Yuk Wah Wong and Raymond J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th ACL*, pages 960–967, Prague, Czech Republic.

W. A. Woods. 1973. Progress in natural language understanding: An application to lunar geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition*, pages 441–450, New York, NY.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd ICML*, pages 2048–2057, Lille, France.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2015. Recurrent neural network regularization. In *Proceedings of the ICLR*, San Diego, California.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the 19th AAAI*, pages 1050–1055, Portland, Oregon.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of the 21st UAI*, pages 658–666, Toronto, ON.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *In Proceedings of the EMNLP-CoNLL*, pages 678–687, Prague, Czech Republic.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 NAACL*, pages 1416–1421, Denver, Colorado.

Unsupervised Person Slot Filling based on Graph Mining

Dian Yu **Heng Ji**
Computer Science Department
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{yud2,jih}@rpi.edu

Abstract

Slot filling aims to extract the values (*slot fillers*) of specific attributes (*slots types*) for a given entity (*query*) from a large-scale corpus. Slot filling remains very challenging over the past seven years. We propose a simple yet effective unsupervised approach to extract slot fillers based on the following two observations: (1) a trigger is usually a salient node relative to the query and filler nodes in the dependency graph of a context sentence; (2) a relation is likely to exist if the query and candidate filler nodes are strongly connected by a relation-specific trigger. Thus we design a graph-based algorithm to automatically identify triggers based on personalized PageRank and Affinity Propagation for a given (*query*, *filler*) pair and then label the slot type based on the identified triggers. Our approach achieves 11.6%-25% higher F-score over state-of-the-art English slot filling methods. Our experiments also demonstrate that as long as a few trigger seeds, name tagging and dependency parsing capabilities exist, this approach can be quickly adapted to any language and new slot types. Our promising results on Chinese slot filling can serve as a new benchmark.

1 Introduction

The goal of the Text Analysis Conference Knowledge Base Population (TAC-KBP) Slot Filling (SF) task (McNamee and Dang, 2009; Ji et al., 2010; Ji et al., 2011; Surdeanu and Ji, 2014) is to extract the values (***fillers***) of specific attributes (***slot types***) for a given entity (***query***) from a large-scale corpus and provide justification sentences to support these slot fillers. KBP defines 25 slot types for persons (*e.g.*, *spouse*) and 16 slots for organizations (*e.g.*, *founder*). For example, given a person query "*Dominich Dunne*" and slot type *spouse*, a SF system may extract a slot filler "*Ellen Griffin*" and its justification sentence $E1$ as shown in Figure 1.

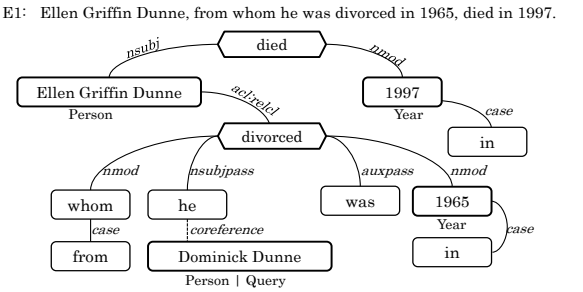


Figure 1: Extended dependency tree for E1.

Slot filling remains a very challenging task. The two most successful state-of-the-art techniques are as follows.

(1) Supervised classification. Considering any pair of query and candidate slot filler as an instance, these approaches train a classifier from manually labeled data through active learning (Angeli et al., 2014b) or noisy labeled data through distant supervision (Angeli et al., 2014a; Surdeanu et al., 2010) to predict the existence of a specific relation between them.

(2) Pattern matching. These approaches extract and generalize lexical and syntactic patterns automatically or semi-automatically (Sun et al., 2011; Li et al., 2012; Yu et al., 2013; Hong et al., 2014). They usually suffer from low recall due to numerous different ways to express a certain relation type (Surdeanu and Ji, 2014). For example, none of the top-ranked patterns (Li et al., 2012) based on dependency paths in Table 1 can capture the *spouse* slot in *E1*.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 44–53,
Berlin, Germany, August 7-12, 2016.

Query	poss^{-1}	Slot Filler
Query	poss^{-1} [**wife**-**widow**-**husband**] appos	Slot Filler
Query	nsubj^{-1} **married** dobj	Slot Filler
Query	appos **wife** prep_of	Slot Filler
Query	nsubjpass^{-1} **survived** agent	Slot Filler

Table 1: Dependency patterns for slot *spouse*.

Both of the previous methods have poor portability to a new language or a new slot type. Furthermore, both methods focus on the flat relation representation between the query and the candidate slot filler, while ignoring the global graph structure among them and other facts in the context.

When multiple facts about a person entity are presented in a sentence, the author (e.g., a news reporter or a discussion forum poster) often uses explicit *trigger* words or phrases to indicate their relations with the entity. As a result, these interdependent facts and query entities are strongly connected via syntactic or semantic relations.

Many slot types, especially when the queries are person entities, are indicated by such *triggers*. We call these slots ***trigger-driven*** slots. In this paper, we define a ***trigger*** as the smallest extent of a text which most clearly indicates a slot type. For example, in *E1*, "*divorced*" is a trigger for *spouse* while "*died*" is a trigger for death-related slots.

Considering the limitations of previous flat representations for the relations between a query (Q) and a candidate slot filler (F), we focus on analyzing the whole dependency tree structure that connects Q, F and other semantically related words or phrases in each context sentence. Our main observation is that there often exists a trigger word (T) which plays an important role in connecting Q and F in the dependency tree for trigger-driven slots. From the extended dependency tree shown in Figure 1, we can clearly see that "*divorced*" is most strongly connected to the query mention ("*he*") and the slot filler ("*Ellen Griffin Dunne*"). Therefore we can consider it as a trigger word which explicitly indicates a particular slot type.

Based on these observations, we propose a novel and effective **unsupervised graph mining** approach for person slot filling by deeply exploring the structures of dependency trees. It consists of the following three steps:

- **Step 1 - Candidate Relation Identification:** Construct an extended dependency tree for each sentence including any mention referring to the query entity. Identify candidate slot fillers based on slot type constraints (*e.g.*, the *spouse* fillers are limited to person entities) (Section 2).
- **Step 2 - Trigger Identification:** Measure the importance of each node in the extended dependency tree relative to Q and F, rank them and select the most important ones as the trigger set (Section 3).
- **Step 3 - Slot Typing:** For any given new slot type, automatically expand a few trigger seeds using the Paraphrase Database (Ganitkevitch et al., 2013). Then we use the expanded trigger set to label the slot types of identified triggers (Section 4).

This framework only requires name tagging and dependency parsing as pre-processing, and a few trigger seeds as input, and thus it can be easily adapted to a new language or a new slot type. Experiments on English and Chinese demonstrate that our approach dramatically advances state-of-the-art results for both pre-defined KBP slot types and new slot types.

2 Candidate Relation Identification

We first present how to build an extended dependency graph for each evidence sentence (Section 2.1) and generate query and filler candidate mentions (Section 2.2).

2.1 Extended Dependency Tree Construction

Given a sentence containing N words, we construct an undirected graph $G = (V, E)$, where $V = \{v_1, \ldots, v_N\}$ represents the words in a sentence, E is an edge set, associated with each edge e_{ij} representing a dependency relation between v_i and v_j. We first apply a dependency parser to generate basic uncollapsed dependencies by ignoring the direction of edges. Figure 1 shows the dependency tree built from the example sentence. In addition, we annotate an entity, time or value mention node with its type. For example, in Figure 1, "*Ellen Griffin Dunne*" is annotated as a person, and "*1997*" is annotated as a year. Finally we perform co-reference resolution, which introduces implicit links between nodes that refer to the same entity. We replace any nominal or pronominal entity mention with its coreferential name mention. For example, "*he*" is replaced by "*Dominick Dunne*" in Figure 1. Formally, an extended dependency tree is an annotated tree of entity mentions, phrases and their links.

2.2 Query Mention and Filler Candidate Identification

Given a query q and a set of relevant documents, we construct a dependency tree for each sentence. We identify a person entity e as a query mention if e matches the last name of q or e shares two or more tokens with q. For example, "*he/Dominick Dunne*" in Figure 1 is identified as a mention referring to the query *Dominick Dunne*. For each sentence which contains at least one query mention, we regard all other entities, values and time expressions as candidate fillers and generate a set of entity pairs (q, f), where q is a query mention, and f is a candidate filler. In Example $E1$, we can extract three entity pairs (*i.e.*, {*Dominick Dunne*} $\times$ {*Ellen Griffin Dunne*, 1997, 1965}). For each entity pair, we represent the query mention and the filler candidate as two sets of nodes Q and F respectively, where $Q, F \subseteq V$.

3 Trigger Identification

In this section, we proceed to introduce an unsupervised graph-based method to identify triggers for each query and candidate filler pair. We rank all trigger candidates (Section 3.1) and then keep the top ones as the trigger set (Section 3.2).

3.1 Trigger Candidate Ranking

As we have discussed in Section 1, we can consider trigger identification problem as finding the important nodes relative to Q and F in G. Algorithms such as Pagerank (Page et al., 1999) are designed to compute the global importance of each node relative to all other nodes in a graph. By redefining the importance according to our preference toward F and Q, we can extend PageRank to generate relative importance scores.

We use the random surfer model (Page et al., 1999) to explain our motivation. Suppose a random surfer keeps visiting adjacent nodes in G at random. The expected percentage of surfers visiting each node converges to the PageRank score. We extend PageRank by introducing a "back probability" β to determine how often surfers jump back to the ***preferred nodes*** (*i.e.*, Q or F) so that the converged score can be used to estimate the relative probability of visiting these preferred nodes.

Given G and a set of preferred nodes R where $R \subseteq V$, we denote the relative importance for all $v \in V$ with respect to R as $I(v \mid R)$, following the work of White and Smyth (2003).

For a node v_k, we denote $N(k)$ as the set of neighbors of v_k. We use $\pi(k)$, the k-th component of the vector $\boldsymbol{\pi}$, to denote the stationary distribution of v_k where $1 \leq k \leq |V|$. We define a preference vector $\boldsymbol{p_R} = \{p_1, ..., p_{|V|}\}$ such that the probabilities sum to 1, and p_k denotes the relative importance attached to v_k. p_k is set to $1/|R|$ for $v_k \in R$, otherwise 0. Let $\boldsymbol{A}$ be the matrix corresponding to the graph G where $A_{jk} = 1/|N(k)|$ and $A_{jk} = 0$ otherwise.

For a given $\boldsymbol{p_R}$, we can obtain the personalized PageRank equation (Jeh and Widom, 2003):

$$\boldsymbol{\pi} = (1 - \beta)\boldsymbol{A\pi} + \beta\boldsymbol{p_R} \quad (1)$$

where $\beta \in [0, 1]$ determines how often surfers jump back to the nodes in R. We set $\beta = 0.3$ in our experiment. The solution π to Equation 1 is a steady-state importance distribution induced by $\boldsymbol{p_R}$. Based on a theorem of Markov Theory, a solution π with $\sum_{k=1}^{|V|} \pi(k) = 1$ always exists and is unique (Motwani and Raghavan, 1996).

We define relative importance scores based on the personalized ranks described above, *i.e.*, $I(v \mid R) = \pi(v)$ after convergence, and we compute the importance scores for all the nodes in V relative to Q and F respectively.

A query mention in a sentence is more likely to be involved in multiple relations while a filler is usually associated with only one slot type. Therefore we combine two relative importance scores by assigning a higher priority to $I(v \mid F)$ as follows.

$$\mathcal{I}(v \mid \{Q, F\}) = I(v \mid F) + I(v \mid F) \cdot I(v \mid Q) \quad (2)$$

We discard a trigger candidate if it is (or part of) an entity which can only act as a query or a slot filler. We assume a trigger can only be a noun, verb, adjective, adverb or preposition. In addition, verbs, nouns and adjectives are more informative to be triggers. Thus, we remove any trigger candidate v if it has a higher $\mathcal{I}(v \mid \{Q, F\})$ than the first top-ranked verb/noun/adjective trigger candidate.

For example, we rank the candidate triggers based on the query and slot filler pair ("*Dominick Dunne*", "*Ellen Griffin Dunne*") as shown in Figure 2.

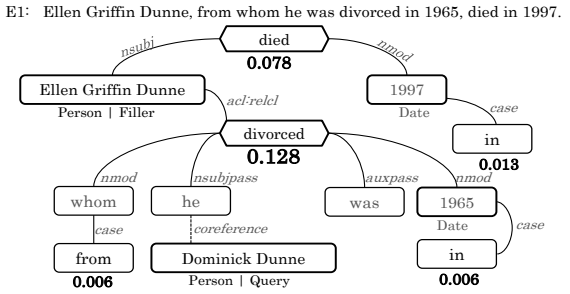


Figure 2: Importance scores of trigger candidates relative to query and filler in E1.

3.2 Trigger Candidate Selection

Given Q and F, we can obtain a relative importance score $\mathcal{I}(v \mid \{Q, F\})$ for each candidate trigger node v in V as shown in Section 3.1. We denote the set of trigger candidates as $T = \{t_1, \cdots, t_n\}$ where $n \leq |V|$.

Since a relation can be indicated by a single trigger word, a trigger phrase or even multiple non-adjacent trigger words, it is difficult to set a single threshold even for one slot type. Instead, we aim to automatically classify top ranked candidates into one group (*i.e.*, a trigger set) so that they all have similar higher scores compared to other candidates.

Therefore, we define this problem as a clustering task. We mainly consider clustering algorithms which do not require pre-specified number of clusters.

We apply the affinity propagation approach to take as input a collection of real-valued similarity scores between pairs of candidate triggers. Real-valued ***messages*** are exchanged between candidate triggers until a high-quality set of ***exemplars*** (centers of clusters), and corresponding clusters gradually emerges (Frey and Dueck, 2007).

There are two kinds of messages exchanged between candidate triggers: one is called ***responsibility*** $\gamma(i, j)$, sent from t_i to a candidate exemplar t_j; the other is ***availability*** $\alpha(i, j)$, sent from the candidate exemplar t_j to t_i.

The calculation of each procedure iterates until convergence. To begin with, the availabilities are initialized to zero: $\alpha(i, j) = 0$. Then the responsibilities are computed using the following rule:

$$\gamma(i, j) \leftarrow s(i, j) - \max_{j's.t.j' \neq j} \{\alpha(i, j') + s(i, j')\} \quad (3)$$

where the similarity score $s(i, j)$ indicates how well t_j is suited to be the exemplar for t_i. Whereas the above responsibility update lets all candidate exemplars compete for the ownership of a trigger candidate t_i, the following availability update gathers evidence from trigger candidates as to whether each candidate exemplar would make a good exemplar:

$$\alpha(i, j) \leftarrow \min \Big\{0, \gamma(j, j) + \sum_{i's.t.i' \notin \{i,j\}} \max\{0, \gamma(i', j)\}\Big\} \quad (4)$$

Given T, we can generate an $n \times n$ affinity matrix $\boldsymbol{M}$ which serves as the input of the affinity propagation. M_{ij} represents the negative squared difference in relative importance score between t_i and t_j (Equation 5).

$$M_{ij} = -(\mathcal{I}(i \mid \{Q, F\}) - \mathcal{I}(j \mid \{Q, F\}))^2 \quad (5)$$

We compute the average importance score for all the clusters after convergence and keep the one with the highest average score as the trigger set. For example, given the query and slot filler pair in Figure 3, we obtain trigger candidates $T = \{died, divorced, from, in, in\}$ and their corresponding relative importance scores. After the above clustering, we obtain three clusters and choose the cluster $\{divorced\}$ with the highest average relative importance score (0.128) as the trigger set.

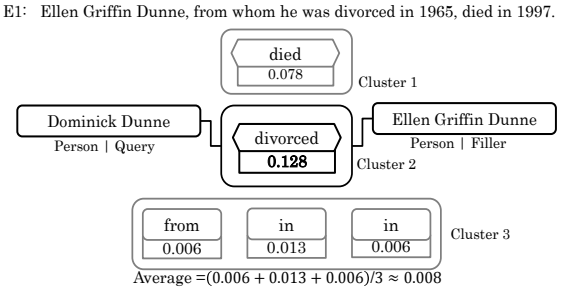


Figure 3: Trigger candidate filtering for E1.

4 Slot Type Labeling

In this section, we will introduce how to label the slot type for an identified relation tuple (Q, T, F). The simplest solution is to match T against existing trigger gazetteers for certain types of slots. For

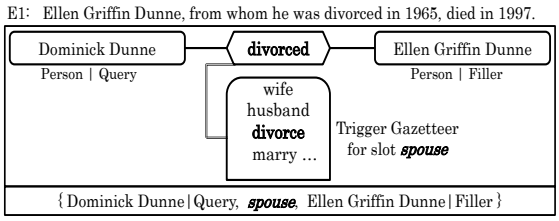


Figure 4: Example of slot type labeling.

example, Figure 4 shows how we label the relation as a *spouse* slot type.

In fact, some trigger gazetteers have already been constructed by previous work such as (Yu et al., 2015). However, manual construction of these triggers heavily rely upon labeled training data and high-quality patterns, which would be unavailable for a new language or a new slot type.

Inspired by the trigger-based event extraction work (Bronstein et al., 2015), we propose to extract trigger seeds from the slot filling annotation guideline [1] and then expand them by paraphrasing techniques. For each slot type we manually select two trigger seeds from the guideline and then use the Paraphrase Database (PPDB) (Ganitkevitch et al., 2013; Pavlick et al., 2015) to expand these seeds. Specifically, we select top-20 lexical paraphrases based on similarity scores as our new triggers for each slot type. Some examples are shown in Table 2.

Seeds	Slot Types	Expanded Triggers
assassinate	*death*	kill, die, slay, murder
graduate	*schools*	PhD, supervisor, diploma
sister	*siblings*	twin, half-brother, sibling
marriage	*spouse*	married, spouse, matrimony

Table 2: PPDB-based trigger expansion examples.

5 Filler Validation

After we label each relation tuple, we perform the following validation steps to filter noise and remove redundancy. For many slot types, there are some specific constraints on entity types of slot fillers defined in the task specification. For example, *employee_or_member_of* fillers should be either organizations or geopolitical entities, while family slots (*e.g.*, *spouse* and *children*) expect person entities. We apply these constraints to further validate all relation tuples.

[1] http://www.nist.gov/tac/2015/KBP/ColdStart/guidelines/TAC_KBP_2015_Slot_Descriptions_V1.0.pdf

Moreover, ***single-value*** slots can only have a single filler (*e.g.*, *date_of_birth*), while ***list-value*** slots can take multiple fillers (*e.g.*, *cities_of_residence*). However, we might extract conflicting relation tuples from multiple sentences and sources. For each relation tuple, it can also be extracted from multiple sentences, and thus it may receive multiple relative importance scores. We aim to keep the most reliable relation tuple for a single-value slot.

For a single-value slot, suppose we have a collection of relation tuples R which share the same query. Given $r \in R$ with a set of relative importance scores $\mathcal{I} = \{i_1, i_2, \cdots, i_n\}$, we can regard the average score of $\mathcal{I}$ as the credibility score of r. The reason is that the higher the relative importance score, the more likely the tuple is to be correct. In our experiments, we use the weighted arithmetic mean as follows so that higher scores can contribute more to the final average:

$$\bar{i} = \frac{\sum_{k=1}^{n} w_k \cdot i_k}{\sum_{k=1}^{n} w_k} \quad (6)$$

where w_k denotes the non-negative weight of i_k. When we regard the weight w_k equal to the score i_k, Equation 6 can be simplified as:

$$\bar{i} = \frac{\sum_{k=1}^{n} w_k^2}{\sum_{k=1}^{n} w_k} \quad (7)$$

We calculate the weighted mean $\bar{i}$ for each $r \in R$ and keep the relation tuple with the highest $\bar{i}$.

6 Experiments

6.1 Data and Scoring Metric

In order to evaluate the quality of our proposed framework and its portability to a new language, we use TAC-KBP2013 English Slot Filling (ESF), TAC-KBP 2015 English Cold Start Slot Filling (CSSF) and TAC-KBP2015 Chinese Slot Filling (CSF) data sets for which we can compare with the ground truth and state-of-the-art results reported in previous work. The source collection includes news documents, web blogs and discussion forum posts. In ESF there are 50 person queries and on average 20 relevant documents per query; while in CSF there are 51 person queries, and on average 5 relevant documents per query.

Slot Type	Our Approach	Roth'13	Angeli'14
siblings	**62.9**	48.0	40
other_family	**42.4**	11.8	0
spouse	58.7	40.0	**66**
children	**66.7**	27.3	27
parents	43.1	**47.8**	39
schools_attended	**81.4**	30.2	60
date_of_birth	87.0	60.0	**92**
date_of_death	**73.2**	3.2	48
state_of_birth	**55.6**	30.8	17
state_of_death	**88.2**	53.3	0
city_of_birth	**70.0**	64.0	25
city_of_death	72.7	**73.7**	30
country_of_birth	**75.0**	0.0	0
country_of_death	**70.0**	46.2	18
states_of_residence	**57.1**	25.6	12
cities_of_res.	**61.4**	38.8	38
countries_of_res.	**45.7**	20.0	41
employee_of	**43.8**	18.5	38
Overall	**57.4**	32.3	–

Table 3: English Slot Filling F_1 (%) (KBP2013 SF data set).

Slot Type	Our Approach	Angeli'15
siblings	**48.0**	26.1
other_family	0.0	**33.3**
spouse	14.3	**15.4**
children	**72.8**	0.0
parents	**25.0**	14.3
schools_attended	**63.6**	42.1
date_of_birth	0.0	**80.0**
date_of_death	**44.0**	0.0
state_of_birth	0.0	**33.3**
state_of_death	0.0	**15.4**
city_of_birth	0.0	**85.7**
city_of_death	0.0	0.0
country_of_birth	0.0	**66.7**
country_of_death	**100.0**	0.0
states_of_residence	0.0	0.0
cities_of_res.	0.0	**50.0**
countries_of_res.	0.0	0.0
employee_of	**60.0**	26.7
Overall	**39.2**	27.6

Table 4: English Cold Start Slot Filling F_1 (%) (KBP2015 CSSF data set).

We only test our method on 18 trigger-driven person slot types shown in Table 3. Some other slot types (*e.g.*, *age*, *origin*, *religion* and *title*) do not rely on lexical triggers in most cases; instead the query mention and the filler are usually adjacent or seperated by a comma. In addition, we do not deal with the two remaining trigger-driven person slot types (*i.e.*, *cause_of_death* and *charges*) since these slots often expect other types of concepts (*e.g.*, a disease or a crime phrase).

We use the official TAC-KBP slot filling evaluation scoring metrics: *Precision* (P), *Recall* (R) and *F-score* (F_1) (Ji et al., 2010) to evaluate our results.

6.2 English Slot Filling

We apply Stanford CoreNLP (Manning et al., 2014) for English part-of-speech (POS) tagging, name tagging, time expression extraction, dependency parsing and coreference resolution. In Table 3 we compare our approach with two state-of-the-art English slot filling methods: a distant supervision method (Roth et al., 2013) and a hybrid method that combines distant and partial supervision (Angeli et al., 2014b). Our method outperforms both methods dramatically. KBP2015 English cold start slot filling is a task which combines entity mention extraction and slot filling (Surdeanu and Ji, 2014). Based on the released evaluation queries from KBP2015 Cold Start Slot Filling, our approach achieves 39.2% overall F-score on 18 person trigger-driven slot types, which is significantly better than state-of-the-art (Angeli et al., 2015) on the same set of news documents (Table 4).

Compared to the previous work, our method discards a trigger-driven relation tuple if it is not supported by triggers. For example, "*Poland*" is mistakenly extracted as the country of residence of "*Mandelbrot*" by distant supervision (Roth et al., 2013) from the following sentence:

A professor emeritus at Yale University, ***Mandelbrot*** *was born in* ***Poland*** *but as a child moved with his family to* ***France*** *where he was educated.*

maybe because the relation tuple (*Mandelbrot*, *live_in*, *Poland*) indeed exists in external knowledge bases. Given the same entity pair, our method identifies "*born*" as the trigger word and labels the slot type as *country_of_birth*.

When there are several triggers indicating different slot types in a sentence, our approach performs better in associating each trigger with the filler it dominates by analyzing the whole dependency tree. For example, given a sentence:

Haig *is survived by his wife of 60 years, Patricia; his children Alexander, Brian and* ***Barbara****; eight grandchildren; and his brother, the Rev. Francis R. Haig.*

(*Haig*, *sibling*, *Barbara*) is the only relation tuple extracted from the above sentence by the previous method. Given the entity pair (*Haig*, *Barbara*), the relative importance score of "*children*" (0.1) is higher than the score of "*brother*" (0.003),

and "*children*" is kept as the only trigger candidate after clustering. Therefore, we extract the tuple (*Haig*, *children*, *Barbara*) instead. In addition, we successfully identify the missing fillers for other slot types: *spouse* (*Patricia*), *children* (*Alexander, Brian and Barbara*) and *siblings* (*Francis R. Haig*) by identifying their corresponding triggers.

In addition, flat relation representations fail to extract the correct relation (*i.e.*, *alternate_names*) between "*Dandy Don*" and "*Meredith*" since "*brother*" is close to both of them in the following sentence:

In high school and at Southern Methodist University, where, already known as ***Dandy Don*** *(a nickname bestowed on him by his brother) ,* ***Meredith*** *became an all-American.*

6.3 Adapting to New Slot Types

Our framework can also be easily adapted to new slot types. We evaluate it on three new person list-value slot types: *friends*, *colleagues* and *collaborators*.

We use "*friend*" as the slot-specific trigger for the slot *friends* and "*colleague*" for the slot *colleagues*. "*collaborate*", "*cooperate*" and "*partner*" are used to type the slot *collaborators*.

We manually annotate ground truth for evaluation. It is difficult to find all the correct fillers for a given query from millions of documents. Therefore, we only calculate precision. Experiments show we can achieve 56.3% for *friends*, 100% for *colleagues* and 60% for *collaborators* (examples shown in Table 5).

6.4 Impact of Trigger Mining

In Section 3.2, we keep top-ranked trigger candidates based on clustering rather than threshold tuning. We explore a range of thresholds for comparison, as shown in Figure 5. Our approach achieves 57.4% F-score, which is comparable to the highest F-score 58.1% obtained by threshold tuning.

We also measure the impact of the size of the trigger gazetteer. We already outperform state-of-the-art by using PPDB to expand triggers mined from guidelines as shown in Table 6. As the size of the trigger gazetteer increases, our method (marked with a $\star$) achieves better performance.

6.5 Chinese Slot Filling

As long as we have the following resources: (1) a POS tagger, (2) a name tagger, (3) a dependency parser and (4) slot-specific trigger gazetteers, we can apply the framework to a new language. Coreference resolution is optional.

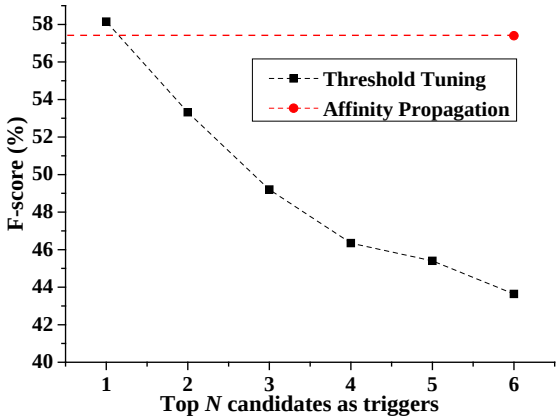


Figure 5: The effect of the number of trigger candidates on ESF.

Method	Size	F_1 (%)
State-of-the-art (Roth et al., 2013)	–	32.3
Guideline seeds*	20	27.3
Guideline seeds + PPDB expansion*	220	**38.9**
Manually Constructed Trigger Gazetteers*	7,463	**57.4**

Table 6: The effect of trigger gazetteers on ESF (size: the number of triggers).

We demonstrate the portability of our framework to Chinese since all the resources mentioned above are available. We apply Stanford CoreNLP (Manning et al., 2014) for Chinese POS tagging, name tagging (Wang et al., 2013) and dependency parsing (Levy and Manning, 2003). To explore the impact of the quality of annotation resources, we also use a Chinese language analysis tool: Language Technology Platform (LTP) (Che et al., 2010). We use the full set of Chinese trigger gazetteers published by Yu et al. (2015). Experimental results (Table 7) demonstrate that our approach can serve as a new and promising benchmark. As far as we know, there are no results available for comparison.

However, the performance of Chinese SF is heavily influenced by the relatively low performance of name tagging since our method returns an empty result if it fails to find any query metnion. About 20% and 16% queries cannot be recognized by CoreNLP and LTP respectively. One reason is that many Chinese names are also common words. For example, a buddhist monk's name "觉醒"(*wake*) is identified as a verb rather than a person entity.

Evidence Sentence	Slot Type	Query	Extracted Fillers
Many of ***his*** subjects were friends from his previous life , such as ***Elizabeth Taylor*** and ***Gloria Vanderbilt*** .	*friends*	Dominick Dunne	Gloria Vanderbilt; Elizabeth Taylor
Toby Keith hit an emotional note with a performance of "Cryin' For Me (Wayman's Song)," dedicated to ***his*** late friend, jazz artist and former basketball star ***Wayman Tisdale***, who died last May.	*friends*	Wayman Tisdale	Toby Keith
"I think all of her writing came from ***her*** heart," ***Michael Glaser***, a longtime colleague at St. Mary's and former Maryland poet laureate, said last week.	*colleagues*	Lucille Clifton	Michael Glaser
Cunningham has collaborated on two books: "Changes: Notes on Choreography," with Frances Starr, and "The Dancer and the Dance," with ***Jacqueline Lesschaeve***.	*collaborators*	Merce Cunningham	Jacqueline Lesschaeve

Table 5: Examples for new slot types.

A dependency parser is indispensable to produce reliable rankings of trigger candidates. Unfortunately, a high-quality parser for a new language is often not available because of language-specific features. For example, in Chinese a single sentence about a person's biography often contains more than five co-ordinated clauses, each of which includes a trigger. Therefore a dependency parser adapted from English often mistakenly identifies one of the triggers as a main predicate of the sentence.

In addition, Chinese is a very concise language. For example, a "[Person Name][Organization Suffix]"structure can indicate various different types of relations between the person name and the organization: "杨明牙医诊所"(*Yang Ming Clinic*) indicates ownership, "邵逸夫图书馆"(*Shao Yifu Library*) indicates sponsorship, "丰子恺研究中心"(*Feng Zikai Research Center*) indicates research theme, and "罗京治丧委员会"(*Luojing Commemoration Committee*) indicates commemoration. None of them includes an explicit trigger nor indicates employment relation. It requires more fine-grained dependency relation types to distinguish them.

Finally, compared to English, Chinese tends to have more variants for some types of triggers (e.g., there are at least 31 different titles for "*wife*"in Chinese). Some of them are implicit and require shallow inference. For example, "投奔"(*to seek shelter or asylum*) indicates a residence relation in most cases.

Slot Type	CoreNLP-based	LTP-based
siblings	40.0	**57.1**
other_family	**40.0**	0.0
spouse	40.0	**48.0**
children	19.0	**21.4**
parents	0.0	**25.0**
schools_attended	11.1	**17.1**
date_of_birth	**42.4**	0.0
date_of_death	**48.5**	0.0
state_of_birth	38.1	**52.2**
state_of_death	55.6	**70.0**
city_of_birth	**28.6**	26.7
city_of_death	33.3	**42.9**
country_of_birth	11.8	11.8
country_of_death	0.0	0.0
states_of_residence	**30.8**	29.6
cities_of_residence	27.3	**34.8**
country_of_residence	**6.5**	0.0
employee_of	31.0	**31.2**
Overall	**29.6**	28.3

Table 7: Chinese Slot Filling F_1 (%) (KBP2015 CSF data set).

7 Related Work

Besides the methods based on distant supervision (*e.g.*, (Surdeanu et al., 2010; Roth et al., 2013; Angeli et al., 2014b)) discussed in Section 6.2, pattern-based methods have also been proven to be effective in SF in the past years (Sun et al., 2011; Li et al., 2012; Yu et al., 2013). Dependency-based patterns achieve better performance since they can capture long-distance relations. Most of these approaches assume that a relation exists between Q and F if there is a dependency path connecting Q and F and all the words on the path are equally regarded as trigger candidates. We explore the complete graph structure of a sentence rather than chains/subgraphs as in previous work. Our previous research focused on identifying the relation between F and T by extracting filler candidates from the identified scope of a trigger (*e.g.*, (Yu et al., 2015)). We found that each slot-specific trigger has its own scope, and corresponding fillers seldom appear outside its scope. We did not compare with results from this previous approach which did not consider redundancy removal required in the official evaluations.

Soderland et al. (2013) built their SF system based on Open Information Extraction (IE) technology. Our method achieves much higher recall since dependency trees can capture the relations among query, slot filler and trigger in more complicated long sentences. In addition, our triggers are automatically labeled so that we do not need to design manual rules to classify relation phrases as in Open IE.

8 Conclusions and Future Work

In this paper, we demonstrate the importance of deep mining of dependency structures for slot filling. Our approach outperforms state-of-the-art and can be rapidly portable to a new language or a new slot type, as long as there exists capabilities of name tagging, POS tagging, dependency parsing and trigger gazetteers.

In the future we aim to label slot types based on contextual information as well as sentence structures instead of trigger gazetteers only. There are two primary reasons. First, a trigger can serve for multiple slot types. For example, slot *children* and its inverse slot *parents* share a subset of triggers. Second, a trigger word can have multiple different meanings. For example, a *sibling* trigger word "*sister*" can also represent a female member of a religious community. We attempt to combine multi-prototype approaches (*e.g.*, (Reisinger and Mooney, 2010)) to better disambiguate senses of trigger words.

Besides considering the cross-sentence conflicts, we also want to investigate the within-sentence conflicts caused by the competition of triggers. A trigger identified by our approach is the most important node in the dependency tree relative to the given entity pair. However, this trigger might be more important to another entity pair, which shares the same filler, in the same sentence. A promising solution is to rank all the entities in the sentence based on their importance relative to the identified trigger and the filler candidate.

Acknowledgement

We would like to thank Chris Callison-Burch for providing English and Chinese paraphrase resources. This work was supported by the DARPA LORELEI Program No. HR0011-15-C-0115, DARPA DEFT Program No. FA8750-13-2-0041, ARL NS-CTA No. W911NF-09-2-0053, NSF CAREER Award IIS-1523198. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

G. Angeli, S. Gupta, M. Jose, C. Manning, C. Ré, J. Tibshirani, J. Wu, S. Wu, and C. Zhang. 2014a. Stanford's 2014 slot filling systems. In *Proc. Text Analysis Conference (TAC 2014)*.

G. Angeli, J. Tibshirani, J. Wu, and C. Manning. 2014b. Combining distant and partial supervision for relation extraction. In *Proc. Empirical Methods on Natural Language Processing (EMNLP 2014)*.

G. Angeli, V. Zhong, D. Chen, J. Bauer, A. Chang, V. Spitkovsky, and C. Manning. 2015. Bootstrapped self training for knowledge base population. In *Proc. Text Analysis Conference (TAC 2015)*.

O. Bronstein, I. Dagan, Q. Li, H. Ji, and A. Frank. 2015. Seed-based event trigger labeling: How far can event descriptions get us? In *Proc. Association for Computational Linguistics (ACL 2015)*.

W. Che, Z. Li, and T. Liu. 2010. Ltp: A chinese language technology platform. In *Proc. Computational Linguistics (COLING 2010)*.

B. Frey and D. Dueck. 2007. Clustering by passing messages between data points. *science*.

J. Ganitkevitch, B. Van Durme, and C. Callison-Burch. 2013. PPDB: The paraphrase database. In *Proc. North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2013)*.

Y. Hong, X. Wang, Y. Chen, J. Wang, T. Zhang, J. Zheng, D. Yu, and Q. Li. 2014. Rpi blender tac-kbp2014 knowledge base population system. In *Proc. Text Analysis Conference (TAC 2014)*.

G. Jeh and J. Widom. 2003. Scaling personalized web search. In *Proc. World Wide Web (WWW 2003)*.

H. Ji, R. Grishman, H. Dang, K. Griffitt, and Joe Ellis. 2010. An overview of the tac2010 knowledge base population track. In *Proc. Text Analysis Conference (TAC 2010)*.

H. Ji, R. Grishman, and H. Dang. 2011. An overview of the tac2011 knowledge base population track. In *Proc. Text Analysis Conference (TAC 2011)*.

R. Levy and C. Manning. 2003. Is it harder to parse chinese, or the chinese treebank? In *Proc. Association for Computational Linguistics (ACL 2003)*.

Y. Li, S. Chen, Z. Zhou, J. Yin, H. Luo, L. Hong, W. Xu, G. Chen, and J. Guo. 2012. Pris at tac2012 kbp track. In *Proc. Text Analysis Conference (TAC 2012)*.

C. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. Bethard, and D. McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proc. Association for Computational Linguistics (ACL 2014)*.

P. McNamee and H. Dang. 2009. Overview of the tac 2009 knowledge base population track. In *Proc. Text Analysis Conference (TAC 2009)*.

R. Motwani and P. Raghavan. 1996. Randomized algorithms. *ACM Computing Surveys (CSUR)*.

L. Page, S. Brin, R. Motwani, and T. Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

E. Pavlick, P. Rastogi, J. Ganitkevitch, and C. Van Durme, B.and Callison-Burch. 2015. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proc. Association for Computational Linguistics (ACL 2015)*.

J. Reisinger and R. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Proc. North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2010)*.

B. Roth, T. Barth, M. Wiegand, M. Singh, and D. Klakow. 2013. Effective slot filling based on shallow distant supervision methods. In *Proc. Text Analysis Conference (TAC 2013)*.

S. Soderland, J. Gilmer, R. Bart, O. Etzioni, and D. Weld. 2013. Open ie to kbp relations in 3 hours. In *Proc. Text Analysis Conference (TAC 2013)*.

A. Sun, R. Grishman, B. Min, and W. Xu. 2011. Nyu 2011 system for kbp slot filling. In *Proc. Text Analysis Conference (TAC 2011)*.

M. Surdeanu and H. Ji. 2014. Overview of the english slot filling track at the tac2014 knowledge base population evaluation. In *Proc. Text Analysis Conference (TAC 2014)*.

M. Surdeanu, D. McClosky, J. Tibshirani, J. Bauer, A. Chang, V. Spitkovsky, and C. Manning. 2010. A simple distant supervision approach for the tac-kbp slot filling task. In *Proc. Text Analysis Conference (TAC 2010)*.

M. Wang, W. Che, and C. Manning. 2013. Joint word alignment and bilingual named entity recognition using dual decomposition. In *Proc. Association for Computational Linguistics (ACL 2013)*.

S. White and P. Smyth. 2003. Algorithms for estimating relative importance in networks. In *Proc. Knowledge discovery and data mining (KDD 2003)*.

D. Yu, H. Li, T. Cassidy, Q. Li, H. Huang, Z. Chen, H. Ji, Y. Zhang, and D. Roth. 2013. Rpi-blender tac-kbp2013 knowledge base population system. In *Proc. Text Analysis Conference (TAC 2013)*.

D. Yu, H. Ji, S. Li, and C. Lin. 2015. Why read if you can scan: Scoping strategy for biographical fact extraction. In *Proc. North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL-HLT 2015)*.

A Multi-media Approach to Cross-lingual Entity Knowledge Transfer

Di Lu[1], **Xiaoman Pan**[1], **Nima Pourdamghani**[2],
Shih-Fu Chang[3], **Heng Ji**[1], **Kevin Knight**[2]
[1] Computer Science Department, Rensselaer Polytechnic Institute
{lud2,panx2,jih}@rpi.edu
[2] Information Sciences Institute, University of Southern California
{damghani,knight}@isi.edu
[3] Electrical Engineering Department, Columbia University
sfchang@ee.columbia.edu

Abstract

When a large-scale incident or disaster occurs, there is often a great demand for rapidly developing a system to extract detailed and new information from low-resource languages (LLs). We propose a novel approach to discover comparable documents in high-resource languages (HLs), and project Entity Discovery and Linking results from HLs documents back to LLs. We leverage a wide variety of language-independent forms from multiple data modalities, including image processing (image-to-image retrieval, visual similarity and face recognition) and sound matching. We also propose novel methods to learn entity priors from a large-scale HL corpus and knowledge base. Using Hausa and Chinese as the LLs and English as the HL, experiments show that our approach achieves 36.1% higher Hausa name tagging F-score over a costly supervised model, and 9.4% higher Chinese-to-English Entity Linking accuracy over state-of-the-art.

1 Introduction

In many situations such as disease outbreaks and natural calamities, we often need to develop an Information Extraction (IE) component (e.g., a name tagger) within a very limited time to extract information from low-resource languages (LLs) (e.g., locations where Ebola outbreaks from Hausa documents). The main challenge lies in the lack of labeled data and linguistic processing tools in these languages. A potential solution is to extract and project knowledge from high-resource languages (HLs) to LLs.

A large amount of non-parallel, domain-rich, topically-related comparable corpora naturally exist across LLs and HLs for breaking incidents, such as coordinated news streams (Wang et al., 2007) and code-switching social media (Voss et al., 2014; Barman et al., 2014). However, without effective Machine Translation techniques, even just identifying such data in HLs is not a trivial task. Fortunately many of such comparable documents are presented in multiple data modalities (text, image and video), because press releases with multimedia elements generate up to 77% more views than text-only releases (Newswire, 2011). In fact, they often contain the same or similar images and videos, which are language-independent.

In this paper we propose to use images as a hub to automatically discover comparable corpora. Then we will apply Entity Discovery and Linking (EDL) techniques in HLs to extract entity knowledge, and project results back to LLs by leveraging multi-source multi-media techniques. In the following we will elaborate motivations and detailed methods for two most important EDL components: name tagging and Cross-lingual Entity Linking (CLEL). For CLEL we choose Chinese as the LL and English as HL because Chinese-to-English is one of the few language pairs for which we have ground-truth annotations from official shared tasks (e.g., TAC-KBP (Ji et al., 2015)). Since Chinese name tagging is a well-studied problem, we choose Hausa instead of Chinese as the LL for name tagging experiment, because we can use the ground truth from the DARPA LORELEI program[1] for evaluation.

Entity and Prior Transfer for Name Tagging: In the first case study, we attempt to use HL extraction results directly to validate and correct names

[1] http://www.darpa.mil/program/low-resource-languages-for-emergent-incidents

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 54–65,
Berlin, Germany, August 7-12, 2016.

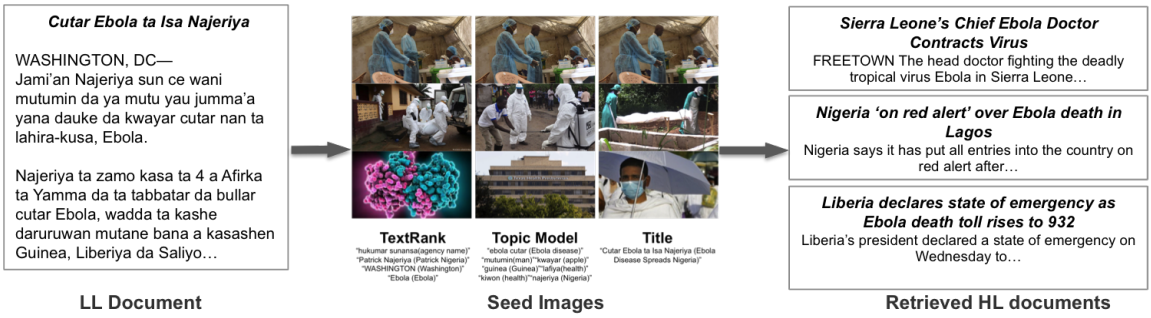


Figure 1: Image Anchored Comparable Corpora Retrieval.

extracted from LLs. For example, in the Hausa document in Figure 1, it would be challenging to identify the location name "*Najeriya*" directly from the Hausa document because it's different from its English counterpart. But since its translation "*Nigeria*" appears in the topically-related English document, we can use it to infer and validate its name boundary.

Even if topically-related documents don't exist in an HL, similar scenarios (e.g., disease outbreaks) and similar activities of the same entity (e.g., meetings among politicians) often repeat over time. Moreover, by running a high-performing HL name tagger on a large amount of documents, we can obtain ***entity prior*** knowledge which shows the probability of a related name appearing in the same context. For example, if we already know that "*Nigeria*", "*Borno*", "*Goodluck Jonathan*", "*Boko Haram*" are likely to appear, then we could also expect "*Mouhammed Ali Ndume*" and "*Mohammed Adoke*" might be mentioned because they were both important politicians appointed by Goodluck Jonathan to consider opening talks with Boko Haram. Or more generally if we know the LL document is about politics in China in 1990s, we could estimate that famous politicians during that time such as "*Deng Xiaoping*" are likely to appear in the document.

Next we will project these names extracted from HL documents directly to LL documents to identify and verify names. In addition to textual evidence, we check visual similarity to match an HL name with its equivalent in LL. And we apply face recognition techniques to verify person names by image search. This idea matches human knowledge acquisition procedure as well. For example, when a child is watching a cartoon and shifting between versions in two languages, s/he can easily infer translation pairs for the same concept whose images appear frequently (e.g., "宝宝 (*baby*)" and "螃蟹 (*crab*)" in "*Dora Exploration*", "海盗 (*pirate*)" in "*the Garden Guardians*", and "亨利 (*Henry*)" in "*Thomas Train*"), as illustrated in Figure 2.

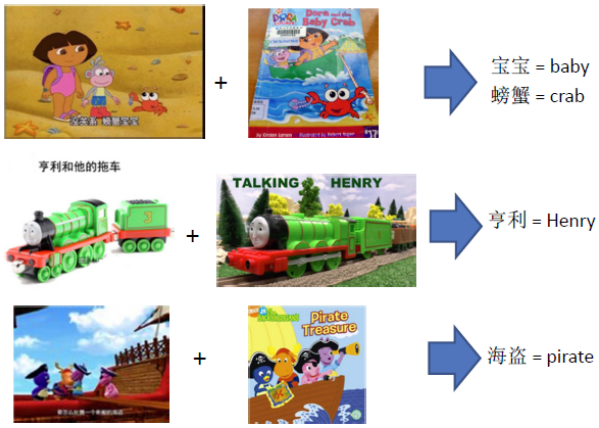


Figure 2: Examples of Cartoons in Chinese (left) and English (right).

Representation and Structured Knowledge Transfer for Entity Linking: Besides data sparsity, another challenge for low-resource language IE lies in the lack of knowledge resources. For example, there are advanced knowledge representation parsing tools available (e.g., Abstract Meaning Representation (AMR) (Banarescu et al., 2013)) and large-scale knowledge bases for English Entity Linking, but not for other languages, including some medium-resource ones such as Chinese. For example, the following documents are both about the event of Pistorius killing his girl friend Reeva:

- **LL document:** 南非残疾运动员皮斯托瑞斯被指控杀害女友瑞娃于其茨瓦内的家中。皮斯托瑞斯是南非著名的残疾人田径选手，有"刀锋战士"之称。...(*The disabled South African sportsman Oscar Pis-*

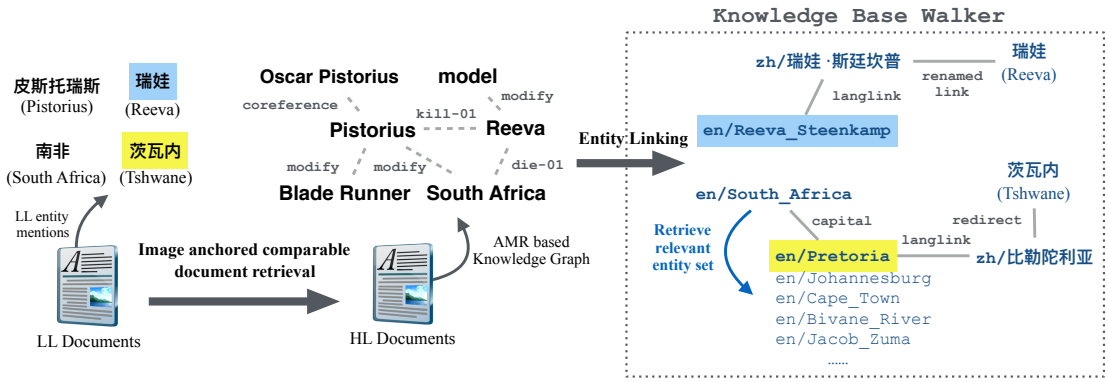


Figure 3: Cross-lingual Knowledge Transfer for Entity Linking.

torius was charged to killing his girl friend Reeva at his home in Tshwane. Pistorius is a famous runner in South Africa, also named as "Blade Runner"...)

- **HL document:** *In the early morning of Thursday, 14 February 2013, "Blade Runner" Oscar Pistorius shot and killed South African model Reeva Steenkamp...*

From the LL documents we may only be able to construct co-occurrence based knowledge graph and thus it's difficult to link rare entity mentions such as "瑞娃 (*Reeva*)" and "茨瓦内 (*Tshwane*)" to an English knowledge base (KB). But if we apply an HL (e.g., English) entity linker, we could construct much richer knowledge graphs from HL documents using deep knowledge representations such as AMR, as shown in Figure 3, and link all entity mentions to the KB accurately. Moreover, if we start to *walk* through the KB, we can easily reach from English related entities to the entities mentioned in LL documents. For example, we can walk from "*South Africa*" to its capital "*Pretoria*" in the KB, which is linked to its LL form "比勒陀利亚" through a language link and then is re-directed to "茨瓦内" mentioned in the LL document through a redirect link. Therefore we can infer that "茨瓦内" should be linked to "*Pretoria*" in the KB.

Compared to most previous cross-lingual projection methods, our approach does not require domain-specific parallel corpora or lexicons, or in fact, any parallel data at all. It also doesn't require any labeled data in LLs. Using Hausa and Chinese as the LLs and English as HL for case study, experiments demonstrate that our approach can achieve 36.1% higher Hausa name tagging over a costly supervised model trained from 337 documents, and 9.4% higher Chinese-to-English Entity Linking accuracy over a state-of-the-art system.

2 Approach Overview

Figure 4 illustrates the overall framework. It consists of two steps: (1) Apply language-independent key phrase extraction methods on each LL document, then use key phrases as a query to retrieve seed images, and then use the seed images to retrieve matching images, and retrieve HL documents containing these images (Section 3); (2) Extract knowledge from HL documents, and design knowledge transfer methods to refine LL extraction results.

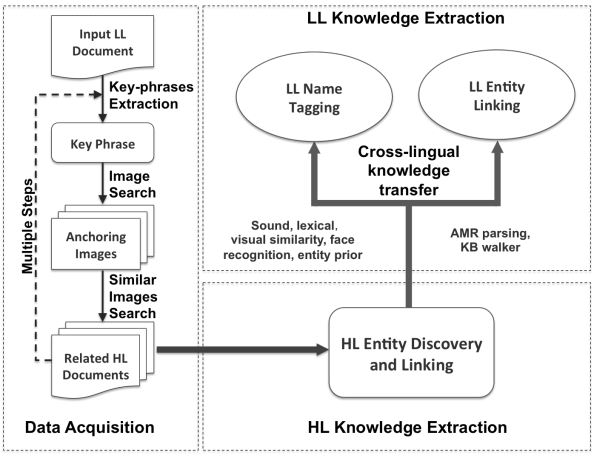


Figure 4: Overall Framework.

We will present two case studies on name tagging (Section 5) and cross-lingual entity linking (CLEL) (Section 6) respectively. Our projection approach consists of a series of non-traditional multi-media multi-source methods based on textual and visual similarity, face recognition, as well as entity priors learned from both unstructured data and structured KB.

3 Comparable Corpora Discovery

In this section we will describe the detailed steps of acquiring HL documents for a given LL document via anchoring images. Using a cluster of images as a hub, we attempt to connect topically-related documents in LL and HL. We will walk through each step for the motivating example in Figure 1.

3.1 Key Phrase Extraction

For an LL document (e.g., Figure 1 for the walk-through example), we start by extracting its key phrases using the following three language-independent methods: (1) TextRank (Mihalcea and Tarau, 2004), which is a graph-based ranking model to determine key phrases. (2) Topic modeling based on Latent Dirichlet allocation (LDA) model (Blei et al., 2003), which can generate a small number of key phrases representing the main topics of each document. (3) The title of the document if it's available.

3.2 Seed Image Retrieval

Using the extracted key phrases together as one single query, we apply Google Image Search to retrieve top 15 ranked images as seeds. To reduce the noise introduced by image search, we filter out images smaller than 100×100 pixels because they are unlikely to appear in the main part of web pages. We also filter out an image if its web page contains less than half of the tokens in the query. Figure 1 shows the anchoring images retrieved for the walk-through example.

3.3 HL Document Retrieval

Using each seed image, we apply Google image-to-image search to retrieve more matching images, and then use the TextCat tool (Cavnar et al., 1994) as a language identifier to select HL documents containing these images. It shows three English documents retrieved for the first image in Figure 1.

For related topics, more images may be available in HLs than LLs. To compensate this data sparsity problem, using the HL documents retrieved as a seed set, we repeat the above steps one more time by extracting key phrases from the HL seed set to retrieve more images and gather more HL documents. For example, a Hausa document about "*Arab Spring*" includes protests that happened in Algeria, Bahrain, Iran, Libya, Yemen and Jordan. The HL documents retrieved by LL key phrases and images in the first step missed the detailed information about protests in Iran. However the second step based on key phrases and images from HL successfully retrieved detailed related documents about protests in Iran.

Applying the above multimedia search, we automatically discover domain-rich non-parallel data. Next we will extract facts from HLs and project them to LLs.

4 HL Entity Discovery and Linking

4.1 Name Tagging

After we acquire HL (English in this paper) comparable documents, we apply a state-of-the-art English name tagger (Li et al., 2014) based on structured perceptron to extract names. From the output we filter out uninformative names such as news agencies. If the same name receives multiple types across documents, we use the majority one.

4.2 Entity Linking

We apply a state-of-the-art Abstract Meaning Representation (AMR) parser (Wang et al., 2015a) to generate rich semantic representations. Then we apply an AMR based entity linker (Pan et al., 2015) to link all English entity mentions to the corresponding entities in the English KB. Given a name n_h, this entity linker first constructs a Knowledge Graph $g(n_h)$ with n_h at the hub and leaf nodes obtained from names reachable by AMR graph traversal from n_h. A subset of the leaf nodes are selected as collaborators of n_h. Names connected by AMR conjunction relations are grouped into sets of coherent names. For each name n_h, an initial ranked list of entity candidates $\mathcal{E} = \{e_1, ..., e_M\}$ is generated based on a salience measure (Medelyan and Legg, 2008). Then a Knowledge Graph $g(e_m)$ is generated for each entity candidate e_m in n_h's entity candidate list $\mathcal{E}$. The entity candidates are then re-ranked according to Jaccard Similarity, which computes the similarity between $g(n_h)$ and $g(e_m)$: $J(g(n_h), g(e_m)) = \frac{|g(n_h)\cap g(e_m)|}{|g(n_h)\cup g(e_m)|}$. Finally, the entity candidate with the highest score is selected as the appropriate entity for n_h. Moreover, the Knowledge Graphs of coherent mentions will be merged and linked collectively.

4.3 Entity Prior Acquisition

Given the English entities discovered from the above, we aim to automatically mine related en-

tities to further expand the expected entity set. We use a large English corpus and English knowledge base respectively as follows.

If a name n_h appears frequently in these retrieved English documents,[2] we further mine other related names n'_h which are very likely to appear in the same context as n_h in a large-scale news corpus (we use English Gigaword V5.0 corpus[3] in our experiment). For each pair of names $\langle n'_h, n_h \rangle$, we compute $P(n'_h|n_h)$ based on their co-occurrences in the same sentences. If $P(n'_h|n_h)$ is larger than a threshold,[4] and n'_h is a person name, then we add n'_h into the expected English name set.

Let $\mathcal{E}^0 = \{e_1, ..., e_N\}$ be the set of entities in the KB that all mentions in English documents are linked to. For each $e_i \in \mathcal{E}^0$, we 'walk' one step from it in the KB to retrieve all of its neighbors $\mathcal{N}(e_i)$. We denote the set of neighbor nodes as $\mathcal{E}^1 = \{\mathcal{N}(e_1), ..., \mathcal{N}(e_N)\}$. Then we extend the expected English entity set as $\mathcal{E}^0 \cup \mathcal{E}^1$. Table 1 shows some retrieved neighbors for entity "*Elon Musk*".

Relation	Neighbor
is founder of	SpaceX
is founder of	Tesla Motors
is spouse of	Justine Musk
birth place	Pretoria
alma mater	University of Pennsylvania
parents	Errol Musk
relatives	Kimbal Musk

Table 1: Neighbors of Entity "*Elon Musk*".

5 Knowledge Transfer for Name Tagging

In this section we will present the first case study on name tagging, using English as HL and Hausa as LL.

5.1 Name Projection

After expanding the English expected name set using entity prior, next we will try to carefully select, match and project each expected name (n_h) from English to the one (n_l) in Hausa documents. We scan through every n-gram (n in the order 3, 2, 1) in Hausa documents to see if any of them match an English name based on the following multi-media language-independent low-cost heuristics.

Spelling: If n_h and n_l are identical (e.g., "*Brazil*"), or with an edit distance of one after lower-casing and removing punctuation (e.g., n_h = "*Mogadishu*" and n_l = "*Mugadishu*"), or substring match (n_h = "*Denis Samsonov*" and n_l = "*Samsonov*").

Pronunciation: We check the pronunciations of n_h and n_l based on Soundex (Odell, 1956), Metaphone (Philips, 1990) and NYSIIS (Taft, 1970) algorithms. We consider two codes match if they are exactly the same or one code is a part of the other. If at least two coding systems match between n_h and n_l, we consider they are equivalents.

Visual Similarity: When two names refer to the same entity, they usually share certain visual patterns in their related images. For example, using the textual clues above is not sufficient to find the Hausa equivalent "*Majalisar Dinkin Duniya*" for "*United Nations*", because their pronunciations are quite different. However, Figure 5 shows the images retrieved by "*Majalisar Dinkin Duniya*" and "*United Nations*" are very similar.[5]

We first retrieve top 50 images for each mention using Google image search. Let $\mathcal{I}_h$ and $\mathcal{I}_l$ denote two sets of images retrieved by an n_h and a candidate n_l (e.g., n_h = "*United Nations*" and n_l = "*Majalisar Dinkin Duniya*" in Figure 5), $i_h \in \mathcal{I}_h$ and $i_l \in \mathcal{I}_l$. We apply the Scale-invariant feature transform (SIFT) detector (Lowe, 1999) to count the number of matched key points between two images, $K(i_h, i_l)$, as well as the key points in each image, $P(i_h)$ and $P(i_l)$. SIFT key point is a circular image region with an orientation, which can provide feature description of the object in the image. Key points are maxima/minima of the Difference of Gaussians after the image is convolved with Gaussian filters at different scales. They usually lie in high-contrast regions. Then we define the similarity ($0 \sim 1$) between two phrases as:

$$S(n_h, n_l) = \max_{i_h \in \mathcal{I}_h} \max_{i_l \in \mathcal{I}_l} \frac{K(i_h, i_l)}{\min(P(i_h), P(i_l))} \quad (1)$$

Based on empirical results from a separate small development set, we decide two phrases match if $S(n_h, n_l) > 10\%$. This visual similarity computation method, though seemingly simple, has been

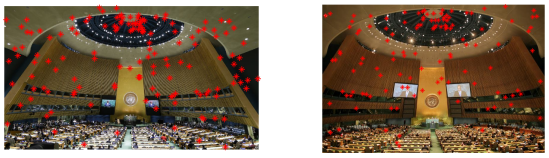
[2] for our experiment we choose those that appear more than 10 times

[3] https://catalog.ldc.upenn.edu/LDC2011T07

[4] 0.02 in our experiment.

[5] Although existing Machine Translation (MT) tools like Google Translate can correctly translate this example phrase from Hausa to English, here we use it as an example to illustrate the low-resource setting when direct MT is not available.

one of the principal techniques in detecting near-duplicate visual content (Ke et al., 2004).



(a) Majalisar Dinkin Duniya (b) United Nations

Figure 5: Matched SIFT Key points.

5.2 Person Name Verification through Face Recognition

For each name candidate, we apply Google image search to retrieve top 10 images (examples in Figure 6). If more than 5 images contain and only contain 1-2 faces, we classify the name as a person. We apply face detection technique based on Haar Feature (Viola and Jones, 2001). This technique is a machine learning based approach where a cascade function is trained from a large amount of positive and negative images. In the future we will try other alternative methods using different feature sets such as Histograms of Oriented Gradients (Dalal and Triggs, 2005).

Figure 6: Face Recognition for Validating Person Name 'Nawaz Shariff'.

6 Knowledge Transfer for Entity Linking

In this section we will present the second case study on Entity Linking, using English as HL and Chinese as LL. We choose this language pair because its ground-truth Entity Linking annotations are available through the TAC-KBP program (Ji et al., 2015).

6.1 Baseline LL Entity Linking

We apply a state-of-the-art language-independent cross-lingual entity linking approach (Wang et al., 2015b) to link names from Chinese to an English KB. For each name n, this entity linker uses the cross-lingual surface form dictionary $\langle f, \{e_1, e_2, ..., e_M\}\rangle$, where $\mathcal{E} = \{e_1, e_2, ..., e_M\}$ is the set of entities with surface form f in the KB according to their properties (e.g., labels, names, aliases), to locate a list of candidate entities $e \in \mathcal{E}$ and compute the importance score by an entropy based approach.

6.2 Representation and Structured Knowledge Transfer

Then for each expected English entity e_h, if there is a cross-lingual link to link it to an LL (Chinese) entry e_l in the KB, we added the title of the LL entry or its redirected/renamed page c_l as its LL translation. In this way we are able to collect a set of pairs of $\langle c_l, e_h\rangle$, where c_l is an expected LL name, and e_h is its corresponding English entity in the KB. For example, in Figure 3, we can collect pairs including "(瑞娃, *Reeva Steenkamp*)", "(瑞娃·斯廷坎普, *Reeva Steenkamp*)", "(茨瓦内, *Pretoria*)" and "(比勒陀利亚, *Pretoria*)". For each mention in an LL document, we then check whether it matches any c_l, if so then use e_h to override the baseline LL Entity Linking result. Table 2 shows some $\langle c_l, e_h\rangle$ pairs with frequency. Our approach not only successfully retrieves translation variants of "*Beijing*" and "*China Central TV*", but also alias and abbreviations.

e_h	e_l	c_l	Freq.
Beijing	北京市	北京(Beijing)	553
		北京市(Beijing City)	227
		燕京(Yanjing)	15
		京师(Jingshi)	3
		北平(Beiping)	2
		首都(Capital)	1
		蓟(Ji)	1
		燕都(Yan capital)	1
China Central TV	中国中央电视台	央视(Central TV)	19
		CCTV	16
		中央电视台(Central TV)	13
		中国央视(China Central TV)	3

Table 2: Representation and Structured Knowledge Transfer for Expected English Entities "*Beijing*" and "*China Central TV*".

7 Experiments

In this section we will evaluate our approach on name tagging and Cross-lingual Entity Linking.

7.1 Data

For name tagging, we randomly select 30 Hausa documents from the DARPA LORELEI program as our test set. It includes 63 person names (PER), 64 organizations (ORG) 225 geo-political entities

(GPE) and locations (LOC). For this test set, in total we retrieved 810 topically-related English documents. We found that 80% names in the ground truth appear at least once in the retrieved English documents, which shows the effectiveness of our image-anchored comparable data discovery method.

For comparison, we trained a supervised Hausa name tagger based on Conditional Random Fields (CRFs) from the remaining 337 labeled documents, using lexical features (character ngrams, adjacent tokens, capitalization, punctuations, numbers and frequency in the training data).

We learn entity priors by running the Stanford name tagger (Manning et al., 2014) on English Gigaword V5.0 corpus.[6] The corpus includes 4.16 billion tokens and 272 million names (8.28 million of which are unique).

For Cross-lingual Entity Linking, we use 30 Chinese documents from the TAC-KBP2015 Chinese-to-English Entity Linking track (Ji et al., 2015) as our test set. It includes 678 persons, 930 geo-political names, 437 organizations and 88 locations. The English KB is derived from BaseKB, a cleaned version of English Freebase. 89.7% of these mentions can be linked to the KB. Using the multi-media approach, we retrieved 235 topically-related English documents.

7.2 Name Tagging Performance

Table 3 shows name tagging performance. We can see that our approach dramatically outperforms the supervised model. We conduct the Wilcoxon Matched-Pairs Signed-Ranks Test on ten folders. The results show that the improvement using visual evidence is significant at a 95% confidence level and the improvement using entity prior is significant at a 99% confidence level. Visual Evidence greatly improves organization tagging because most of them cannot be matched by spelling or pronunciation.

Face detection helps identify many person names missed by the supervised name tagger. For example, in the following sentence, "*Nawaz Shariff*" is mistakenly classified as a location by the supervised model due to the designator "*kasar (country)*" appearing in its left context. Since faces can be detected from all of the top 10 retrieved images (Figure 6), we fix its type to person.

- **Hausa document**: "*Yansanda sun dauki wannan matakin ne kwana daya bayanda PM kasar* ***Nawaz Shariff*** *ya fidda sanarwar inda ya bukaci...* (*The Police took this step a day after the PM of the country* ***Nawaz Shariff*** *threw out the statement in which he demanded that...*)"

[6] https://catalog.ldc.upenn.edu/LDC2011T07

Face detection is also effective to resolve classification ambiguity. For example, the common person name "*Haiyan*" can also be used to refer to the Typhoon in Southeast Asia. Both of our HL and LL name taggers mistakenly label "*Haiyan*" as a person in the following documents:

- **Hausa document**: "*...a yayinda mahaukaciyar guguwar teku da aka lakawa suna* ***Haiyan*** *ta fada tsibiran Leyte da Samar.* (*...as the violent typhoon, which has been given the name,* ***Haiyan****, has swept through the island of Leyte and Samar.*)"

- **Retrieved English comparable document**: "*As* ***Haiyan*** *heads west toward Vietnam, the Red Cross is at the forefront of an international effort to provide food, water, shelter and other relief...*"

In contrast using face detection results we successfully remove it based on processing the retrieved images as shown in Figure 7.

Figure 7: Top 9 Retrieved Images for 'Haiyan'.

Entity priors successfully provide more detailed and richer background knowledge than the comparable English documents. For example, the main topic of one Hausa document is the former president of Nigeria *Olusegun Obasanjo* accusing the current President *Goodluck Jonathan*, and a comment by the former 1990s military administrator of Kano *Bawa Abdullah Wase* is quoted. But *Bawa Abdullah Wase* is not mentioned in any related English documents. However, based on entity priors we observe that "*Bawa Abdullah Wase*" appears frequently in the same contexts as "*Nigeria*" and "*Kano*", and thus we successfully project it back to the Hausa sentence: "*Haka ma* ***Bawa Abdullahi Wase*** *ya ce akawai abun dubawa a kalamun*

System	Identification F-score				Classification Accuracy	Overall F-score
	PER	ORG	LOC[7]	ALL		
Supervised	36.52	38.64	42.38	40.25	76.84	30.93
Our Approach	**77.69**	**60.00**	**70.55**	**70.59**	**95.00**	**67.06**
Our Approach w/o Visual Evidence	73.77	46.58	70.74	67.98	94.77	64.43
Our Approach w/o Entity Prior	64.91	60.00	70.55	67.59	94.71	64.02

Table 3: Name Tagging Performance (%).

tsohon shugaban kasa kuma tsohon jamiin tsaro. (*In the same vein, Bawa Abdullahi Wase said that there were things to take away from the former President's words.*)". The impact of entity priors on person names is much more significant than other categories because multiple person entities often co-occur in some certain events or related topics which might not be fully covered in the retrieved English documents. In contrast most expected organizations and locations already exist in the retrieved English documents.

For the same local topic, Hausa documents usually describe more details than English documents, and include more unsalient entities. For example, for the president election in Ivory Coast, a Hausa document mentions the officials of the electoral body such as "*Damana Picasse*": "*Wani wakilin hukumar zaben daga jamiyyar shugaba Gbagbo,* ***Damana Picasse****, ya kekketa takardun sakamakon a gaban yan jarida, ya kuma ce ba na halal ba ne.* (*An official of the electoral body from president Gbagbo's party, Damana Picasse, tore up the result document in front of journalists, and said it is not legal.*)". In contrast, no English comparable documents mention their names. The entity prior method is able to extract many names which appear frequently together with the president name "*Gbagbo*".

7.3 Entity Linking Performance

Table 4 presents the Cross-lingual Entity Linking performance. We can see that our approach significantly outperforms our baseline and the best reported results on the same test set (Ji et al., 2015). Our approach is particularly effective for rare nicknames (e.g., "*C罗*" (C Luo) is used to refer to *Cristiano Ronaldo*) or ambiguous abbreviations (e.g., "*邦联*" (federal) can refer to *Confederate States of America*, *邦联制* (*Confederation*) and many other entities) for which the contexts in LLs are not sufficient for making correct linking decisions due to the lack of rich knowledge representation. Our approach produces worse linking results than the baseline for a few cases when the same abbreviation is used to refer to multiple entities in the same document. For example, when "*巴*" is used to refer to both "*巴西* (*Brazil*)" or "*巴勒斯坦* (*Palestine*)" in the same document, our approach mistakenly links all mentions to the same entity.

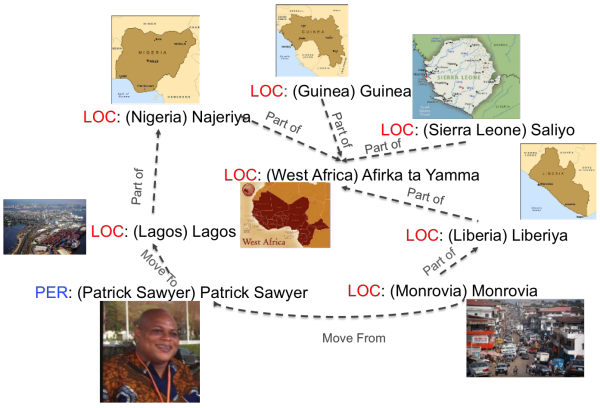


Figure 8: An Example of Cross-lingual Cross-media Knowledge Graph.

7.4 Cross-lingual Cross-media Knowledge Graph

As an end product, our framework will construct cross-lingual cross-media knowledge graphs. An example about the Ebola scenario is presented in Figure 8, including entity nodes extracted from both Hausa (LL) and English (HL), anchored by images; and edges extracted from English.

8 Related Work

Some previous cross-lingual projection methods focused on transferring data/annotation (e.g., (Padó and Lapata, 2009; Kim et al., 2010; Faruqui and Kumar, 2015)), shared feature representation/model (e.g., (McDonald et al., 2011; Kozhevnikov and Titov, 2013; Kozhevnikov and Titov, 2014)), or expectation (e.g., (Wang and Manning, 2014)). Most of them relied on a large

Approach	Overall					Linkable Entities				
	PER	ORG	GPE	LOC	ALL	PER	ORG	GPE	LOC	ALL
Baseline	49.12	60.18	80.97	80.68	66.57	67.27	67.61	81.05	80.68	74.70
State-of-the-art	49.85	64.30	75.38	**96.59**	65.87	68.28	72.24	75.46	**96.59**	73.91
Our Approach	**52.36**	**67.05**	**93.33**	93.18	**74.92**	**71.72**	**75.32**	**93.43**	93.18	**84.06**
Our Approach w/o KB Walker	50.44	67.05	84.41	90.91	70.32	69.09	75.32	84.50	90.91	78.91

Table 4: Cross-lingual Entity Linking Accuracy (%).

amount of parallel data to derive word alignment and translations, which are inadequate for many LLs. In contrast, we do not require any parallel data or bi-lingual lexicon. We introduce new cross-media techniques for projecting HLs to LLs, by inferring projections using domain-rich, non-parallel data automatically discovered by image search and processing. Similar image-mediated approaches have been applied to other tasks such as cross-lingual document retrieval (Funaki and Nakayama, 2015) and bilingual lexicon induction (Bergsma and Van Durme, 2011). Besides visual similarity, their method also relied on distributional similarity computed from a large amount of unlabeled data, which might not be available for some LLs.

Our name projection and validation approaches are similar to other previous work on bi-lingual lexicon induction from non-parallel corpora (e.g., (Fung and Yee, 1998; Rapp, 1999; Shao and Ng, 2004; Munteanu and Marcu, 2005; Sproat et al., 2006; Klementiev and Roth, 2006; Hassan et al., 2007; Udupa et al., 2009; Ji, 2009; Darwish, 2010; Noeman and Madkour, 2010; Bergsma and Van Durme, 2011; Radford et al., ; Irvine and Callison-Burch, 2013; Irvine and Callison-Burch, 2015)) and name translation mining from multilingual resources such as Wikipedia (e.g. (Sorg and Cimiano, 2008; Adar et al., 2009; Nabende, 2010; Lin et al., 2011)). We introduce new multimedia evidence such as visual similarity and face recognition for name validation, and also exploit a large amount of monolingual HL data for mining entity priors to expand the expected entity set.

For Cross-lingual Entity Linking, some recent work (Finin et al., 2015) also found cross-lingual coreference resolution can greatly reduce ambiguity. Some other methods also utilized global knowledge in the English KB to improve linking accuracy via quantifying link types (Wang et al., 2015b), computing pointwise mutual information for the Wikipedia categories of consecutive pairs of entities (Sil et al., 2015), or using linking as feedback to improve name classification (Sil and Yates, 2013; Heinzerling et al., 2015; Besancon et al., 2015; Sil et al., 2015).

9 Conclusions and Future Work

We describe a novel multi-media approach to effectively transfer entity knowledge from high-resource languages to low-resource languages. In the future we will apply visual pattern recognition and concept detection techniques to perform deep content analysis of the retrieved images, so we can do matching and inference on concept/entity level instead of shallow visual similarity. We will also extend anchor image retrieval from document-level into phrase-level or sentence-level to obtain richer background information. Furthermore, we will exploit edge labels while walking through a knowledge base to retrieve more relevant entities. Our long-term goal is to extend this framework to other knowledge extraction and population tasks such as event extraction and slot filling to construct multimedia knowledge bases effectively from multiple languages with low cost.

Acknowledgments

This work was supported by the U.S. DARPA LORELEI Program No. HR0011-15-C-0115, ARL/ARO MURI W911NF-10-1-0533, DARPA Multimedia Seedling grant, DARPA DEFT No. FA8750-13-2-0041 and FA8750-13-2-0045, and NSF CAREER No. IIS-1523198. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Eytan Adar, Michael Skinner, and Daniel S Weld. 2009. Information arbitrage across multi-lingual

wikipedia. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 94–103.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of Association for Computational Linguistics 2013 Workshop on Linguistic Annotation and Interoperability with Discourse*.

Utsab Barman, Amitava Das, Joachim Wagner, and Jennifer Foster. 2014. Code mixing: A challenge for language identification in the language of social media. In *Proceedings of Conference on Empirical Methods in Natural Language Processing Workshop on Computational Approaches to Code Switching*.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Romaric Besancon, Hani Daher, Hervé Le Borgne, Olivier Ferret, Anne-Laure Daquo, and Adrian Popescu. 2015. Cea list participation at tac edl english diagnostic task. In *Proceedings of the Text Analysis Conference*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3(993-1022).

William B Cavnar, John M Trenkle, et al. 1994. N-gram-based text categorization. In *Proceedings of SDAIR1994*.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Kareem Darwish. 2010. Transliteration mining with phonetic conflation and iterative training. In *Proceedings of the 2010 Named Entities Workshop*.

Manaal Faruqui and Shankar Kumar. 2015. Multilingual open relation extraction using cross-lingual projection. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*.

Tim Finin, Dawn Lawrie, Paul McNamee, James Mayfield, Douglas Oard, Nanyun Peng, Ning Gao, Yiu-Chang Lin, Josh MacLin, and Tim Dowd. 2015. HLTCOE participation in TAC KBP 2015: Cold start and TEDL. In *Proceedings of the Text Analysis Conference*.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Pascale Fung and Lo Yuen Yee. 1998. An ir approach for translating new words from nonparallel and comparable texts. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*.

Ahmed Hassan, Haytham Fahmy, and Hany Hassan. 2007. Improving named entity translation by exploiting comparable and parallel corpora. *AMML07*.

Benjamin Heinzerling, Alex Judea, and Michael Strube. 2015. Hits at tac kbp 2015: Entity discovery and linking, and event nugget detection. In *Proceedings of the Text Analysis Conference*.

Ann Irvine and Chris Callison-Burch. 2013. Combining bilingual and comparable corpora for low resource machine translation. In *Proc. WMT*.

Ann Irvine and Chris Callison-Burch. 2015. Discriminative Bilingual Lexicon Induction. *Computational Linguistics*, 1(1).

Heng Ji, Joel Nothman, Ben Hachey, and Radu Florian. 2015. Overview of tac-kbp2015 tri-lingual entity discovery and linking. In *Proceedings of the Text Analysis Conference*.

Heng Ji. 2009. Mining name translations from comparable corpora by creating bilingual information networks. In *Proceedings of the 2nd Workshop on Building and Using Comparable Corpora: from Parallel to Non-parallel Corpora*.

Yan Ke, Rahul Sukthankar, Larry Huston, Yan Ke, and Rahul Sukthankar. 2004. Efficient near-duplicate detection and sub-image retrieval. In *Proceedings of ACM International Conference on Multimedia*.

Seokhwan Kim, Minwoo Jeong, Jonghoon Lee, and Gary Geunbae Lee. 2010. A cross-lingual annotation projection approach for relation detection. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Alexandre Klementiev and Dan Roth. 2006. Weakly supervised named entity transliteration and discovery from multilingual comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

Mikhail Kozhevnikov and Ivan Titov. 2013. Cross-lingual transfer of semantic role labeling models. In *Proceedings of Association for Computational Linguistics*.

Mikhail Kozhevnikov and Ivan Titov. 2014. Cross-lingual model transfer using feature representation projection. In *Proceedings of Association for Computational Linguistics*.

Qi Li, Heng Ji, Yu Hong, and Sujian Li. 2014. Constructing information networks using one single model. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Wen-Pin Lin, Matthew Snover, and Heng Ji. 2011. Unsupervised language-independent name translation mining from wikipedia infoboxes. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 43–52. Association for Computational Linguistics.

David G Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on Computer vision*.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *In Proceedings of Association for Computational Linguistics*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Olena Medelyan and Catherine Legg. 2008. Integrating cyc and wikipedia: Folksonomy meets rigorously defined common-sense. In *Wikipedia and Artificial Intelligence: An Evolving Synergy, Papers from the 2008 AAAI Workshop*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4)(477–504).

Peter Nabende. 2010. Mining transliterations from wikipedia using pair hmms. In *Proceedings of the 2010 Named Entities Workshop*, pages 76–80. Association for Computational Linguistics.

PR Newswire. 2011. Earned media evolved. In *White Paper*.

Sara Noeman and Amgad Madkour. 2010. Language independent transliteration mining system using finite state automata framework. In *Proceedings of the 2010 Named Entities Workshop*, pages 57–61. Association for Computational Linguistics.

Margaret King Odell. 1956. *The Profit in Records Management*. Systems (New York).

Sebastian Padó and Mirella Lapata. 2009. Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.

Xiaoman Pan, Taylor Cassidy, Ulf Hermjakob, Heng Ji, and Kevin Knight. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics–Human Language Technologies*.

Lawrence Philips. 1990. Hanging on the metaphone. *Computer Language*, 7(12).

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

Reinhard Rapp. 1999. Automatic identification of word translations from unrelated English and German corpora. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 519–526.

Li Shao and Hwee Tou Ng. 2004. Mining new word translations from comparable corpora. In *Proceedings of the 20th international conference on Computational Linguistics*, page 618.

Avirup Sil and Alexander Yates. 2013. Re-ranking for joint named-entity recognition and linking. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*.

Avirup Sil, Georgiana Dinu, and Radu Florian. 2015. The ibm systems for trilingual entity discovery and linking at tac 2015. In *Proceedings of the Text Analysis Conference*.

Philipp Sorg and Philipp Cimiano. 2008. Enriching the crosslingual link structure of wikipedia-a classification-based approach. In *Proceedings of the AAAI 2008 Workshop on Wikipedia and Artifical Intelligence*.

Richard Sproat, Tao Tao, and ChengXiang Zhai. 2006. Named entity transliteration with comparable corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*.

Robert L Taft. 1970. *Name Search Techniques*. New York State Identification and Intelligence System, Albany, New York, US.

Raghavendra Udupa, K Saravanan, A Kumaran, and Jagadeesh Jagarlamudi. 2009. Mint: A method for effective and scalable mining of named entity transliterations from large comparable corpora. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 799–807.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Clare R Voss, Stephen Tratz, Jamal Laoudi, and Douglas M Briesch. 2014. Finding romanized arabic dialect in code-mixed tweets. In *Proceedings of Language Resources and Evaluation Conference*.

Mengqiu Wang and Christopher D Manning. 2014. Cross-lingual projected expectation regularization for weakly supervised learning. *Transactions of the Association for Computational Linguistics*.

Xuanhui Wang, ChengXiang Zhai, Xiao Hu, and Richard Sproat. 2007. Mining correlated bursty topic patterns from coordinated text streams. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*.

Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015a. Boosting Transition-based AMR Parsing with Refined Actions and Auxiliary Analyzers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Han Wang, Jin Guang Zheng, Xiaogang Ma, Peter Fox, and Heng Ji. 2015b. Language and domain independent entity linking with quantified collective validation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Models and Inference for Prefix-Constrained Machine Translation

Joern Wuebker, Spence Green, John DeNero, Saša Hasan
Lilt, Inc.
first_name@lilt.com

Minh-Thang Luong
Stanford University
lmthang@stanford.edu

Abstract

We apply phrase-based and neural models to a core task in interactive machine translation: suggesting how to complete a partial translation. For the phrase-based system, we demonstrate improvements in suggestion quality using novel objective functions, learning techniques, and inference algorithms tailored to this task. Our contributions include new tunable metrics, an improved beam search strategy, an n-best extraction method that increases suggestion diversity, and a tuning procedure for a hierarchical joint model of alignment and translation. The combination of these techniques improves next-word suggestion accuracy dramatically from 28.5% to 41.2% in a large-scale English-German experiment. Our recurrent neural translation system increases accuracy yet further to 53.0%, but inference is two orders of magnitude slower. Manual error analysis shows the strengths and weaknesses of both approaches.

1 Introduction

A core prediction task in interactive machine translation (MT) is to complete a partial translation (Ortiz-Martínez et al., 2009; Koehn et al., 2014). Sentence completion enables interfaces that are richer than basic post-editing of MT output. For example, the translator can receive updated suggestions after each word typed (Langlais et al., 2000). However, we show that completing partial translations by naïve constrained decoding—the standard in prior work—yields poor suggestion quality. We describe new phrase-based objective functions, learning techniques, and inference algorithms for the sentence completion task.[1] We then compare this improved phrase-based system to a state-of-the-art recurrent neural translation system in large-scale English-German experiments.

A system for completing partial translations takes as input a source sentence and a prefix of the target sentence. It predicts a suffix: a sequence of tokens that extends the prefix to form a full sentence. In an interactive setting, the first words of the suffix are critical; these words are the focus of the user's attention and can typically be appended to the translation with a single keystroke. We introduce a tuning metric that scores correctness of the whole suffix, but is particularly sensitive to these first words.

Phrase-based inference for this task involves aligning the prefix to the source, then generating the suffix by translating the unaligned words. We describe a beam search strategy and a hierarchical joint model of alignment and translation that together improve suggestions dramatically. For English-German news, next-word accuracy increases from 28.5% to 41.2%.

An interactive MT system could also display multiple suggestions to the user. We describe an algorithm for efficiently finding the n-best next words directly following a prefix and their corresponding best suffixes. Our experiments show that this approach to n-best list extraction, combined with our other improvements, increased next-word suggestion accuracy of 10-best lists from 33.4% to 55.5%.

We also train a recurrent neural translation system to maximize the conditional likelihood of the next word following a translation prefix, which is both a standard training objective in neural translation and an ideal fit for our task. This neural system provides even more accurate predictions than our improved phrase-based system. However, inference is two orders of magnitude slower, which is prob-

[1] Code available at:
https://github.com/stanfordnlp/phrasal

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 66–75,
Berlin, Germany, August 7-12, 2016.

lematic for an interactive setting. We conclude with a manual error analysis that reveals the strengths and weaknesses of both the phrase-based and neural approaches to suffix prediction.

2 Evaluating Suffix Prediction

Let $\mathcal{F}$ and $\mathcal{E}$ denote the set of all source and target language strings, respectively. Given a source sentence $f \in \mathcal{F}$ and target prefix $e_p \in \mathcal{E}$, a predicted suffix $e_s \in \mathcal{E}$ can be evaluated by comparing the full sentence $e = e_p e_s$ to a reference e^*. Let e^*_s denote the suffix of the reference that follows e_p.

We define three metrics below that score translations by the characteristics that are most relevant in an interactive setting: the accuracy of the first words of the suffix and the overall quality of the suffix. Each metric takes example triples (f, e_p, e^*) produced during an interactive MT session in which e_p was generated in the process of constructing e^*.

A *simulated* corpus of examples can be produced from a parallel corpus of (f, e^*) pairs by selecting prefixes of each e^*. An *exhaustive* simulation selects all possible prefixes, while a *sampled* simulation selects only k prefixes uniformly at random for each e^*. Computing metrics for exhaustive simulations is expensive because it requires performing suffix prediction inference for every prefix: $|e^*|$ times for each reference.

Word Prediction Accuracy (WPA) or *next-word accuracy* (Koehn et al., 2014) is 1 if the first word of the predicted suffix e_s is also the first word of reference suffix e^*_s, and 0 otherwise. Averaging over examples gives the frequency that the word following the prefix was predicted correctly. In a sampled simulation, all reference words that follow the first word of a sampled suffix are ignored by the metric, so most reference information is unused.

Number of Predicted Words (#prd) is the maximum number of contiguous words at the start of the predicted suffix that match the reference. Like WPA, this metric is 0 if the first word of e_s is not also the first word of e^*_s. In a sampled simulation, all reference words that follow the first mis-predicted word in the sampled suffix are ignored. While it is possible that the metric will require the full reference suffix, most reference information is unused in practice.

Prefix-BLEU (pxBLEU): BLEU (Papineni et al., 2002) is computed from the geometric mean of clipped n-gram precisions $prec_n(\cdot, \cdot)$ and a brevity penalty $BP(\cdot, \cdot)$. Given a sequence of references $E^* = e^*_1, \ldots, e^*_t$ and corresponding predictions $E = e_1, \ldots, e_t$,

$$\textsc{Bleu}(E, E^*) = BP(E, E^*) \cdot \prod_{n=1}^{4} prec_n(E, E^*)^{\frac{1}{4}}$$

Ortiz-Martínez et al. (2010) use BLEU directly for training an interactive system, but we propose a variant that only scores the predicted suffix and not the input prefix. The pxBLEU metric computes $\textsc{Bleu}(\hat{E}, \hat{E}^*)$ for the following constructed sequences $\hat{E}$ and $\hat{E}^*$:

- For each (f, e_p, e^*) and suffix prediction e_s, $\hat{E}$ includes the full sentence $e = e_p e_s$.
- For each (f, e_p, e^*), $\hat{E}^*$ is a masked copy of e^* in which all prefix words that do not match any word in e are replaced by null tokens.

This construction maintains the original computation of the brevity penalty, but does not include the prefix in the precision calculations. Unlike the two previous metrics, the pxBLEU metric uses all available reference information.

In order to account for boundary conditions, the reference e^* is masked by the prefix e_p as follows: we replace each of the first $|e_p - 3|$ words with a null token e_{null}, unless the word also appears in the suffix e^*_s. Masking retains the last three words of the prefix so that the first words after the prefix can contribute to the precision of all n-grams that overlap with the prefix, up to $n = 4$. Words that also appear in the suffix are retained so that their correct prediction in the suffix can contribute to those precisions, which would otherwise be clipped.

2.1 Loss Functions for Learning

All of these metrics can be used as the tuning objective of a phrase-based machine translation system. Tuning toward a sampled simulation that includes one or two prefixes per reference is much faster than using an exhaustive set of prefixes. A linear combination of these metrics can be used to trade off the relative importance of the full suffix and the words immediately following the prefix. With a combined metric, learning can focus on these words while using all available information in the references.

2.2 Keystroke Ratio (KSR)

In addition to these metrics, suffix prediction can be evaluated by the widely used keystroke ratio (KSR) metric (Och et al., 2003). This ratio assumes that

any number of characters from the beginning of the suggested suffix can be appended to the user prefix using a single keystroke. It computes the ratio of key strokes required to enter the reference interactively to the character count of the reference. Our MT architecture does not permit tuning to KSR.

Other methods of quantifying effort in an interactive MT system are more appropriate for user studies than for direct evaluation of MT predictions. For example, measuring pupil dilation, pause duration and frequency (Schilperoord, 1996), mouse-action ratio (Sanchis-Trilles et al., 2008), or source difficulty (Bernth and McCord, 2000) would certainly be relevant for evaluating a full interactive system, but are beyond the scope of this work.

3 Phrase-Based Inference

In the log-linear approach to phrase-based translation (Och and Ney, 2004), the distribution of translations $e \in \mathcal{E}$ given a source sentence $f \in \mathcal{F}$ is:

$$p(e|f;w) = \sum_{\substack{r: \\ src(r)=f \\ tgt(r)=e}} \frac{1}{Z(f)} \exp\left[w^\top \phi(r)\right] \quad (1)$$

Here, r is a phrasal derivation with source and target projections $src(r)$ and $tgt(r)$, $w \in \mathbb{R}^d$ is the vector of model parameters, $\phi(\cdot) \in \mathbb{R}^d$ is a feature map, and $Z(f)$ is an appropriate normalizing constant.

For the same model, the distribution over suffixes $e_s \in \mathcal{E}$ must also condition on a prefix $e_p \in \mathcal{E}$:

$$p(e_s|e_p, f;w) = \sum_{\substack{r: \\ src(r)=f \\ tgt(r)=e_p e_s}} \frac{1}{Z(f)} \exp\left[w^\top \phi(r)\right] \quad (2)$$

In phrase-based decoding, the best scoring derivation r given a source sentence f and weights w is found efficiently by beam search, with one beam for every count of source words covered by a partial derivation (known as the *source coverage cardinality*). To predict a suffix conditioned on a prefix by constrained decoding, Barrachina et al. (2008) and Ortiz-Martínez et al. (2009) modify the beam search by discarding hypotheses (partial derivations) that do not match the prefix e_p.

We propose *target beam search*, a two-step inference procedure. The first step is to produce a phrase-based alignment between the target prefix and a subset of the source words. The target is aligned left-to-right by appending aligned phrase pairs. However, each beam is associated with a target word count, rather than a source word count. Therefore, each beam contains hypotheses for a fixed prefix of target words. Phrasal translation candidates are bundled and sorted with respect to each target phrase rather than each source phrase. Crucially, the source distortion limit is not enforced during alignment, so that long-range reorderings can be analyzed correctly.

The second step generates the suffix using standard beam search.[2] Once the target prefix is completely aligned, each hypothesis from the final target beam is copied to an appropriate source beam. Search starts with the lowest-count source beam that contains at least one hypothesis. Here, we re-instate the distortion limit with the following modification to avoid search failures: The decoder can always translate any source position before the last source position that was covered in the alignment phase.

3.1 Synthetic Phrase Pairs

The phrase pairs available during decoding may not be sufficient to align the target prefix to the source. Pre-compiled phrase tables (Koehn et al., 2003) are typically pruned, and dynamic phrase tables (Levenberg et al., 2010) require sampling for efficient lookup.

To improve alignment coverage, we include additional *synthetic phrases* extracted from word-level alignments between the source sentence and target prefix inferred using unpruned lexical statistics.

We first find the intersection of two directional word alignments. The directional alignments are obtained similar to IBM Model 2 (Brown et al., 1993) by aligning the most likely source word to each target word. Given a source sequence $f = f_1 \dots f_{|f|}$ and a target sequence $e = e_1 \dots e_{|e|}$, we define the alignment $a = a_1 \dots a_{|e|}$, where $a_i = j$ means that e_i is aligned to f_j. The likelihood is modeled by a single-word lexicon probability that is provided by our translation model and an alignment probability modeled as a Poisson distribution $Poisson(k, \lambda)$ in the distance to the diagonal.

$$a_i = \operatorname*{arg\,max}_{j \in \{1,\dots,|f|\}} p(a_i = j|f, e) \quad (3)$$

$$p(a_i = j|f, e) = p(e_i|f_j) \cdot p(a_i|j) \quad (4)$$

$$p(e_i|f_j) = \frac{cnt(e_i, f_j)}{cnt(f_j)} \quad (5)$$

$$p(a_i|j) = Poisson(|a_i - j|, 1.0) \quad (6)$$

[2] We choose cube pruning (Huang and Chiang, 2007) as the beam-filling strategy.

Here, $cnt(e_i, f_j)$ is the count of all word alignments between e_i and f_j in the training bitext, and $cnt(f_j)$ the monolingual occurrence count of f_j.

We perform standard phrase extraction (Och et al., 1999; Koehn et al., 2003) to obtain our synthetic phrases, whose translation probabilities are again estimated based on the single-word probabilities $p(e_i|f_j)$ from our translation model. Given a synthetic phrase pair (e, f), the phrase translation probability is computed as

$$p(e|f) = \prod_{1 \leq i \leq |e|} \max_{1 \leq j \leq |f|} p(e_i|f_j) \quad (7)$$

Additionally, we introduce three indicator features that count the number of synthetic phrase pairs, source words and target words, respectively.

4 Tuning

In order to tune the model for suffix prediction, we optimize the weights w in Equation 2 to maximize the metrics introduced in Section 2. Model tuning is performed with AdaGrad (Duchi et al., 2011), an online subgradient method. It features an adaptive learning rate and comes with good theoretical guarantees. See Green et al. (2013) for the details of applying AdaGrad to phrase-based translation.

The same model scores both alignment of the prefix and translation of the suffix. However, different feature weights may be appropriate for scoring each step of the inference process. In order to learn different weights for alignment and translation within a unified joint model, we apply the hierarchical adaptation method of Wuebker et al. (2015), which is based on *frustratingly easy domain adaptation* (FEDA) (Daumé III, 2007). We define three sub-segment domains: PREFIX, OVERLAP and SUFFIX. The PREFIX domain contains all phrases that are used for aligning the prefix with the source sentence. Phrases that span both prefix and suffix additionally belong to the OVERLAP domain. Finally, once the prefix has been completely covered, the SUFFIX domain applies to all phrases that are used to translate the remainder of the sentence. The ROOT domain spans the entire phrasal derivation.

Formally, given a set of domains $\mathcal{D} = \{\text{ROOT}, \text{PREFIX}, \text{OVERLAP}, \text{SUFFIX}\}$, each feature is replicated for each domain $d \in \mathcal{D}$. These replicas can be interpreted as domain-specific "offsets" to the baseline weights. For an original feature vector ϕ with a set of domains $D \subseteq \mathcal{D}$, the replicated feature vector contains $|\mathcal{D}|$ copies f_d of each feature $f \in \phi$, one for each $d \in \mathcal{D}$.

$$f_d = \begin{cases} f, & d \in D \\ 0, & \text{otherwise.} \end{cases} \quad (8)$$

The weights of the replicated feature space are initialized with 0 except for the ROOT domain, where we copy the baseline weights w.

$$w_d = \begin{cases} w, & d \text{ is ROOT} \\ 0, & \text{otherwise.} \end{cases} \quad (9)$$

All our phrase-based systems are first tuned without prefixes or domains to maximize BLEU. When tuning for suffix prediction, we keep these baseline weights w_{ROOT} fixed to maintain baseline translation quality and only update the weights corresponding to the PREFIX, OVERLAP and SUFFIX domains.

5 Diverse n-best Extraction

Consider the interactive MT application setting in which the user is presented with an autocomplete list of alternative translations (Langlais et al., 2000). The user query may be satisfied if the machine predicts the correct completion in its top-n output. However, it is well-known that n-best lists are poor approximations of MT structured output spaces (Macherey et al., 2008; Gimpel et al., 2013). Even very large values of n can fail to produce alternatives that differ in the first words of the suffix, which limits n-best KSR and WPA improvements at test time. For tuning, WPA is often zero for every item on the n-best list, which prevents learning.

Fortunately, the prefix can help efficiently enumerate diverse next-word alternatives. If we can find all edges in the decoding lattice that span the prefix e_p and suffix e_s, then we can generate diverse alternatives in precisely the right location in the target. Let $G = (V, E)$ be the search lattice created by decoding, where V are nodes and E are the edges produced by rule applications. For any $w \in V$, let $\text{parent}(w)$ return v s.t. $v, w \in E$, $\text{target}(w)$ return the target sequence e defined by following the next pointers from w, and $\text{length}(w)$ be the length of the target sequence up to w. During decoding, we set parent pointers and also assign monotonically increasing integer ids to each w.

To extract a full sentence completion given an edge $v, w \in E$ that spans the prefix/suffix boundary, we must find the best path to a goal node efficiently. To do this, we sort V in reverse topological order and set forward pointers from each node v to the

Algorithm 1 Diverse n-best list extraction

Require: Lattice $G = (V, E)$, prefix length P
$M = []$ ▷ Marked nodes
for $w \in V$ in reverse topological order **do**
  $v = \text{parent}(w)$ ▷ $v, w \in E$
  if $\text{length}(v) \leq P$ and $\text{length}(w) > P$ **then**
    Add w to M ▷ Mark node
  end if
  $v.\text{child} = v.\text{child} \oplus w$ ▷ Child pointer update
end for
$N = []$ ▷ n-best target strings
for $m \in M$ **do**
  Add $\text{target}(m)$ to N
end for
return N

child node on the best goal path. During this traversal, we also mark all child nodes of edges that span the prefix/suffix boundary. Finally, we use the parent and child pointers to extract an n-best list of translations. Algorithm 1 shows the full procedure.

6 Neural machine translation

Neural machine translation (NMT) models the conditional probability $p(e|f)$ of translating a source sentence f to a target sentence e. In the *encoder-decoder* NMT framework (Sutskever et al., 2014; Cho et al., 2014), an *encoder* computes a representation $\boldsymbol{s}$ for each source sentence. From that source representation, the *decoder* generates a translation one word at a time by maximizing:

$$\log p(e|f) = \sum_{i=1}^{|e|} \log p\left(e_i|e_{<i}, f, \boldsymbol{s}\right) \qquad (10)$$

The individual probabilities in Equation 10 are often parameterized by a recurrent neural network which repeatedly predicts the next word e_i given all previous target words $e_{<i}$. Since this model generates translations by repeatedly predicting next words, it is a natural choice for the sentence completion task. Even in unconstrained decoding, it predicts one word at a time conditioned on the most likely prefix.

We modified the state-of-the-art English-German NMT system described in (Luong et al., 2015) to conduct a beam search that constrains the translation to match a fixed prefix.[3] As we decode from left to right, the decoder transitions from a constrained prefix decoding mode to unconstrained beam search. In the constrained mode—the next word to predict e_i is known—we set the beam size to 1, aggregate the score of predicting e_i immediately without having to sort the softmax distribution over all words, and feed e_i directly to the next time step. Once the prefix has been consumed, the decoder switches to standard beam search with a larger beam size (12 in our experiments). In this mode, the most probable word e_i is passed to the next time step.

7 Experimental Results

We evaluate our models and methods for English-French and English-German on two domains: *software* and *news*.

The phrase-based systems are built with Phrasal (Green et al., 2014), an open source toolkit. We use a dynamic phrase table (Levenberg et al., 2010) and tune parameters with AdaGrad. All systems have 42 dense baseline features. We align the bitexts with *mgiza* (Gao and Vogel, 2008) and estimate 5-gram language models (LMs) with *KenLM* (Heafield et al., 2013).

The English-French bilingual training data consists of 4.9M sentence pairs from the Common Crawl and Europarl corpora from WMT 2015 (Bojar et al., 2015). The LM was estimated from the target side of the bitext.

For English-German we run large-scale experiments. The bitext contains 19.9M parallel segments collected from WMT 2015 and the OPUS collection (Skadiņš et al., 2014). The LM was estimated from the target side of the bitext and the monolingual Common Crawl corpus (Buck et al., 2014), altogether 37.2B running words.

The software test set includes 10k sentence pairs from the Autodesk post editing corpus[4]. For the news domain we chose the English-French *newstest2014* and English-German *newstest2015* sets provided for the WMT 2016[5] shared task. The translation systems were tuned towards the specific domain, using another 10k segments from the Autodesk data or the *newstest2013* data set, respectively. On the English-French tune set we randomly select one target prefix from each sentence pair for rapid experimentation. On all other test and tune sets we select two target prefixes at random.[6] The

[3] We used the trained models provided by the authors of (Luong et al., 2015) using the codebase at `https://github.com/lmthang/nmt.matlab`.

[4] `https://autodesk.app.box.com/Autodesk-PostEditing`

[5] `http://www.statmt.org/wmt16`

[6] We briefly experimented with larger sets of prefixes and also exhaustive simulation in tuning, but did not observe significant improvements.

selected prefixes remain fixed throughout all experiments.

For NMT, we report results both using a single network and an ensemble of eight models using various attention mechanisms (Luong et al., 2015).

7.1 Phrase-based Results

Tables 1 and 2 show the main phrase-based results. The baseline system corresponds to constrained beam search, which performed best in (Ortiz-Martínez et al., 2009) and (Barrachina et al., 2008), where it was referred to as *phrase-based (PB)* and *phrase-based model (PBM)*, respectively. Our *target beam search* strategy improves all metrics on both test sets.

For English-French, we observe absolute improvements of up to 3.2% pxBLEU, 11.4% WPA and 10.6% KSR. We experimented with four different prefix-constrained tuning criteria: pxBLEU, WPA, #prd, and the linear combination $\frac{(\text{pxBLEU}+\text{WPA})}{2}$. We see that tuning towards prefix decoding increases all metrics. Across our two test sets, the combined metric yielded the most stable results. Here, we obtain gains of up to 3.0% pxBLEU, 3.1% WPA and 2.1% KSR. We continue using the linear combination criterion for all subsequent experiments.

For English-German—the large-scale setting—we observe similar total gains of up to 3.9% pxBLEU, 11.2% WPA and 8.2% KSR. The target beam search procedure contributes the most gain among our various improvements. Table 3 illustrates the differences in the translation output on three example sentences taken from the *newstest2015* test set. It is clearly visible that both target beam search and prefix tuning improve the prefix alignment, which results in better translation suffixes.

7.2 Diverse n-best Results

To improve recall in interactive MT, the user can be presented with multiple alternative sentence completions (Langlais et al., 2000), which correspond to an n-best list of translation hypotheses generated by the prefix-constrained inference procedure. The diverse extraction scheme introduced in section 5 is particularly designed for next-word prediction recall. Table 4 shows results for 10-best lists.

We see that WPA is increased by up to 15.3% by including the 10-best candidates, 11.3% being contributed by our novel diverse n-best extraction. Jointly, target beam search, prefix tuning and diverse n-best extraction lead to an absolute improvement of up to 23.5% over the baseline 10-best oracle. We believe that $n = 10$ suggestions are the maximum number of candidates that should be presented to a user, but we also ran experiments with $n = 3$ and $n = 5$, which would result in an interface with reduced cognitive load. These settings yield 5.5% and 10.0% WPA gains respectively on English-German news.

7.3 Comparison with NMT

We compare this phrase-based system to the NMT system described in Section 6 for English-German. Table 5 shows the results. We observe a clear advantage of NMT over our best phrase-based system when comparing WPA. For pxBLEU, the phrase-based model outperforms the single neural network system on the Autodesk set, but underperforms the ensemble. This stands in contrast to unconstrained full-sentence translation quality, where the phrase-based system is slightly better than the ensemble. The neural system substantially outperforms the phrase-based system for all metrics in the *news* domain.

In an interactive setting, the system must make predictions in near real-time, so we report average decoding times. We observe a clear time vs. accuracy trade-off; the phrase-based is 10.6 to 31.3 times faster than the single network NMT system and more than 100 times faster than the ensemble. Crucially, the phrase-based system runs on a CPU, while NMT requires a GPU for these speeds. Further, the 10-best oracle WPA of the phrase-based system is higher than the NMT ensemble in both genres.

Following the example of Neubig et al. (2015), we performed a manual analysis of the first 100 segments on the newstest2015 data set in order to qualitatively compare the constrained translations produced by the phrase-based and single network NMT systems. We observe four main error categories in which the translations differ, for which we have given examples in Table 6. NMT is generally better with long-range verb reorderings, which often lead to the verb being dropped by the phrase-based system. E.g. the word *erscheinen* in Ex. 1 and *veröffentlicht* in Ex. 2 are missing in the phrase-based translation. Also, the NMT engine often produces better German grammar and morphological agreement, e.g. *kein* vs. *keine* in Ex. 3 or the verb conjugations in Ex. 4. Especially interesting is that the NMT system generated the negation *nicht* in the second half of Ex. 3. This word does not have

	tuning criterion	autodesk pxBLEU	WPA	#prd	KSR	newstest2014 pxBLEU	WPA	#prd	KSR
baseline	BLEU	57.9	41.1	1.49	57.8	40.9	38.0	0.96	61.7
target beam search	BLEU	61.0	47.2	1.74	50.3	44.1	49.4	1.35	51.1
+ prefix tuning	$\frac{(\text{pxBLEU}+\text{WPA})}{2}$	64.0	50.3	1.95	48.2	44.7	50.9	1.40	50.5
	pxBLEU	64.0	50.1	1.95	48.2	44.9	50.3	1.38	50.8
	WPA	62.4	50.2	1.88	48.1	43.3	50.5	1.34	51.7
	#prd	63.8	49.7	1.95	48.4	44.1	50.3	1.37	50.7

Table 1: Phrase-based results on the English-French task. We compare the baseline with the target beam search proposed in this work. Prefix tuning is evaluated with four different tuning criteria.

	autodesk pxBLEU	WPA	#prd	KSR	newstest2015 pxBLEU	WPA	#prd	KSR
baseline	58.5	37.8	1.54	64.7	32.1	28.5	0.61	72.7
target beam search	61.2	44.6	1.78	58.0	36.0	39.7	0.84	64.5
+ prefix tuning	62.2	46.0	1.85	57.2	36.0	41.2	0.88	63.7

Table 2: Phrase-based results on English-German, tuned to the linear combination of pxBLEU and WPA.

a direct correspondence in the English source, but makes the sentence feel more natural in German. On the other hand, NMT sometimes drops content words, as in Ex. 5, where *middle-class jobs*, *Minnesota* and *Progressive Caucus co-chair* remain entirely untranslated by NMT. Finally, incorrect prefix alignment sometimes leads to incorrect portions of the source sentence being translated after the prefix or even superfluous output by the phrase-based engine, like *, die* in Ex. 6. Table 7 summarizes how many times each of the systems produced a better output than the other, broken down by category.

8 Related Work

Target-mediated interactive MT was first proposed by Foster et al. (1997) and then further developed within the TransType (Langlais et al., 2000) and TransType2 (Esteban et al., 2004; Barrachina et al., 2008) projects. In TransType2, several different approaches were evaluated. Barrachina et al. (2008) reports experimental results that show the superiority of phrase-based models over stochastic finite state transducers and alignment templates, which were extended for the interactive translation paradigm by Och et al. (2003). Ortiz-Martínez et al. (2009) confirm this observation, and find that their own suggested method using partial statistical phrase-based alignments performs on a similar level on most tasks. The approach using phrase-based models is used as the baseline in this paper.

In order to make the interaction sufficiently responsive, Barrachina et al. (2008) resort to search within a word graph, which is generated by the translation decoder without constraints at the beginning of the workflow. A given prefix is then matched to the paths within the word graph. This approach was recently refined with more permissive matching criteria by Koehn et al. (2014), who report strong improvements in prediction accuracy.

Instead of using a word graph, it is also possible to perform a new search for every interaction (Bender et al., 2005; Ortiz-Martínez et al., 2009), which is the approach we have adopted. Ortiz-Martínez et al. (2009) perform the most similar study to our work in the literature. The authors also define prefix decoding as a two-stage process, but focus on investigating different smoothing techniques, while our work includes new metrics, models, and inference.

9 Conclusion

We have shown that both phrase-based and neural translation approaches can be used to complete partial translations. The recurrent neural system provides higher word prediction accuracy, but requires lengthy inference on a GPU. The phrase-based system is fast, produces diverse n-best lists, and provides reasonable prefix-BLEU performance. The complementary strengths of both systems suggest future work in combining these techniques.

We have also shown decisively that simply performing constrained decoding for a phrase-based model is not an effective approach to the task of completing translations. Instead, the learning objective, model, and inference procedure should all

1.	**source**	Suddenly I'm at the National Theatre and I just couldn't quite believe it.
	reference	*"Plötzlich war ich im* Nationaltheater und ich konnte es kaum glauben.
	baseline	*"Plötzlich war ich im* Nationaltheater bin und ich konnte es einfach nicht glauben.
	target beam search	*"Plötzlich war ich im* National Theatre und das konnte ich nicht ganz glauben.
	+ prefix tuning	*"Plötzlich war ich im* National Theatre, und ich konnte es einfach nicht glauben.
2.	**source**	"A little voice inside me said, 'You're going to have to do 10 minutes while they fix the computer." "
	reference	*"Eine kleine Stimme sagte mir "Du musst jetzt 10 Minuten überbrücken*, während sie den Computer reparieren." "
	baseline	*"Eine kleine Stimme sagte mir "Du musst jetzt 10 Minuten überbrücken*, sie legen die müssen, während der Computer."
	target beam search	*"Eine kleine Stimme sagte mir "Du musst jetzt 10 Minuten überbrücken* zu tun, während sie den Computer reparieren".
	+ prefix tuning	*"Eine kleine Stimme sagte mir "Du musst jetzt 10 Minuten überbrücken*, während sie den Computer reparieren." "
3.	**source**	Yemeni media report that there is traffic chaos in the capital.
	reference	*Jemenitische Medien berichten von einem Verkehrschaos* in der Hauptstadt.
	baseline	*Jemenitische Medien berichten von einem Verkehrschaos* ist der Verkehr in der Hauptstadt.
	target beam search	*Jemenitische Medien berichten von einem Verkehrschaos* gibt es in der Hauptstadt.
	+ prefix tuning	*Jemenitische Medien berichten von einem Verkehrschaos* in der Hauptstadt.

Table 3: Translation examples from the English-German *newstest2015* test set. We compare the prefix decoding output of the baseline against target beam search both with and without prefix tuning. The prefix is printed in *italics*.

		English-French				**English-German**			
		autodesk		**newstest2014**		**autodesk**		**newstest2015**	
		WPA	KSR	WPA	KSR	WPA	KSR	WPA	KSR
baseline	1-best	41.1	57.8	38.0	61.7	37.8	64.7	28.5	72.7
	10-best	48.6	53.3	42.7	58.5	43.9	60.2	33.4	69.5
target beam search	1-best	50.3	48.2	50.9	50.5	46.0	57.2	41.2	63.7
	10-best	56.8	43.7	54.9	47.3	51.1	53.2	46.6	60.3
	10-best diverse	64.5	39.1	66.2	41.4	57.3	48.4	55.5	54.5

Table 4: Oracle results on the English-French and English-German tasks. We compare the single best result with oracle scores on 10-best lists with standard and diverse n-best extraction on both target beam search with prefix tuning and the phrase-based baseline system.

	autodesk				**newstest2015**			
English-German	BLEU	pxBLEU	WPA	secs / segment	BLEU	pxBLEU	WPA	secs / segment
target beam search	44.5	62.2	46.0	0.051	22.4	36.0	41.2	0.089
10-best diverse		65.1	57.3			39.5	55.5	
NMT single	40.6	61.2	52.3	1.6	23.2	39.2	50.4	1.3
NMT ensemble	44.3	64.7	54.9	7.7	26.3	42.1	53.0	10.0

Table 5: English-German results for the phrase-based system with target beam search and tuned to a combined metric, compared with the recurrent neural translation system. The *10-best diverse* line contains oracle scores from a 10-best list; all other scores are computed for a single suffix prediction per example. We also report unconstrained full-sentence BLEU scores. The phrase-based timing results include prefix alignment and synthetic phrase extraction.

be tailored to the task. The combination of these changes can adapt a phrase-based translation system to perform prefix alignment and suffix prediction jointly with fewer search errors and greater accuracy for the critical first words of the suffix. In light of the dramatic improvements in prediction quality that result from the techniques we have described, we look forward to investigating the effect on user experience for interactive translation systems that employ these methods.

1. **source**	He is due to appear in Karratha Magistrates Court on September 23.
reference	*Er* soll am 23. September vor dem Amtsgericht in Karratha erscheinen.
phrase-based	*Er* ist aufgrund der in Karratha Magistrates Court am 23. September.
NMT	*Er* wird am 23. September in Karratah Magistrates Court erscheinen.
2. **source**	The research, funded by the [...], will be published today in the Medical Journal of Australia.
reference	*Die von [...] finanzierte Studie wird heute im Medical Journal of Australia* veröffentlicht.
phrase-based	*Die von [...] finanzierte Studie wird heute im Medical Journal of Australia*.
NMT	*Die von [...] finanzierte Studie wird heute im Medical Journal of Australia* veröffentlicht.
3. **source**	But it is certainly not a radical initiative - at least by American standards.
reference	*Aber es ist* mit Sicherheit keine radikale Initiative - jedenfalls nicht nach amerikanischen Standards.
phrase-based	*Aber es ist* sicherlich kein radikale Initiative - zumindest von den amerikanischen Standards.
NMT	*Aber es ist* gewiss keine radikale Initiative - zumindest nicht nach amerikanischem Maßstab.
4. **source**	Now everyone knows that the labor movement did not diminish the strength of the nation but enlarged it.
reference	*Jetzt wissen alle, dass die Arbeiterbewegung die Stärke der Nation nicht* einschränkte, sondern sie vergrößerte.
phrase-based	*Jetzt wissen alle, dass die Arbeiterbewegung die Stärke der Nation nicht* schmälern, aber vergrößert .
NMT	*Jetzt wissen alle, dass die Arbeiterbewegung die Stärke der Nation nicht* verringert, sondern erweitert hat.
5. **source**	"As go unions, so go middle-class jobs," says Ellison, the Minnesota Democrat who serves as a Congressional Progressive Caucus co-chair.
reference	*"So wie Gewerkschaften sterben*, sterben auch die Mittelklassejobs," sagte Ellison, ein Demokrat aus Minnesota und stellvertretender Vorsitzender des Progressive Caucus im Kongress.
phrase-based	*"So wie Gewerkschaften sterben*, so Mittelklasse-Jobs", sagt Ellison, der Minnesota Demokrat, dient als Congressional Progressive Caucus Mitveranstalter.
NMT	*"So wie Gewerkschaften sterben*, so gehen die gehen," sagt Ellison, der Liberalen, der als Kongresses des eine dient.
6. **source**	The opposition politician, Imran Khan, accuses Prime Minister Sharif of rigging the parliamentary elections, which took place in May last year.
reference	*Der Oppositionspolitiker Imran Khan wirft Premier Sharif vor, bei der Parlamentswahl im Mai vergangenen Jahres betrogen zu haben.*
phrase-based	*Der Oppositionspolitiker Imran Khan wirft Premier Sharif vor, bei der Parlamentswahl im Mai vergangenen Jahres betrogen zu haben.* , die
NMT	*Der Oppositionspolitiker Imran Khan wirft Premier Sharif vor, bei der Parlamentswahl im Mai vergangenen Jahres betrogen zu haben.*

Table 6: Example sentences from the English-German *newstest2015* test set. We compare the prefix decoding output of phrase-based target beam search against the single network neural machine translation (NMT) engine, printing the prefix in *italics*. The examples illustrate the four error categories *missing verb* (Ex. 1 and 2), *grammar / morphology* (Ex. 3 and 4), *missing content words* (Ex. 5) and *alignment* (Ex. 6).

#better	**phrase-based**	**NMT**
missing verb	1	19
grammar / morphology	0	15
missing content words	17	3
alignment	0	6

Table 7: Result of the manual analysis on the first 100 segments of the English-German *newstest2015* test set. For each of the four error categories we count how many times one of the systems produced a better output.

Acknowledgments

Minh-Thang Luong was partially supported by NSF Award IIS-1514268 and partially supported by a gift from Bloomberg L.P.

References

Sergio Barrachina, Oliver Bender, Francisco Casacuberta, Jorge Civera, Elsa Cubel, Shahram Khadivi, et al. 2008. Statistical approaches to computer-assisted translation. *Computational Linguistics*, 35(1):3–28.

Oliver Bender, Saša Hasan, David Vilar, Richard Zens, and Hermann Ney. 2005. Comparison of generation strategies for interactive machine translation. In *EAMT*.

Arendse Bernth and Michael C. McCord. 2000. The effect of source analysis on translation confidence. In *AMTA*.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, et al. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *WMT*.

Peter F. Brown, Stephan A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The

Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Christian Buck, Kenneth Heafield, and Bas van Ooyen. 2014. N-gram counts and language models from the common crawl. In *LREC*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *EMNLP*.

Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

José Esteban, José Lorenzo, Antonio S. Valderrábanos, and Guy Lapalme. 2004. TransType2 - an innovative computer-assisted translation system. In *ACL*.

George Foster, Pierre Isabelle, and Pierre Plamondon. 1997. Target-Text Mediated Interactive Machine Translation. *Machine Translation*, 12(1–2):175–194.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *EMNLP*.

Spence Green, Sida Wang, Daniel Cer, and Christopher D. Manning. 2013. Fast and adaptive online training of feature-rich translation models. In *ACL*.

Spence Green, Daniel Cer, and Christopher D. Manning. 2014. Phrasal: A toolkit for new directions in statistical machine translation. In *WMT*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL*.

Liang Huang and David Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *ACL*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL*.

Philipp Koehn, Chara Tsoukala, and Herve Saint-Amand. 2014. Refinements to interactive translation prediction based on search graphs. In *ACL*.

Philippe Langlais, George Foster, and Guy Lapalme. 2000. TransType: a Computer-Aided Translation Typing System. In *NAACL Workshop on Embedded Machine Translation Systems*.

Abby Levenberg, Chris Callison-Burch, and Miles Osborne. 2010. Stream-based translation models for statistical machine translation. In *NAACL*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Wolfgang Macherey, Franz Josef Och, Ignacio Thayer, and Jakop Uszkoreit. 2008. Lattice-based minimum error rate training for statistical machine translation. In *EMNLP*.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural reranking improves subjective quality of machine translation: NAIST at WAT2015. In *2nd Workshop on Asian Translation (WAT2015)*.

Franz Josef Och and Hermann Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–450.

Franz Josef Och, Christoph Tillmann, and Hermann Ney. 1999. Improved alignment models for statistical machine translation. In *EMNLP*.

Franz Josef Och, Richard Zens, and Hermann Ney. 2003. Efficient search for interactive statistical machine translation. In *EACL*.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2009. Interactive machine translation based on partial statistical phrase-based alignments. In *RANLP*.

Daniel Ortiz-Martínez, Ismael García-Varea, and Francisco Casacuberta. 2010. Online learning for interactive statistical machine translation. In *NAACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *ACL*.

Germán Sanchis-Trilles, Daniel Ortiz-Martínez, Jorge Civera, Francisco Casacuberta, Enrique Vidal, and Hieu Hoang. 2008. Improving interactive machine translation via mouse actions. In *EMNLP*.

Joost Schilperoord. 1996. *It's about Time: Temporal Aspects of Cognitive Processes in Text Production*. Rodopi.

Raivis Skadiņš, Jörg Tiedemann, Roberts Rozis, and Daiga Deksne. 2014. Billions of parallel words for free: Building and using the EU bookshop corpus. In *LREC*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Joern Wuebker, Spence Green, and John DeNero. 2015. Hierarchical incremental adaptation for statistical machine translation. In *EMNLP*.

Modeling Coverage for Neural Machine Translation

Zhaopeng Tu[†] **Zhengdong Lu**[†] **Yang Liu**[‡] **Xiaohua Liu**[†] **Hang Li**[†]

[†]Noah's Ark Lab, Huawei Technologies, Hong Kong
{tu.zhaopeng,lu.zhengdong,liuxiaohua3,hangli.hl}@huawei.com
[‡]Department of Computer Science and Technology, Tsinghua University, Beijing
liuyang2011@tsinghua.edu.cn

Abstract

Attention mechanism has enhanced state-of-the-art Neural Machine Translation (NMT) by jointly learning to align and translate. It tends to ignore past alignment information, however, which often leads to over-translation and under-translation. To address this problem, we propose coverage-based NMT in this paper. We maintain a coverage vector to keep track of the attention history. The coverage vector is fed to the attention model to help adjust future attention, which lets NMT system to consider more about untranslated source words. Experiments show that the proposed approach significantly improves both translation quality and alignment quality over standard attention-based NMT.[1]

1 Introduction

The past several years have witnessed the rapid progress of end-to-end Neural Machine Translation (NMT) (Sutskever et al., 2014; Bahdanau et al., 2015). Unlike conventional Statistical Machine Translation (SMT) (Koehn et al., 2003; Chiang, 2007), NMT uses a single and large neural network to model the entire translation process. It enjoys the following advantages. First, the use of distributed representations of words can alleviate the curse of dimensionality (Bengio et al., 2003). Second, there is no need to explicitly design features to capture translation regularities, which is quite difficult in SMT. Instead, NMT is capable of learning representations directly from the training data. Third, Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) enables NMT to capture long-distance reordering, which is a significant challenge in SMT.

NMT has a serious problem, however, namely lack of *coverage*. In phrase-based SMT (Koehn et al., 2003), a decoder maintains a coverage vector to indicate whether a source word is translated or not. This is important for ensuring that each source word is translated in decoding. The decoding process is completed when all source words are "covered" or translated. In NMT, there is no such coverage vector and the decoding process ends only when the end-of-sentence mark is produced. We believe that lacking coverage might result in the following problems in conventional NMT:

1. Over-translation: some words are unnecessarily translated for multiple times;

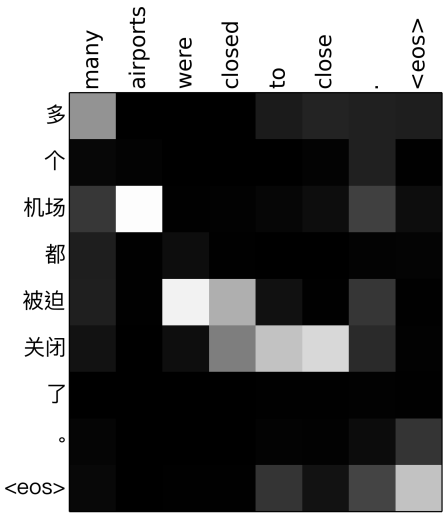
2. Under-translation: some words are mistakenly untranslated.

Specifically, in the state-of-the-art attention-based NMT model (Bahdanau et al., 2015), generating a target word heavily depends on the relevant parts of the source sentence, and a source word is involved in generation of all target words. As a result, over-translation and under-translation inevitably happen because of ignoring the "coverage" of source words (i.e., number of times a source word is translated to a target word). Figure 1(a) shows an example: the Chinese word "*guānbì*" is over translated to "*close*(*d*)" twice, while "*bèipò*" (means "*be forced to*") is mistakenly untranslated.

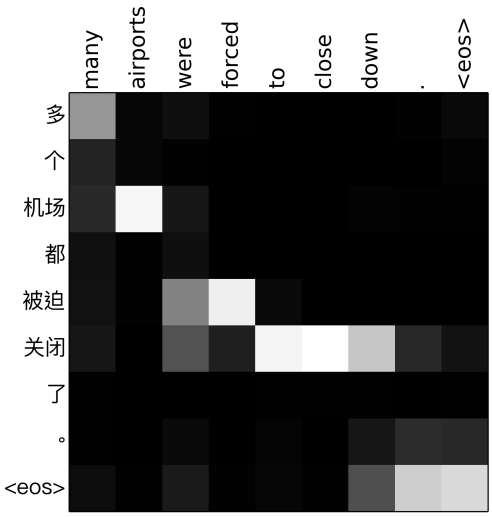
In this work, we propose a coverage mechanism to NMT (NMT-COVERAGE) to alleviate the over-translation and under-translation problems. Basically, we append a coverage vector to the intermediate representations of an NMT model, which are sequentially updated after each attentive read

[1]Our code is publicly available at https://github.com/tuzhaopeng/NMT-Coverage.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 76–85,
Berlin, Germany, August 7-12, 2016.



(a) Over-translation and under-translation generated by NMT.



(b) Coverage model alleviates the problems of over-translation and under-translation.

Figure 1: Example translations of (a) NMT without coverage, and (b) NMT with coverage. In conventional NMT without coverage, the Chinese word "*guānbì*" is over translated to "*close*(*d*)" twice, while "*bèipò*" (means "*be forced to*") is mistakenly untranslated. Coverage model alleviates these problems by tracking the "coverage" of source words.

during the decoding process, to keep track of the attention history. The coverage vector, when entering into attention model, can help adjust the future attention and significantly improve the overall alignment between the source and target sentences. This design contains many particular cases for coverage modeling with contrasting characteristics, which all share a clear linguistic intuition and yet can be trained in a data driven fashion. Notably, we achieve significant improvement even by simply using the sum of previous alignment probabilities as coverage for each word, as a successful example of incorporating linguistic knowledge into neural network based NLP models.

Experiments show that NMT-COVERAGE significantly outperforms conventional attention-based NMT on both translation and alignment tasks. Figure 1(b) shows an example, in which NMT-COVERAGE alleviates the over-translation and under-translation problems that NMT without coverage suffers from.

2 Background

Our work is built on attention-based NMT (Bahdanau et al., 2015), which simultaneously conducts dynamic alignment and generation of the target sentence, as illustrated in Figure 2. It

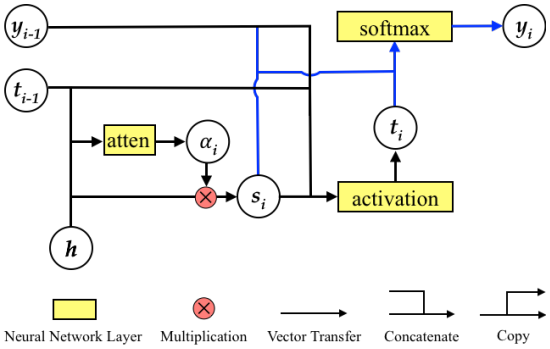


Figure 2: Architecture of attention-based NMT. Whenever possible, we omit the source index j to make the illustration less cluttered.

produces the translation by generating one target word y_i at each time step. Given an input sentence $\mathbf{x} = \{x_1, \ldots, x_J\}$ and previously generated words $\{y_1, \ldots, y_{i-1}\}$, the probability of generating next word y_i is

$$P(y_i|y_{<i}, \mathbf{x}) = softmax\big(g(y_{i-1}, \mathbf{t}_i, \mathbf{s}_i)\big) \quad (1)$$

where g is a non-linear function, and $\mathbf{t}_i$ is a decoding state for time step i, computed by

$$\mathbf{t}_i = f(\mathbf{t}_{i-1}, y_{i-1}, \mathbf{s}_i) \quad (2)$$

Here the activation function $f(\cdot)$ is a Gated Recurrent Unit (GRU) (Cho et al., 2014b), and $\mathbf{s}_i$ is

a distinct source representation for time i, calculated as a weighted sum of the source annotations:

$$\mathbf{s}_i = \sum_{j=1}^{J} \alpha_{i,j} \cdot \mathbf{h}_j \quad (3)$$

where $\mathbf{h}_j = [\overrightarrow{h}_j^\top; \overleftarrow{h}_j^\top]^\top$ is the annotation of x_j from a bi-directional Recurrent Neural Network (RNN) (Schuster and Paliwal, 1997), and its weight $\alpha_{i,j}$ is computed by

$$\alpha_{i,j} = \frac{\exp(e_{i,j})}{\sum_{k=1}^{J} \exp(e_{i,k})} \quad (4)$$

and

$$\begin{aligned} e_{i,j} &= a(\mathbf{t}_{i-1}, \mathbf{h}_j) \\ &= v_a^\top \tanh(W_a \mathbf{t}_{i-1} + U_a \mathbf{h}_j) \end{aligned} \quad (5)$$

is an *attention model* that scores how well y_i and $\mathbf{h}_j$ match. With the attention model, it avoids the need to represent the entire source sentence with a single vector. Instead, the decoder selects parts of the source sentence to pay attention to, thus exploits an *expected annotation* $\mathbf{s}_i$ over possible alignments $\alpha_{i,j}$ for each time step i.

However, the attention model fails to take advantage of past alignment information, which is found useful to avoid over-translation and under-translation problems in conventional SMT (Koehn et al., 2003). For example, if a source word is translated in the past, it is less likely to be translated again and should be assigned a lower alignment probability.

3 Coverage Model for NMT

In SMT, a coverage set is maintained to keep track of which source words have been translated ("covered") in the past. Let us take $\mathbf{x} = \{x_1, x_2, x_3, x_4\}$ as an example of input sentence. The initial coverage set is $\mathcal{C} = \{0, 0, 0, 0\}$ which denotes that no source word is yet translated. When a translation rule $bp = (x_2 x_3, y_m y_{m+1})$ is applied, we produce one hypothesis labelled with coverage $\mathcal{C} = \{0, 1, 1, 0\}$. It means that the second and third source words are translated. The goal is to generate translation with full coverage $\mathcal{C} = \{1, 1, 1, 1\}$. A source word is translated when it is covered by one translation rule, and it is not allowed to be translated again in the future (i.e., *hard coverage*). In this way, each source word is guaranteed to be translated and only be translated once. As shown, coverage is essential for SMT since it avoids gaps and overlaps in translation of source words.

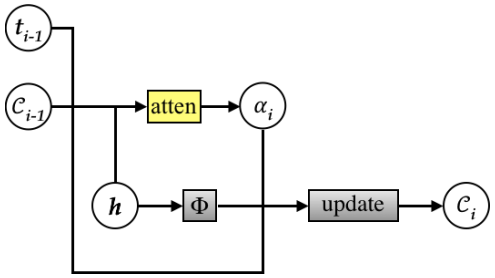


Figure 3: Architecture of coverage-based attention model. A coverage vector $\mathcal{C}_{i-1}$ is maintained to keep track of which source words have been translated before time i. Alignment decisions α_i are made jointly taking into account past alignment information embedded in $\mathcal{C}_{i-1}$, which lets the attention model to consider more about untranslated source words.

Modeling coverage is also important for attention-based NMT models, since they generally lack a mechanism to indicate whether a certain source word has been translated, and therefore are prone to the "coverage" mistakes: some parts of source sentence have been translated more than once or not translated. For NMT models, directly modeling coverage is less straightforward, but the problem can be significantly alleviated by keeping track of the attention signal during the decoding process. The most natural way for doing that would be to append a coverage vector to the annotation of each source word (i.e., $\mathbf{h}_j$), which is initialized as a zero vector but updated after every attentive read of the corresponding annotation. The coverage vector is fed to the attention model to help adjust future attention, which lets NMT system to consider more about untranslated source words, as illustrated in Figure 3.

3.1 Coverage Model

Since the coverage vector summarizes the attention record for $\mathbf{h}_j$ (and therefore for a small neighbor centering at the j^{th} source word), it will discourage further attention to it if it has been heavily attended, and implicitly push the attention to the less attended segments of the source sentence since the attention weights are normalized to one. This can potentially solve both coverage mistakes mentioned above, when modeled and learned properly.

Formally, the coverage model is given by

$$\mathcal{C}_{i,j} = g_{update}\big(\mathcal{C}_{i-1,j}, \alpha_{i,j}, \Phi(\mathbf{h}_j), \Psi\big) \quad (6)$$

where

- $g_{update}(\cdot)$ is the function that updates $\mathcal{C}_{i,j}$ after the new attention $\alpha_{i,j}$ at time step i in the decoding process;
- $\mathcal{C}_{i,j}$ is a d-dimensional coverage vector summarizing the history of attention till time step i on $\mathbf{h}_j$;
- $\Phi(\mathbf{h}_j)$ is a word-specific feature with its own parameters;
- Ψ are auxiliary inputs exploited in different sorts of coverage models.

Equation 6 gives a rather general model, which could take different function forms for $g_{update}(\cdot)$ and $\Phi(\cdot)$, and different auxiliary inputs Ψ (e.g., previous decoding state $\mathbf{t}_{i-1}$). In the rest of this section, we will give a number of representative implementations of the coverage model, which either leverage more linguistic information (Section 3.1.1) or resort to the flexibility of neural network approximation (Section 3.1.2).

3.1.1 Linguistic Coverage Model

We first consider at linguistically inspired model which has a small number of parameters, as well as clear interpretation. While the linguistically-inspired coverage in NMT is similar to that in SMT, there is one key difference: it indicates what percentage of source words have been translated (i.e., *soft coverage*). In NMT, each target word y_i is generated from all source words with probability $\alpha_{i,j}$ for source word x_j. In other words, the source word x_j is involved in generating all target words and the probability of generating target word y_i at time step i is $\alpha_{i,j}$. Note that unlike in SMT in which each source word is *fully translated* at one decoding step, the source word x_j is *partially translated* at each decoding step in NMT. Therefore, the coverage at time step i denotes the translated ratio of that each source word is translated.

We use a scalar ($d = 1$) to represent linguistic coverage for each source word and employ an accumulate operation for g_{update}. The initial value of linguistic coverage is zero, which denotes that the corresponding source word is not translated yet. We iteratively construct linguistic coverages through accumulation of alignment probabilities generated by the attention model, each of which is normalized by a distinct context-dependent weight. The coverage of source word x_j at time step i is computed by

$$\mathcal{C}_{i,j} = \mathcal{C}_{i-1,j} + \frac{1}{\Phi_j}\alpha_{i,j} = \frac{1}{\Phi_j}\sum_{k=1}^{i}\alpha_{k,j} \quad (7)$$

where Φ_j is a pre-defined weight which indicates the number of target words x_j is expected to generate. The simplest way is to follow Xu et al. (2015) in image-to-caption translation to fix $\Phi = 1$ for all source words, which means that we directly use the sum of previous alignment probabilities without normalization as coverage for each word, as done in (Cohn et al., 2016).

However, in machine translation, different types of source words may contribute differently to the generation of target sentence. Let us take the sentence pairs in Figure 1 as an example. The noun in the source sentence "*jīchǎng*" is translated into one target word "*airports*", while the adjective "*bèipò*" is translated into three words "*were forced to*". Therefore, we need to assign a distinct Φ_j for each source word. Ideally, we expect $\Phi_j = \sum_{i=1}^{I}\alpha_{i,j}$ with I being the total number of time steps in decoding. However, such desired value is not available before decoding, thus is not suitable in this scenario.

Fertility To predict Φ_j, we introduce the concept of *fertility*, which is firstly proposed in word-level SMT (Brown et al., 1993). Fertility of source word x_j tells how many target words x_j produces. In SMT, the fertility is a random variable Φ_j, whose distribution $p(\Phi_j = \phi)$ is determined by the parameters of word alignment models (e.g., IBM models). In this work, we simplify and adapt fertility from the original model and compute the fertility Φ_j by[2]

$$\Phi_j = \mathcal{N}(x_j|\mathbf{x}) = N \cdot \sigma(U_f\mathbf{h}_j) \quad (8)$$

where $N \in \mathbb{R}$ is a predefined constant to denote the maximum number of target words one source

[2]Fertility in SMT is a random variable with a set of fertility probabilities, $n(\Phi_j|x_j) = p(\Phi_{<j}, \mathbf{x})$, which depends on the fertilities of previous source words. To simplify the calculation and adapt it to the attention model in NMT, we define the fertility in NMT as a constant number, which is independent of previous fertilities.

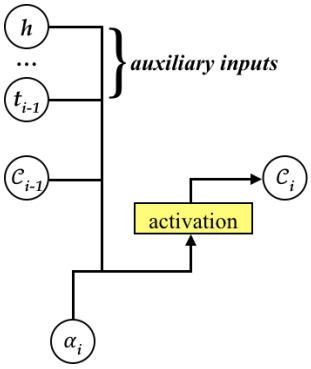


Figure 4: NN-based coverage model.

word can produce, $\sigma(\cdot)$ is a logistic sigmoid function, and $U_f \in \mathbb{R}^{1\times 2n}$ is the weight matrix. Here we use $\mathbf{h}_j$ to denote $(x_j|\mathbf{x})$ since $\mathbf{h}_j$ contains information about the whole input sentence with a strong focus on the parts surrounding x_j (Bahdanau et al., 2015). Since Φ_j does not depend on i, we can pre-compute it before decoding to minimize the computational cost.

3.1.2 Neural Network Based Coverage Model

We next consider Neural Network (NN) based coverage model. When $\mathcal{C}_{i,j}$ is a vector ($d > 1$) and $g_{update}(\cdot)$ is a neural network, we actually have an RNN model for coverage, as illustrated in Figure 4. In this work, we take the following form:

$$\mathcal{C}_{i,j} = f(\mathcal{C}_{i-1,j}, \alpha_{i,j}, \mathbf{h}_j, \mathbf{t}_{i-1})$$

where $f(\cdot)$ is a nonlinear activation function and $\mathbf{t}_{i-1}$ is the auxiliary input that encodes past translation information. Note that we leave out the word-specific feature function $\Phi(\cdot)$ and only take the input annotation $\mathbf{h}_j$ as the input to the coverage RNN. It is important to emphasize that the NN-based coverage model is able to be fed with arbitrary inputs, such as the previous attentional context $\mathbf{s}_{i-1}$. Here we only employ $\mathcal{C}_{i-1,j}$ for past alignment information, $\mathbf{t}_{i-1}$ for past translation information, and $\mathbf{h}_j$ for word-specific bias.[3]

Gating The neural function $f(\cdot)$ can be either a simple activation function $\tanh$ or a gating function that proves useful to capture long-distance dependencies. In this work, we adopt GRU for the gating activation since it is simple yet powerful (Chung et al., 2014). Please refer to (Cho et al., 2014b) for more details about GRU.

Discussion Intuitively, the two types of models summarize coverage information in "different languages". Linguistic models summarize coverage information in human language, which has a clear interpretation to humans. Neural models encode coverage information in "neural language", which can be "understood" by neural networks and let them to decide how to make use of the encoded coverage information.

3.2 Integrating Coverage into NMT

Although attention based model has the capability of jointly making alignment and translation, it does not take into consideration translation history. Specifically, a source word that has significantly contributed to the generation of target words in the past, should be assigned lower alignment probabilities, which may not be the case in attention based NMT. To address this problem, we propose to calculate the alignment probabilities by incorporating past alignment information embedded in the coverage model.

Intuitively, at each time step i in the decoding phase, coverage from time step $(i-1)$ serves as an additional input to the attention model, which provides complementary information of that how likely the source words are translated in the past. We expect the coverage information would guide the attention model to focus more on untranslated source words (i.e., assign higher alignment probabilities). In practice, we find that the coverage model does fulfill the expectation (see Section 5). The translated ratios of source words from linguistic coverages negatively correlate to the corresponding alignment probabilities.

More formally, we rewrite the attention model in Equation 5 as

$$\begin{aligned} e_{i,j} &= a(\mathbf{t}_{i-1}, \mathbf{h}_j, \mathcal{C}_{i-1,j}) \\ &= v_a^\top \tanh(W_a \mathbf{t}_{i-1} + U_a \mathbf{h}_j + V_a \mathcal{C}_{i-1,j}) \end{aligned}$$

where $\mathcal{C}_{i-1,j}$ is the coverage of source word x_j before time i. $V_a \in \mathbb{R}^{n\times d}$ is the weight matrix for coverage with n and d being the numbers of hidden units and coverage units, respectively.

[3] In our preliminary experiments, considering more inputs (e.g., current and previous attentional contexts, unnormalized attention weights $e_{i,j}$) does not always lead to better translation quality. Possible reasons include: 1) the inputs contains duplicate information, and 2) more inputs introduce more back-propagation paths and therefore make it difficult to train. In our experience, one principle is to only feed the coverage model inputs that contain distinct information, which are complementary to each other.

4 Training

We take end-to-end learning for the NMT-COVERAGE model, which learns not only the parameters for the "original" NMT (i.e., θ for encoding RNN, decoding RNN, and attention model) but also the parameters for coverage modeling (i.e., η for annotation and guidance of attention) . More specifically, we choose to maximize the likelihood of reference sentences as most other NMT models (see, however (Shen et al., 2016)):

$$(\theta^*, \eta^*) = \arg\max_{\theta,\eta} \sum_{n=1}^{N} \log P(\mathbf{y}_n|\mathbf{x}_n; \theta, \eta) \quad (9)$$

No auxiliary objective For the coverage model with a clearer linguistic interpretation (Section 3.1.1), it is possible to inject an auxiliary objective function on some intermediate representation. More specifically, we may have the following objective:

$$(\theta^*, \eta^*) = \arg\max_{\theta,\eta} \sum_{n=1}^{N} \Bigg\{ \log P(\mathbf{y}_n|\mathbf{x}_n; \theta, \eta) - \lambda \Big\{ \sum_{j=1}^{J} (\Phi_j - \sum_{i=1}^{I} \alpha_{i,j})^2; \eta \Big\} \Bigg\}$$

where the term $\{ \sum_{j=1}^{J} (\Phi_j - \sum_{i=1}^{I} \alpha_{i,j})^2; \eta \}$ penalizes the discrepancy between the sum of alignment probabilities and the expected fertility for linguistic coverage. This is similar to the more explicit training for fertility as in Xu et al. (2015), which encourages the model to pay equal attention to every part of the image (i.e., $\Phi_j = 1$). However, our empirical study shows that the combined objective consistently worsens the translation quality while slightly improves the alignment quality.

Our training strategy poses less constraints on the dependency between Φ_j and the attention than a more explicit strategy taken in (Xu et al., 2015). We let the objective associated with the translation quality (i.e., the likelihood) to drive the training, as in Equation 9. This strategy is arguably advantageous, since the attention weight on a hidden state $\mathbf{h}_j$ cannot be interpreted as the proportion of the corresponding word being translated in the target sentence. For one thing, the hidden state $\mathbf{h}_j$, after the transformation from encoding RNN, bears the contextual information from other parts of the source sentence, and thus loses the rigid correspondence with the corresponding word. Therefore, penalizing the discrepancy between the sum of alignment probabilities and the expected fertility does not hold in this scenario.

5 Experiments

5.1 Setup

We carry out experiments on a Chinese-English translation task. Our training data for the translation task consists of 1.25M sentence pairs extracted from LDC corpora[4] , with 27.9M Chinese words and 34.5M English words respectively. We choose NIST 2002 dataset as our development set, and the NIST 2005, 2006 and 2008 datasets as our test sets. We carry out experiments of the alignment task on the evaluation dataset from (Liu and Sun, 2015), which contains 900 manually aligned Chinese-English sentence pairs. We use the case-insensitive 4-gram NIST BLEU score (Papineni et al., 2002) for the translation task, and the alignment error rate (AER) (Och and Ney, 2003) for the alignment task. To better estimate the quality of the soft alignment probabilities generated by NMT, we propose a variant of AER, naming *SAER*:

$$SAER = 1 - \frac{|M_A \times M_S| + |M_A \times M_P|}{|M_A| + |M_S|}$$

where A is a candidate alignment, and S and P are the sets of sure and possible links in the reference alignment respectively ($S \subseteq P$). M denotes alignment matrix, and for both M_S and M_P we assign the elements that correspond to the existing links in S and P with probabilities 1 while assign the other elements with probabilities 0. In this way, we are able to better evaluate the quality of the soft alignments produced by attention-based NMT. We use *sign-test* (Collins et al., 2005) for statistical significance test.

For efficient training of the neural networks, we limit the source and target vocabularies to the most frequent 30K words in Chinese and English, covering approximately 97.7% and 99.3% of the two corpora respectively. All the out-of-vocabulary words are mapped to a special token UNK. We set $N = 2$ for the fertility model in the linguistic coverages. We train each model with the sentences of length up to 80 words in the training data. The word embedding dimension is 620 and the size of a hidden layer is 1000. All the other settings are the same as in (Bahdanau et al., 2015).

[4]The corpora include LDC2002E18, LDC2003E07, LDC2003E14, Hansards portion of LDC2004T07, LDC2004T08 and LDC2005T06.

#	System	#Params	MT05	MT06	MT08	Avg.
1	Moses	–	31.37	30.85	23.01	28.41
2	GroundHog	84.3M	30.61	31.12	23.23	28.32
3	+ Linguistic coverage w/o fertility	+1K	31.26†	32.16†‡	24.84†‡	29.42
4	+ Linguistic coverage w/ fertility	+3K	32.36†‡	32.31†‡	24.91†‡	29.86
5	+ NN-based coverage w/o gating ($d = 1$)	+4K	31.94†‡	32.11†‡	23.31	29.12
6	+ NN-based coverage w/ gating ($d = 1$)	+10K	31.94†‡	32.16†‡	24.67†‡	29.59
7	+ NN-based coverage w/ gating ($d = 10$)	+100K	**32.73**†‡	**32.47**†‡	**25.23**†‡	**30.14**

Table 1: Evaluation of translation quality. d denotes the dimension of NN-based coverages, and † and ‡ indicate statistically significant difference ($p < 0.01$) from GroundHog and Moses, respectively. "+" is on top of the baseline system GroundHog.

We compare our method with two state-of-the-art models of SMT and NMT[5]:

- **Moses** (Koehn et al., 2007): an open source phrase-based translation system with default configuration and a 4-gram language model trained on the target portion of training data.
- **GroundHog** (Bahdanau et al., 2015): an attention-based NMT system.

5.2 Translation Quality

Table 1 shows the translation performances measured in BLEU score. Clearly the proposed NMT-COVERAGE significantly improves the translation quality in all cases, although there are still considerable differences among different variants.

Parameters Coverage model introduces few parameters. The baseline model (i.e., GroundHog) has 84.3M parameters. The linguistic coverage using fertility introduces 3K parameters (2K for fertility model), and the NN-based coverage with gating introduces 10K$\times d$ parameters (6K$\times d$ for gating), where d is the dimension of the coverage vector. In this work, the most complex coverage model only introduces 0.1M additional parameters, which is quite small compared to the number of parameters in the existing model (i.e., 84.3M).

Speed Introducing the coverage model slows down the training speed, but not significantly. When running on a single GPU device Tesla K80, the speed of the baseline model is 960 target words per second. System 4 ("+Linguistic coverage with fertility") has a speed of 870 words per second, while System 7 ("+NN-based coverage (d=10)") achieves a speed of 800 words per second.

[5]There are recent progress on aggregating multiple models or enlarging the vocabulary(e.g., in (Jean et al., 2015)), but here we focus on the generic models.

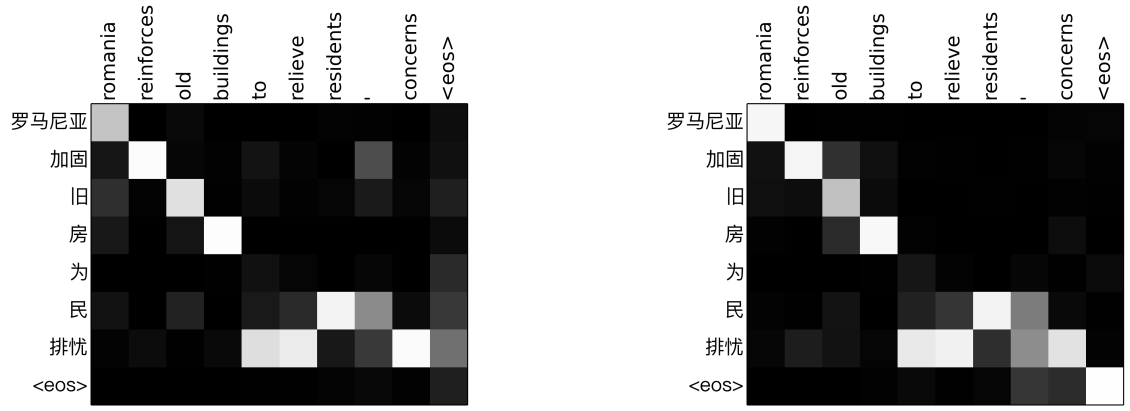
Linguistic Coverages (Rows 3 and 4): Two observations can be made. First, the simplest linguistic coverage (Row 3) already significantly improves translation performance by 1.1 BLEU points, indicating that coverage information is very important to the attention model. Second, incorporating fertility model boosts the performance by better estimating the covered ratios of source words.

NN-based Coverages (Rows 5-7): (1) *Gating* (Rows 5 and 6): Both variants of NN-based coverages outperform GroundHog with averaged gains of 0.8 and 1.3 BLEU points, respectively. Introducing gating activation function improves the performance of coverage models, which is consistent with the results in other tasks (Chung et al., 2014). (2) *Coverage dimensions* (Rows 6 and 7): Increasing the dimension of coverage models further improves the translation performance by 0.6 point in BLEU score, at the cost of introducing more parameters (e.g., from 10K to 100K).[6]

5.3 Alignment Quality

Table 2 lists the alignment performances. We find that coverage information improves attention model as expected by maintaining an annotation summarizing attention history on each source word. More specifically, linguistic coverage with fertility significantly reduces alignment errors under both metrics, in which fertility plays an important role. NN-based coverages, however, does not significantly reduce alignment errors until increasing the coverage dimension from 1 to 10. It indicates that NN-based models need slightly more

[6]In a pilot study, further increasing the coverage dimension only slightly improved the translation performance. One possible reason is that encoding the relatively simple coverage information does not require too many dimensions.



(a) Groundhog

(b) + NN cov. w/ gating ($d = 10$)

Figure 5: Example alignments. Using coverage mechanism, translated source words are less likely to contribute to generation of the target words next (e.g., top-right corner for the first four Chinese words.).

System	SAER	AER
GroundHog	67.00	54.67
+ Ling. cov. w/o fertility	66.75	53.55
+ Ling. cov. w/ fertility	64.85	52.13
+ NN cov. w/o gating ($d = 1$)	67.10	54.46
+ NN cov. w/ gating ($d = 1$)	66.30	53.51
+ NN cov. w/ gating ($d = 10$)	**64.25**	**50.50**

Table 2: Evaluation of alignment quality. The lower the score, the better the alignment quality.

dimensions to encode the coverage information.

Figure 5 shows an example. The coverage mechanism does meet the expectation: the alignments are more concentrated and most importantly, translated source words are less likely to get involved in generation of the target words next. For example, the first four Chinese words are assigned lower alignment probabilities (i.e., darker color) after the corresponding translation "*romania reinforces old buildings*" is produced.

5.4 Effects on Long Sentences

Following Bahdanau et al. (2015), we group sentences of similar lengths together and compute BLEU score and averaged length of translation for each group, as shown in Figure 6. Cho et al. (2014a) show that the performance of Groundhog drops rapidly when the length of input sentence increases. Our results confirm these findings. One main reason is that Groundhog produces much shorter translations on longer sentences (e.g., > 40, see right panel in Figure 6), and thus faces a serious under-translation problem. NMT-COVERAGE alleviates this problem by incorporating coverage information into the attention model, which in general pushes the attention to untranslated parts of the source sentence and implicitly discourages early stop of decoding. It is worthy to emphasize that both NN-based coverages (with gating, $d = 10$) and linguistic coverages (with fertility) achieve similar performances on long sentences, reconfirming our claim that the two variants improve the attention model in their own ways.

As an example, consider this source sentence in the test set:

> *qiáodān běn sàijì píngjūn défēn 24.3fēn , tā zài sān zhōu qián jiēshòu shǒushù , qiúduì zài cǐ qījiān 4 shèng 8 fù .*

Groundhog translates this sentence into:

> *jordan achieved an average score of eight weeks ahead with a surgical operation three weeks ago .*

in which the sub-sentence "*, qiúduì zài cǐ qījiān 4 shèng 8 fù*" is under-translated. With the (NN-based) coverage mechanism, NMT-COVERAGE translates it into:

> *jordan 's average score points to UNK this year . he received surgery before three weeks , with a team in the period of 4 to 8 .*

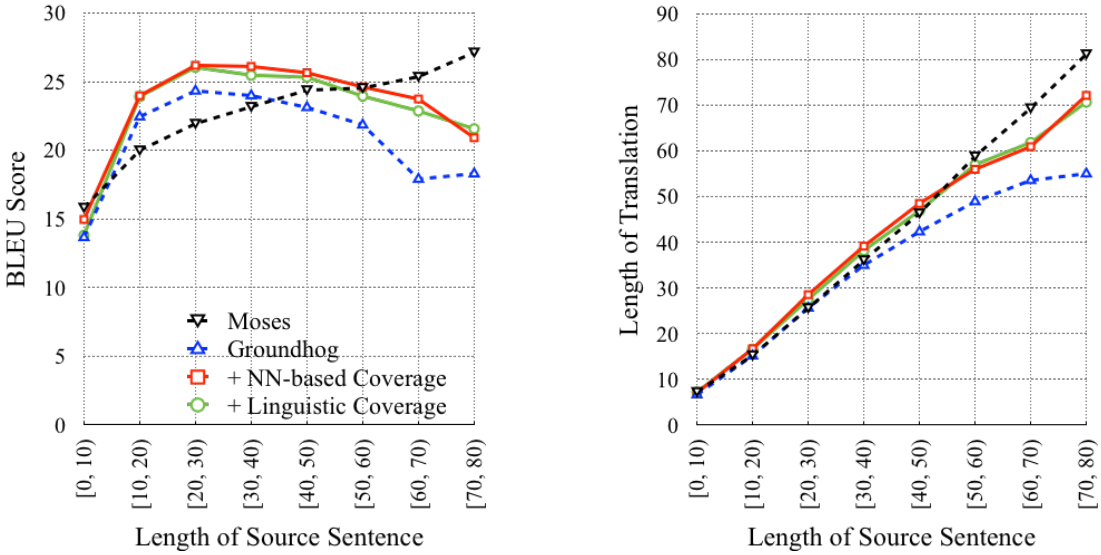


Figure 6: Performance of the generated translations with respect to the lengths of the input sentences. Coverage models alleviate under-translation by producing longer translations on long sentences.

in which the under-translation is rectified.

The quantitative and qualitative results show that the coverage models indeed help to alleviate under-translation, especially for long sentences consisting of several sub-sentences.

6 Related Work

Our work is inspired by recent works on improving attention-based NMT with techniques that have been successfully applied to SMT. Following the success of Minimum Risk Training (MRT) in SMT (Och, 2003), Shen et al. (2016) proposed MRT for end-to-end NMT to optimize model parameters directly with respect to evaluation metrics. Based on the observation that attention-based NMT only captures partial aspects of attentional regularities, Cheng et al. (2016) proposed agreement-based learning (Liang et al., 2006) to encourage bidirectional attention models to agree on parameterized alignment matrices. Along the same direction, inspired by the coverage mechanism in SMT, we propose a coverage-based approach to NMT to alleviate the over-translation and under-translation problems.

Independent from our work, Cohn et al. (2016) and Feng et al. (2016) made use of the concept of "fertility" for the attention model, which is similar in spirit to our method for building the linguistically inspired coverage with fertility. Cohn et al. (2016) introduced a feature-based fertility that includes the total alignment scores for the surrounding source words. In contrast, we make prediction of fertility before decoding, which works as a normalizer to better estimate the coverage ratio of each source word. Feng et al. (2016) used the previous attentional context to represent *implicit fertility* and passed it to the attention model, which is in essence similar to the input-feed method proposed in (Luong et al., 2015). Comparatively, we predict *explicit fertility* for each source word based on its encoding annotation, and incorporate it into the linguistic-inspired coverage for attention model.

7 Conclusion

We have presented an approach for enhancing NMT, which maintains and utilizes a coverage vector to indicate whether each source word is translated or not. By encouraging NMT to pay less attention to translated words and more attention to untranslated words, our approach alleviates the serious over-translation and under-translation problems that traditional attention-based NMT suffers from. We propose two variants of coverage models: *linguistic coverage* that leverages more linguistic information and *NN-based coverage* that resorts to the flexibility of neural network approximation . Experimental results show that both variants achieve significant improvements in terms of translation quality and alignment quality over NMT without coverage.

Acknowledgement

This work is supported by China National 973 project 2014CB340301. Yang Liu is supported by the National Natural Science Foundation of China (No. 61522204) and the 863 Program (2015AA011808). We thank the anonymous reviewers for their insightful comments.

References

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR 2015*.

[Bengio et al.2003] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR*.

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

[Cheng et al.2016] Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based Joint Training for Bidirectional Attention-based Neural Machine Translation. In *IJCAI 2016*.

[Chiang2007] David Chiang. 2007. Hierarchical phrase-based translation. *CL*.

[Cho et al.2014a] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: encoder–decoder approaches. In *SSST 2014*.

[Cho et al.2014b] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP 2014*.

[Chung et al.2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*.

[Cohn et al.2016] Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vylomova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. In *NAACL 2016*.

[Collins et al.2005] Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *ACL 2005*.

[Feng et al.2016] Shi Feng, Shujie Liu, Mu Li, and Ming Zhou. 2016. Implicit distortion and fertility models for attention-based encoder-decoder nmt model. *arXiv*.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*.

[Jean et al.2015] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *ACL 2015*.

[Koehn et al.2003] Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *NAACL 2003*.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: open source toolkit for statistical machine translation. In *ACL 2007*.

[Liang et al.2006] Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *NAACL 2006*.

[Liu and Sun2015] Yang Liu and Maosong Sun. 2015. Contrastive unsupervised word alignment with non-local features. In *AAAI 2015*.

[Luong et al.2015] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP 2015*.

[Och and Ney2003] Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

[Och2003] Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *ACL 2003*.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL 2002*.

[Schuster and Paliwal1997] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

[Shen et al.2016] Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Minimum Risk Training for Neural Machine Translation. In *ACL 2016*.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *NIPS 2014*.

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *ICML 2015*.

Improving Neural Machine Translation Models with Monolingual Data

Rico Sennrich and **Barry Haddow** and **Alexandra Birch**
School of Informatics, University of Edinburgh
{rico.sennrich,a.birch}@ed.ac.uk, bhaddow@inf.ed.ac.uk

Abstract

Neural Machine Translation (NMT) has obtained state-of-the art performance for several language pairs, while only using parallel data for training. Target-side monolingual data plays an important role in boosting fluency for phrase-based statistical machine translation, and we investigate the use of monolingual data for NMT. In contrast to previous work, which combines NMT models with separately trained language models, we note that encoder-decoder NMT architectures already have the capacity to learn the same information as a language model, and we explore strategies to train with monolingual data without changing the neural network architecture. By pairing monolingual training data with an automatic back-translation, we can treat it as additional parallel training data, and we obtain substantial improvements on the WMT 15 task English↔German (+2.8–3.7 BLEU), and for the low-resourced IWSLT 14 task Turkish→English (+2.1–3.4 BLEU), obtaining new state-of-the-art results. We also show that fine-tuning on in-domain monolingual and parallel data gives substantial improvements for the IWSLT 15 task English→German.

1 Introduction

Neural Machine Translation (NMT) has obtained state-of-the art performance for several language pairs, while only using parallel data for training. Target-side monolingual data plays an important role in boosting fluency for phrase-based statistical machine translation, and we investigate the use of monolingual data for NMT.

Language models trained on monolingual data have played a central role in statistical machine translation since the first IBM models (Brown et al., 1990). There are two major reasons for their importance. Firstly, word-based and phrase-based translation models make strong independence assumptions, with the probability of translation units estimated independently from context, and language models, by making different independence assumptions, can model how well these translation units fit together. Secondly, the amount of available monolingual data in the target language typically far exceeds the amount of parallel data, and models typically improve when trained on more data, or data more similar to the translation task.

In (attentional) encoder-decoder architectures for neural machine translation (Sutskever et al., 2014; Bahdanau et al., 2015), the decoder is essentially an RNN language model that is also conditioned on source context, so the first rationale, adding a language model to compensate for the independence assumptions of the translation model, does not apply. However, the data argument is still valid in NMT, and we expect monolingual data to be especially helpful if parallel data is sparse, or a poor fit for the translation task, for instance because of a domain mismatch.

In contrast to previous work, which integrates a separately trained RNN language model into the NMT model (Gülçehre et al., 2015), we explore strategies to include monolingual training data in the training process without changing the neural network architecture. This makes our approach applicable to different NMT architectures.

The main contributions of this paper are as follows:

- we show that we can improve the machine translation quality of NMT systems by mixing monolingual target sentences into the

The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. - Samsung R&D Institute Poland.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 86–96,
Berlin, Germany, August 7-12, 2016.

training set.

- we investigate two different methods to fill the source side of monolingual training instances: using a dummy source sentence, and using a source sentence obtained via back-translation, which we call synthetic. We find that the latter is more effective.

- we successfully adapt NMT models to a new domain by fine-tuning with either monolingual or parallel in-domain data.

2 Neural Machine Translation

We follow the neural machine translation architecture by Bahdanau et al. (2015), which we will briefly summarize here. However, we note that our approach is not specific to this architecture.

The neural machine translation system is implemented as an encoder-decoder network with recurrent neural networks.

The encoder is a bidirectional neural network with gated recurrent units (Cho et al., 2014) that reads an input sequence $x = (x_1, ..., x_m)$ and calculates a forward sequence of hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_m)$, and a backward sequence $(\overleftarrow{h}_1, ..., \overleftarrow{h}_m)$. The hidden states $\overrightarrow{h}_j$ and $\overleftarrow{h}_j$ are concatenated to obtain the annotation vector h_j.

The decoder is a recurrent neural network that predicts a target sequence $y = (y_1, ..., y_n)$. Each word y_i is predicted based on a recurrent hidden state s_i, the previously predicted word y_{i-1}, and a context vector c_i. c_i is computed as a weighted sum of the annotations h_j. The weight of each annotation h_j is computed through an *alignment model* α_{ij}, which models the probability that y_i is aligned to x_j. The alignment model is a single-layer feedforward neural network that is learned jointly with the rest of the network through backpropagation.

A detailed description can be found in (Bahdanau et al., 2015). Training is performed on a parallel corpus with stochastic gradient descent. For translation, a beam search with small beam size is employed.

3 NMT Training with Monolingual Training Data

In machine translation, more monolingual data (or monolingual data more similar to the test set) serves to improve the estimate of the prior probability $p(T)$ of the target sentence T, before taking the source sentence S into account. In contrast to (Gülçehre et al., 2015), who train separate language models on monolingual training data and incorporate them into the neural network through shallow or deep fusion, we propose techniques to train the main NMT model with monolingual data, exploiting the fact that encoder-decoder neural networks already condition the probability distribution of the next target word on the previous target words. We describe two strategies to do this: providing monolingual training examples with an empty (or dummy) source sentence, or providing monolingual training data with a synthetic source sentence that is obtained from automatically translating the target sentence into the source language, which we will refer to as *back-translation*.

3.1 Dummy Source Sentences

The first technique we employ is to treat monolingual training examples as parallel examples with empty source side, essentially adding training examples whose context vector c_i is uninformative, and for which the network has to fully rely on the previous target words for its prediction. This could be conceived as a form of dropout (Hinton et al., 2012), with the difference that the training instances that have the context vector dropped out constitute novel training data. We can also conceive of this setup as multi-task learning, with the two tasks being translation when the source is known, and language modelling when it is unknown.

During training, we use both parallel and monolingual training examples in the ratio 1-to-1, and randomly shuffle them. We define an epoch as one iteration through the parallel data set, and resample from the monolingual data set for every epoch. We pair monolingual sentences with a single-word dummy source side *<null>* to allow processing of both parallel and monolingual training examples with the same network graph.[1] For monolingual minibatches[2], we freeze the network parameters of the encoder and the attention model.

One problem with this integration of monolin-

[1] One could force the context vector c_i to be 0 for monolingual training instances, but we found that this does not solve the main problem with this approach, discussed below.

[2] For efficiency, Bahdanau et al. (2015) sort sets of 20 minibatches according to length. This also groups monolingual training instances together.

gual data is that we cannot arbitrarily increase the ratio of monolingual training instances, or fine-tune a model with only monolingual training data, because different output layer parameters are optimal for the two tasks, and the network 'unlearns' its conditioning on the source context if the ratio of monolingual training instances is too high.

3.2 Synthetic Source Sentences

To ensure that the output layer remains sensitive to the source context, and that good parameters are not unlearned from monolingual data, we propose to pair monolingual training instances with a synthetic source sentence from which a context vector can be approximated. We obtain these through back-translation, i.e. an automatic translation of the monolingual target text into the source language.

During training, we mix synthetic parallel text into the original (human-translated) parallel text and do not distinguish between the two: no network parameters are frozen. Importantly, only the source side of these additional training examples is synthetic, and the target side comes from the monolingual corpus.

4 Evaluation

We evaluate NMT training on parallel text, and with additional monolingual data, on English↔German and Turkish→English, using training and test data from WMT 15 for English↔German, IWSLT 15 for English→German, and IWSLT 14 for Turkish→English.

4.1 Data and Methods

We use Groundhog[3] as the implementation of the NMT system for all experiments (Bahdanau et al., 2015; Jean et al., 2015a). We generally follow the settings and training procedure described by Sennrich et al. (2016).

For English↔German, we report case-sensitive BLEU on detokenized text with mteval-v13a.pl for comparison to official WMT and IWSLT results. For Turkish→English, we report case-sensitive BLEU on tokenized text with multi-bleu.perl for comparison to results by Gülçehre et al. (2015).

Gülçehre et al. (2015) determine the network vocabulary based on the parallel training data, and replace out-of-vocabulary words with a special UNK symbol. They remove monolingual sentences with more than 10% UNK symbols. In contrast, we represent unseen words as sequences of subword units (Sennrich et al., 2016), and can represent any additional training data with the existing network vocabulary that was learned on the parallel data. In all experiments, the network vocabulary remains fixed.

dataset	sentences
$\text{WMT}_{\text{parallel}}$	4 200 000
$\text{WIT}_{\text{parallel}}$	200 000
$\text{WMT}_{\text{mono\_de}}$	160 000 000
$\text{WMT}_{\text{synth\_de}}$	3 600 000
$\text{WMT}_{\text{mono\_en}}$	118 000 000
$\text{WMT}_{\text{synth\_en}}$	4 200 000

Table 1: English↔German training data.

4.1.1 English↔German

We use all parallel training data provided by WMT 2015 (Bojar et al., 2015)[4]. We use the News Crawl corpora as additional training data for the experiments with monolingual data. The amount of training data is shown in Table 1.

Baseline models are trained for a week. Ensembles are sampled from the last 4 saved models of training (saved at 12h-intervals). Each model is fine-tuned with fixed embeddings for 12 hours.

For the experiments with synthetic parallel data, we back-translate a random sample of 3 600 000 sentences from the German monolingual data set into English. The German→English system used for this is the baseline system (`parallel`). Translation took about a week on an NVIDIA Titan Black GPU. For experiments in German→English, we back-translate 4 200 000 monolingual English sentences into German, using the English→German system `+synthetic`. Note that we always use single models for back-translation, not ensembles. We leave it to future work to explore how sensitive NMT training with synthetic data is to the quality of the back-translation.

We tokenize and truecase the training data, and represent rare words via BPE (Sennrich et al., 2016). Specifically, we follow Sennrich et al. (2016) in performing BPE on the joint vocabulary with 89 500 merge operations. The network vo-

[3] github.com/sebastien-j/LV_groundhog

[4] http://www.statmt.org/wmt15/

dataset	sentences
WIT	160 000
SETimes	160 000
$\text{Gigaword}_{\text{mono}}$	177 000 000
$\text{Gigaword}_{\text{synth}}$	3 200 000

Table 2: Turkish→English training data.

cabulary size is 90 000.

We also perform experiments on the IWSLT 15 test sets to investigate a cross-domain setting.[5] The test sets consist of TED talk transcripts. As in-domain training data, IWSLT provides the WIT[3] parallel corpus (Cettolo et al., 2012), which also consists of TED talks.

4.1.2 Turkish→English

We use data provided for the IWSLT 14 machine translation track (Cettolo et al., 2014), namely the WIT[3] parallel corpus (Cettolo et al., 2012), which consists of TED talks, and the SETimes corpus (Tyers and Alperen, 2010).[6] After removal of sentence pairs which contain empty lines or lines with a length ratio above 9, we retain 320 000 sentence pairs of training data. For the experiments with monolingual training data, we use the English LDC Gigaword corpus (Fifth Edition). The amount of training data is shown in Table 2. With only 320 000 sentences of parallel data available for training, this is a much lower-resourced translation setting than English↔German.

Gülçehre et al. (2015) segment the Turkish text with the morphology tool Zemberek, followed by a disambiguation of the morphological analysis (Sak et al., 2007), and removal of non-surface tokens produced by the analysis. We use the same preprocessing[7]. For both Turkish and English, we represent rare words (or morphemes in the case of Turkish) as character bigram sequences (Sennrich et al., 2016). The 20 000 most frequent words (morphemes) are left unsegmented. The networks have a vocabulary size of 23 000 symbols.

To obtain a synthetic parallel training set, we back-translate a random sample of 3 200 000 sentences from Gigaword. We use an English→Turkish NMT system trained with the same settings as the Turkish→English baseline system.

We found overfitting to be a bigger problem than with the larger English↔German data set, and follow Gülçehre et al. (2015) in using Gaussian noise (stddev 0.01) (Graves, 2011), and dropout on the output layer (p=0.5) (Hinton et al., 2012). We also use early stopping, based on BLEU measured every three hours on tst2010, which we treat as development set. For Turkish→English, we use gradient clipping with threshold 5, following Gülçehre et al. (2015), in contrast to the threshold 1 that we use for English↔German, following Jean et al. (2015a).

4.2 Results

4.2.1 English→German WMT 15

Table 3 shows English→German results with WMT training and test data. We find that mixing parallel training data with monolingual data with a dummy source side in a ratio of 1-1 improves quality by 0.4–0.5 BLEU for the single system, 1 BLEU for the ensemble. We train the system for twice as long as the baseline to provide the training algorithm with a similar amount of parallel training instances. To ensure that the quality improvement is due to the monolingual training instances, and not just increased training time, we also continued training our baseline system for another week, but saw no improvements in BLEU.

Including synthetic data during training is very effective, and yields an improvement over our baseline by 2.8–3.4 BLEU. Our best ensemble system also outperforms a syntax-based baseline (Sennrich and Haddow, 2015) by 1.2–2.1 BLEU. We also substantially outperform NMT results reported by Jean et al. (2015a) and Luong et al. (2015), who previously reported SOTA result.[8] We note that the difference is particularly large for single systems, since our ensemble is not as diverse as that of Luong et al. (2015), who used 8 independently trained ensemble components, whereas we sampled 4 ensemble components from the same training run.

4.2.2 English→German IWSLT 15

Table 4 shows English→German results on IWSLT test sets. IWSLT test sets consist of TED talks, and are thus very dissimilar from the WMT

[5] http://workshop2015.iwslt.org/

[6] http://workshop2014.iwslt.org/

[7] github.com/orhanf/zemberekMorphTR

[8] Luong et al. (2015) report 20.9 BLEU (tokenized) on newstest2014 with a single model, and 23.0 BLEU with an ensemble of 8 models. Our best single system achieves a *tokenized* BLEU (as opposed to untokenized scores reported in Table 3) of 23.8, and our ensemble reaches 25.0 BLEU.

name	training instances	BLEU newstest2014 single	BLEU newstest2014 ens-4	BLEU newstest2015 single	BLEU newstest2015 ens-4
syntax-based (Sennrich and Haddow, 2015)		22.6	-	24.4	-
Neural MT (Jean et al., 2015b)		-	-	22.4	-
parallel	37m (parallel)	19.9	20.4	22.8	23.6
+monolingual	49m (parallel) / 49m (monolingual)	20.4	21.4	23.2	24.6
+synthetic	44m (parallel) / 36m (synthetic)	**22.7**	**23.8**	**25.7**	**26.5**

Table 3: English→German translation performance (BLEU) on WMT training/test sets. Ens-4: ensemble of 4 models. Number of training instances varies due to differences in training time and speed.

	name	fine-tuning data	fine-tuning instances	BLEU tst2013	BLEU tst2014	BLEU tst2015
	NMT (Luong and Manning, 2015) (single model)			29.4	-	-
	NMT (Luong and Manning, 2015) (ensemble of 8)			31.4	27.6	30.1
1	parallel	-	-	25.2	22.6	24.0
2	+synthetic	-	-	26.5	23.5	25.5
3	2+WIT_{mono_de}	$WMT_{parallel}$ / WIT_{mono}	200k/200k	26.6	23.6	25.4
4	2+WIT_{synth_de}	WIT_{synth}	200k	28.2	24.4	26.7
5	2+$WIT_{parallel}$	WIT	200k	**30.4**	**25.9**	**28.4**

Table 4: English→German translation performance (BLEU) on IWSLT test sets (TED talks). Single models.

test sets, which are news texts. We investigate if monolingual training data is especially valuable if it can be used to adapt a model to a new genre or domain, specifically adapting a system trained on WMT data to translating TED talks.

Systems 1 and 2 correspond to systems in Table 3, trained only on WMT data. System 2, trained on parallel and synthetic WMT data, obtains a BLEU score of 25.5 on tst2015. We observe that even a small amount of fine-tuning[9], i.e. continued training of an existing model, on WIT data can adapt a system trained on WMT data to the TED domain. By back-translating the monolingual WIT corpus (using a German→English system trained on WMT data, i.e. without in-domain knowledge), we obtain the synthetic data set WIT_{synth}. A single epoch of fine-tuning on WIT_{synth} (system 4) results in a BLEU score of 26.7 on tst2015, or an improvement of 1.2 BLEU. We observed no improvement from fine-tuning on WIT_{mono}, the monolingual TED corpus with dummy input (system 3).

These adaptation experiments with monolingual data are slightly artificial in that parallel training data is available. System 5, which is fine-tuned with the original WIT training data, obtains a BLEU of 28.4 on tst2015, which is an improvement of 2.9 BLEU. While it is unsurprising that in-domain parallel data is most valuable, we find it encouraging that NMT domain adaptation with monolingual data is also possible, and effective, since there are settings where only monolingual in-domain data is available.

[9]We leave the word embeddings fixed for fine-tuning.

name	BLEU 2014	BLEU 2015
PBSMT (Haddow et al., 2015)	28.8	29.3
NMT (Gülçehre et al., 2015)	23.6	-
+shallow fusion	23.7	-
+deep fusion	24.0	-
parallel	25.9	26.7
+synthetic	**29.5**	**30.4**
+synthetic (ensemble of 4)	30.8	31.6

Table 5: German→English translation performance (BLEU) on WMT training/test sets (newstest2014; newstest2015).

The best results published on this dataset are by Luong and Manning (2015), obtained with an ensemble of 8 independently trained models. In a comparison of single-model results, we outperform their model on tst2013 by 1 BLEU.

4.2.3 German→English WMT 15

Results for German→English on the WMT 15 data sets are shown in Table 5. Like for the reverse translation direction, we see substantial improvements (3.6–3.7 BLEU) from adding monolingual training data with synthetic source sentences, which is substantially bigger than the improvement observed with deep fusion (Gülçehre et al., 2015); our ensemble outperforms the previous state of the art on newstest2015 by 2.3 BLEU.

4.2.4 Turkish→English IWSLT 14

Table 6 shows results for Turkish→English. On average, we see an improvement of 0.6 BLEU on the test sets from adding monolingual data with a dummy source side in a 1-1 ratio[10], although we note a high variance between different test sets.

With synthetic training data ($\text{Gigaword}_{\text{synth}}$), we outperform the baseline by 2.7 BLEU on average, and also outperform results obtained via shallow or deep fusion by Gülçehre et al. (2015) by 0.5 BLEU on average. To compare to what extent synthetic data has a regularization effect, even without novel training data, we also back-translate the target side of the parallel training text to obtain the training corpus $\text{parallel}_{\text{synth}}$. Mixing the original parallel corpus with $\text{parallel}_{\text{synth}}$ (ratio 1-1) gives some improvement over the baseline (1.7 BLEU on average), but the novel monolingual training data ($\text{Gigaword}_{\text{mono}}$) gives higher improvements, despite being out-of-domain in relation to the test sets. We speculate that novel in-domain monolingual data would lead to even higher improvements.

4.2.5 Back-translation Quality for Synthetic Data

One question that our previous experiments leave open is how the quality of the automatic back-translation affects training with synthetic data. To investigate this question, we back-translate the same German monolingual corpus with three different German→English systems:

- with our baseline system and greedy decoding
- with our baseline system and beam search (beam size 12). This is the same system used for the experiments in Table 3.
- with the German→English system that was itself trained with synthetic data (beam size 12).

[10] We also experimented with higher ratios of monolingual data, but this led to decreased BLEU scores.

	BLEU		
	DE→EN	EN→DE	
back-translation	2015	2014	2015
none	-	20.4	23.6
parallel (greedy)	22.3	23.2	26.0
parallel (beam 12)	25.0	23.8	26.5
synthetic (beam 12)	28.3	23.9	26.6
ensemble of 3	-	24.2	27.0
ensemble of 12	-	24.7	27.6

Table 7: English→German translation performance (BLEU) on WMT training/test sets (newstest2014; newstest2015). Systems differ in how the synthetic training data is obtained. Ensembles of 4 models (unless specified otherwise).

BLEU scores of the German→English systems, and of the resulting English→German systems that are trained on the different back-translations, are shown in Table 7. The quality of the German→English back-translation differs substantially, with a difference of 6 BLEU on newstest2015. Regarding the English→German systems trained on the different synthetic corpora, we find that the 6 BLEU difference in back-translation quality leads to a 0.6–0.7 BLEU difference in translation quality. This is balanced by the fact that we can increase the speed of back-translation by trading off some quality, for instance by reducing beam size, and we leave it to future research to explore how much the amount of synthetic data affects translation quality.

We also show results for an ensemble of 3 models (the best single model of each training run), and 12 models (all 4 models of each training run). Thanks to the increased diversity of the ensemble components, these ensembles outperform the ensembles of 4 models that were all sampled from the same training run, and we obtain another improvement of 0.8–1.0 BLEU.

4.3 Contrast to Phrase-based SMT

The back-translation of monolingual target data into the source language to produce synthetic parallel text has been previously explored for phrase-based SMT (Bertoldi and Federico, 2009; Lambert et al., 2011). While our approach is technically similar, synthetic parallel data fulfills novel roles

name	training data	instances	BLEU tst2011	tst2012	tst2013	tst2014
baseline (Gülçehre et al., 2015)			18.4	18.8	19.9	18.7
deep fusion (Gülçehre et al., 2015)			20.2	20.2	21.3	**20.6**
baseline	parallel	7.2m	18.6	18.2	18.4	18.3
$\text{parallel}_{\text{synth}}$	parallel/$\text{parallel}_{\text{synth}}$	6m/6m	19.9	20.4	20.1	20.0
$\text{Gigaword}_{\text{mono}}$	parallel/$\text{Gigaword}_{\text{mono}}$	7.6m/7.6m	18.8	19.6	19.4	18.2
$\text{Gigaword}_{\text{synth}}$	parallel/$\text{Gigaword}_{\text{synth}}$	8.4m/8.4m	**21.2**	**21.1**	**21.8**	20.4

Table 6: Turkish→English translation performance (*tokenized* BLEU) on IWSLT test sets (TED talks). Single models. Number of training instances varies due to early stopping.

system	BLEU WMT	IWSLT
parallel	20.1	21.5
+synthetic	20.8	21.6
PBSMT gain	+0.7	+0.1
NMT gain	+2.9	+1.2

Table 8: Phrase-based SMT results (English→German) on WMT test sets (average of newstest201{4,5}), and IWSLT test sets (average of tst201{3,4,5}), and average BLEU gain from adding synthetic data for both PBSMT and NMT.

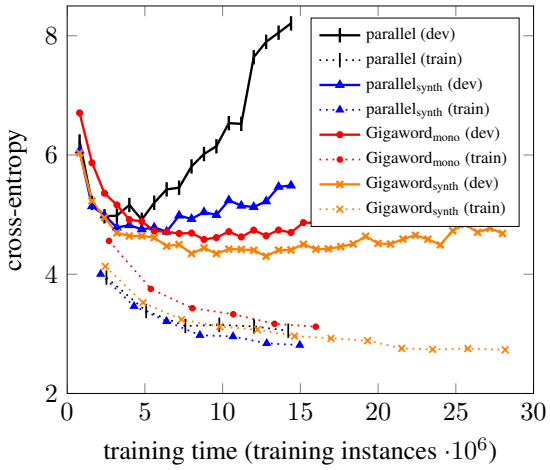


Figure 1: Turkish→English training and development set (tst2010) cross-entropy as a function of training time (number of training instances) for different systems.

in NMT.

To explore the relative effectiveness of back-translated data for phrase-based SMT and NMT, we train two phrase-based SMT systems with Moses (Koehn et al., 2007), using only $\text{WMT}_{\text{parallel}}$, or both $\text{WMT}_{\text{parallel}}$ and $\text{WMT}_{\text{synth\_de}}$ for training the translation and reordering model. Both systems contain the same language model, a 5-gram Kneser-Ney model trained on all available WMT data. We use the baseline features described by Haddow et al. (2015).

Results are shown in Table 8. In phrase-based SMT, we find that the use of back-translated training data has a moderate positive effect on the WMT test sets (+0.7 BLEU), but not on the IWSLT test sets. This is in line with the expectation that the main effect of back-translated data for phrase-based SMT is domain adaptation (Bertoldi and Federico, 2009). Both the WMT test sets and the News Crawl corpora which we used as monolingual data come from the same source, a web crawl of newspaper articles.[11] In contrast, News Crawl is out-of-domain for the IWSLT test sets.

In contrast to phrase-based SMT, which can make use of monolingual data via the language model, NMT has so far not been able to use monolingual data to great effect, and without requiring architectural changes. We find that the effect of synthetic parallel data is not limited to domain adaptation, and that even out-of-domain synthetic data improves NMT quality, as in our evaluation on IWSLT. The fact that the synthetic data is more effective on the WMT test sets (+2.9 BLEU) than on the IWSLT test sets (+1.2 BLEU) supports the hypothesis that domain adaptation contributes to the effectiveness of adding synthetic data to NMT training.

It is an important finding that back-translated data, which is mainly effective for domain adaptation in phrase-based SMT, is more generally useful in NMT, and has positive effects that go beyond domain adaptation. In the next section, we will investigate further reasons for its effectiveness.

[11] The WMT test sets are held-out from News Crawl.

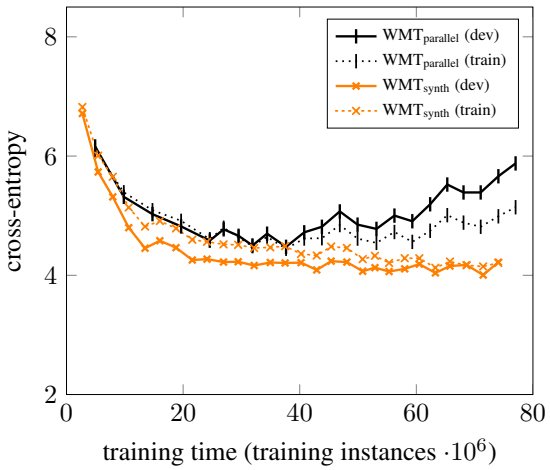


Figure 2: English→German training and development set (newstest2013) cross-entropy as a function of training time (number of training instances) for different systems.

4.4 Analysis

We previously indicated that overfitting is a concern with our baseline system, especially on small data sets of several hundred thousand training sentences, despite the regularization employed. This overfitting is illustrated in Figure 1, which plots training and development set cross-entropy by training time for Turkish→English models. For comparability, we measure training set cross-entropy for all models on the same random sample of the parallel training set. We can see that the model trained on only parallel training data quickly overfits, while all three monolingual data sets (parallel$_{synth}$, Gigaword$_{mono}$, or Gigaword$_{synth}$) delay overfitting, and give better perplexity on the development set. The best development set cross-entropy is reached by Gigaword$_{synth}$.

Figure 2 shows cross-entropy for English→German, comparing the system trained on only parallel data and the system that includes synthetic training data. Since more training data is available for English→German, there is no indication that overfitting happens during the first 40 million training instances (or 7 days of training); while both systems obtain comparable training set cross-entropies, the system with synthetic data reaches a lower cross-entropy on the development set. One explanation for this is the domain effect discussed in the previous section.

A central theoretical expectation is that monolingual target-side data improves the model's fluency, its ability to produce natural target-language sentences. As a proxy to sentence-level fluency, we investigate word-level fluency, specifically words produced as sequences of subword units, and whether NMT systems trained with additional monolingual data produce more natural words. For instance, the English→German systems translate the English phrase *civil rights protections* as a single compound, composed of three subword units: *Bürger|rechts|schutzes*[12], and we analyze how many of these multi-unit words that the translation systems produce are well-formed German words.

system	produced	attested	natural
parallel	1078	53.4%	74.9%
+mono	994	61.6%	84.6%
+synthetic	1217	56.4%	82.5%

Table 9: Number of words in system output that do not occur in parallel training data (count$_{ref}$ = 1168), and proportion that is attested in data, or natural according to native speaker. English→German; newstest2015; ensemble systems.

We compare the number of words in the system output for the newstest2015 test set which are produced via subword units, and that do not occur in the parallel training corpus. We also count how many of them are attested in the full monolingual corpus or the reference translation, which we all consider 'natural'. Additionally, the main authors, a native speaker of German, annotated a random subset ($n = 100$) of unattested words of each system according to their naturalness[13], distinguishing between natural German words (or names) such as *Literatur|klassen* 'literature classes', and nonsensical ones such as **As|best|atten* (a misspelling of *Astbestmatten* 'asbestos mats').

In the results (Table 9), we see that the systems trained with additional monolingual or synthetic data have a higher proportion of novel words attested in the non-parallel data, and a higher proportion that is deemed natural by our annotator. This supports our expectation that additional monolingual data improves the (word-level) fluency of the NMT system.

[12] Subword boundaries are marked with '|'.

[13] For the annotation, the words were blinded regarding the system that produced them.

5 Related Work

To our knowledge, the integration of monolingual data for pure neural machine translation architectures was first investigated by (Gülçehre et al., 2015), who train monolingual language models independently, and then integrate them during decoding through rescoring of the beam (*shallow fusion*), or by adding the recurrent hidden state of the language model to the decoder state of the encoder-decoder network, with an additional controller mechanism that controls the magnitude of the LM signal (*deep fusion*). In deep fusion, the controller parameters and output parameters are tuned on further parallel training data, but the language model parameters are fixed during the fine-tuning stage. Jean et al. (2015b) also report on experiments with reranking of NMT output with a 5-gram language model, but improvements are small (between 0.1–0.5 BLEU).

The production of synthetic parallel texts bears resemblance to data augmentation techniques used in computer vision, where datasets are often augmented with rotated, scaled, or otherwise distorted variants of the (limited) training set (Rowley et al., 1996).

Another similar avenue of research is self-training (McClosky et al., 2006; Schwenk, 2008). The main difference is that self-training typically refers to scenario where the training set is enhanced with training instances with artificially produced output labels, whereas we start with human-produced output (i.e. the translation), and artificially produce an input. We expect that this is more robust towards noise in the automatic translation. Improving NMT with monolingual source data, following similar work on phrase-based SMT (Schwenk, 2008), remains possible future work.

Domain adaptation of neural networks via continued training has been shown to be effective for neural language models by (Ter-Sarkisov et al., 2015), and in work parallel to ours, for neural translation models (Luong and Manning, 2015). We are the first to show that we can effectively adapt neural translation models with monolingual data.

6 Conclusion

In this paper, we propose two simple methods to use monolingual training data during training of NMT systems, with no changes to the network architecture. Providing training examples with dummy source context was successful to some extent, but we achieve substantial gains in all tasks, and new SOTA results, via back-translation of monolingual target data into the source language, and treating this synthetic data as additional training data. We also show that small amounts of in-domain monolingual data, back-translated into the source language, can be effectively used for domain adaptation. In our analysis, we identified domain adaptation effects, a reduction of overfitting, and improved fluency as reasons for the effectiveness of using monolingual data for training.

While our experiments did make use of monolingual training data, we only used a small random sample of the available data, especially for the experiments with synthetic parallel data. It is conceivable that larger synthetic data sets, or data sets obtained via data selection, will provide bigger performance benefits.

Because we do not change the neural network architecture to integrate monolingual training data, our approach can be easily applied to other NMT systems. We expect that the effectiveness of our approach not only varies with the quality of the MT system used for back-translation, but also depends on the amount (and similarity to the test set) of available parallel and monolingual data, and the extent of overfitting of the baseline model. Future work will explore the effectiveness of our approach in more settings.

Acknowledgments

The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. - Samsung R&D Institute Poland. This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 645452 (QT21).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Nicola Bertoldi and Marcello Federico. 2009. Domain adaptation for statistical machine translation with monolingual resources. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*

StatMT 09. Association for Computational Linguistics.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, Lisbon, Portugal. Association for Computational Linguistics.

P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J.D. Lafferty, R.L. Mercer, and P.S. Roossin. 1990. A Statistical Approach to Machine Translation. *Computational Linguistics*, 16(2):79–85.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. WIT^3: Web Inventory of Transcribed and Translated Talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th IWSLT Evaluation Campaign, IWSLT 2014. In *Proceedings of the 11th Workshop on Spoken Language Translation*, pages 2–16, Lake Tahoe, CA, USA.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Alex Graves. 2011. Practical Variational Inference for Neural Networks. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc.

Çaglar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On Using Monolingual Corpora in Neural Machine Translation. *CoRR*, abs/1503.03535.

Barry Haddow, Matthias Huck, Alexandra Birch, Nikolay Bogoychev, and Philipp Koehn. 2015. The Edinburgh/JHU Phrase-based Machine Translation Systems for WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 126–133, Lisbon, Portugal. Association for Computational Linguistics.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015a. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015b. Montreal Neural Machine Translation Systems for WMT'15 . In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140, Lisbon, Portugal. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL-2007 Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Patrik Lambert, Holger Schwenk, Christophe Servan, and Sadaf Abdul-Rauf. 2011. Investigations on Translation Model Adaptation Using Monolingual Data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 284–293, Edinburgh, Scotland. Association for Computational Linguistics.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford Neural Machine Translation Systems for Spoken Language Domains. In *Proceedings of the International Workshop on Spoken Language Translation 2015*, Da Nang, Vietnam.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective Self-training for Parsing. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 152–159, New York. Association for Computational Linguistics.

Henry Rowley, Shumeet Baluja, and Takeo Kanade. 1996. Neural Network-Based Face Detection. In *Computer Vision and Pattern Recognition '96*.

Haşim Sak, Tunga Güngör, and Murat Saraçlar. 2007. Morphological Disambiguation of Turkish Text with Perceptron Algorithm. In *CICLing 2007*, pages 107–118.

Holger Schwenk. 2008. Investigations on Large-Scale Lightly-Supervised Training for Statistical Machine Translation. In *International Workshop on Spoken Language Translation*, pages 182–189.

Rico Sennrich and Barry Haddow. 2015. A Joint Dependency Model of Morphological and Syntactic Structure for Statistical Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2081–2087, Lisbon, Portugal. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, Berlin, Germany.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112, Montreal, Quebec, Canada.

Alex Ter-Sarkisov, Holger Schwenk, Fethi Bougares, and Loïc Barrault. 2015. Incremental Adaptation Strategies for Neural Network Language Models. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 48–56, Beijing, China. Association for Computational Linguistics.

Francis M. Tyers and Murat S. Alperen. 2010. SETimes: A parallel corpus of Balkan languages. In *Workshop on Exploitation of multilingual resources and tools for Central and (South) Eastern European Languages at the Language Resources and Evaluation Conference*, pages 1–5.

Graph-Based Translation Via Graph Segmentation

Liangyou Li and **Andy Way** and **Qun Liu**
ADAPT Centre, School of Computing
Dublin City University, Ireland
{liangyouli,away,qliu}@computing.dcu.ie

Abstract

One major drawback of phrase-based translation is that it segments an input sentence into continuous phrases. To support linguistically informed source discontinuity, in this paper we construct graphs which combine bigram and dependency relations and propose a graph-based translation model. The model segments an input graph into connected subgraphs, each of which may cover a discontinuous phrase. We use beam search to combine translations of each subgraph left-to-right to produce a complete translation. Experiments on Chinese–English and German–English tasks show that our system is significantly better than the phrase-based model by up to +1.5/+0.5 BLEU scores. By explicitly modeling the graph segmentation, our system obtains further improvement, especially on German–English.

1 Introduction

Statistical machine translation (SMT) starts from sequence-based models. The well-known phrase-based (PB) translation model (Koehn et al., 2003) has significantly advanced the progress of SMT by extending translation units from single words to phrases. By using phrases, PB models can capture local phenomena, such as word order, word deletion, and word insertion. However, one of the significant weaknesses in conventional PB models is that only continuous phrases are used, so generalizations such as French *ne ... pas* to English *not* cannot be learned. To solve this, syntax-based models (Galley et al., 2004; Chiang, 2005; Liu et al., 2006; Marcu et al., 2006) take tree structures into consideration to learn translation patterns by using non-terminals for generalization.

Model	C	D	S
(Koehn et al., 2003)	•		sequence
(Galley and Manning, 2010)	•	•	sequence
(Quirk et al., 2005) and (Menezes and Quirk, 2005)		•	tree
This work	•	•	graph

Table 1: Comparison between our work and previous work in terms of three aspects: keeping continuous phrases (C), allowing discontinuous phrases (D), and input structures (S).

However, the expressiveness of these models is confined by hierarchical constraints of the grammars used (Galley and Manning, 2010) since these patterns still cover continuous spans of an input sentence.

By contrast, Quirk et al. (2005), Menezes and Quirk (2005) and Xiong et al. (2007) take treelets from dependency trees as the basic translation units. These treelets are connected and may cover discontinuous phrases. However, their models lack the ability to handle continuous phrases which are not connected in trees but could in fact be extremely important to system performance (Koehn et al., 2003). Galley and Manning (2010) directly extract discontinuous phrases from input sequences. However, without imposing additional restrictions on discontinuity, the amount of extracted rules can be very large and unreliable.

Different from previous work (as shown in Table 1), in this paper we use graphs as input structures and propose a graph-based translation model to translate a graph into a target string. The basic translation unit in this model is a connected subgraph which may cover discontinuous phrases. The main contributions of this work are summarized as follows:

- We propose to use a graph structure to combine a sequence and a tree (Section 3.1). The

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 97–107,
Berlin, Germany, August 7-12, 2016.

graph contains both local relations between words from the sequence and long-distance relations from the tree.

- We present a translation model to translate a graph (Section 3). The model segments the graph into subgraphs and uses beam search to generate a complete translation from left to right by combining translation options of each subgraph.
- We present a set of sparse features to explicitly model the graph segmentation (Section 4). These features are based on edges in the input graph, each of which is either inside a subgraph or connects the subgraph with a previous subgraph.
- Experiments (Section 5) on Chinese–English and German–English tasks show that our model is significantly better than the PB model. After incorporating the segmentation model, our system achieves still further improvement.

2 Review: Phrase-based Translation

We first review the basic PB translation approach, which will be extended to our graph-based translation model. Given a pair of sentences $\langle S, T \rangle$, the conventional PB model is defined as Equation (1):

$$p(\bar{t}_1^I \mid \bar{s}_1^I) = \prod_{i=1}^{I} p(\bar{t}_i|\bar{s}_{a_i}) d(\bar{s}_{a_i}, \bar{s}_{a_{i-1}}) \quad (1)$$

The target sentence T is broken into I phrases $\bar{t}_1 \cdots \bar{t}_I$, each of which is a translation of a source phrase $\bar{s}_{a_i}$. d is a distance-based reordering model. Note that in the basic PB model, the phrase segmentation is not explicitly modeled which means that different segmentations are treated equally (Koehn, 2010).

The performance of PB translation relies on the quality of phrase pairs in a translation table. Conventionally, a phrase pair $\langle \bar{s}, \bar{t} \rangle$ has two properties: (i) $\bar{s}$ and $\bar{t}$ are continuous phrases. (ii) $\langle \bar{s}, \bar{t} \rangle$ is consistent with a word alignment A (Och and Ney, 2004): $\forall (i,j) \in A, s_i \in \bar{s} \Leftrightarrow t_j \in \bar{t}$ and $\exists s_i \in \bar{s}, t_j \in \bar{t}, (i,j) \in A$.

PB decoders generate hypotheses (partial translations) from left to right. Each hypothesis maintains a *coverage vector* to indicate which source words have been translated so far. A hypothesis can be extended on the right by translating an uncovered source phrase. The translation process ends when all source words have been translated.

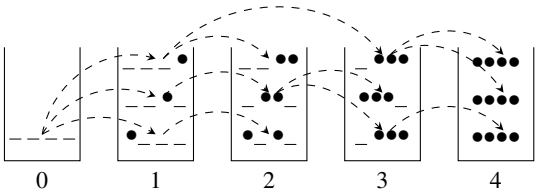


Figure 1: Beam search for phrase-based MT. • denotes a covered source position while _ indicates an uncovered position (Liu and Huang, 2014).

Beam search (as in Figure 1) is taken as an approximate search strategy to reduce the size of the decoding space. Hypotheses which cover the same number of source words are grouped in a stack. Hypotheses can be pruned according to their partial translation cost and an estimated future cost.

3 Graph-Based Translation

Our graph-based translation model extends PB translation by translating an input graph rather than a sequence to a target string. The graph is segmented into a sequence of connected subgraphs, each of which corresponds to a target phrase, as in Equation (2):

$$\begin{aligned} & p(\bar{t}_1^I \mid G(\tilde{s}_1^I)) \\ & = \prod_{i=1}^{I} p(\bar{t}_i|G(\tilde{s}_{a_i})) d(G(\tilde{s}_{a_i}), G(\tilde{s}_{a_{i-1}})) \\ & \approx \prod_{i=1}^{I} p(\bar{t}_i|G(\tilde{s}_{a_i})) d(\tilde{s}_{a_i}, \tilde{s}_{a_{i-1}}) \end{aligned} \quad (2)$$

where $G(\tilde{s}_i)$ denotes a connected source subgraph which covers a (discontinuous) phrase $\tilde{s}_i$.

3.1 Building Graphs

As a more powerful and natural structure for sentence modeling, a graph can model various kinds of word-relations together in a unified representation. In this paper, we use graphs to combine two commonly used relations: bigram relations and dependency relations. Figure 2 shows an example of a graph. Each edge in the graph denotes either a dependency relation or a bigram relation. Note that the graph we use in this paper is directed, connected, node-labeled and may contain cycles.

Bigram relations are implied in sequences and provide local and sequential information on pairs

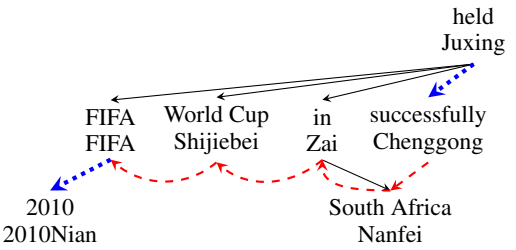


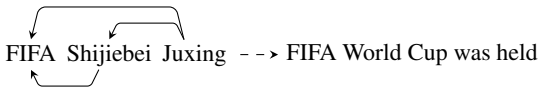
Figure 2: An example graph for a Chinese sentence. Each node includes a Chinese word and its English meaning. Dashed red lines are bigram relations. Solid lines are dependency relations. Dotted blue lines are shared by bigram and dependency relations.

of continuous words. Phrases connected by bigram relations (i.e. continuous phrases) are known to be useful to improve phrase coverage (Hanneman and Lavie, 2009). By contrast, dependency relations come from dependency structures which model syntactic and semantic relations between words. Phrases whose words are connected by dependency relations (also known as treelets) are linguistic-motivated and thus more reliable (Quirk et al., 2005).

By combining these two relations together in graphs, we can make use of both continuous and linguistic-informed discontinuous phrases as long as they are connected subgraphs.

3.2 Training

Different from PB translation, the basic translation units in our model are subgraphs. Thus, during training, we extract subgraph–phrase pairs instead of phrase pairs on parallel graph–string sentences associated with word alignments.[1] An example of a translation rule is as follows:



Note that the source side of a rule in our model is a graph which can be used to cover either a continuous phrase or a discontinuous phrase according to its match in an input graph during decoding.

The algorithm for extracting translation rules is shown in Algorithm 1. This algorithm traverses each phrase pair $\langle \tilde{s}, \bar{t} \rangle$, which is within a length limit and consistent with a given word alignment

[1]Different from translation rules in conventional syntax-based MT, rules in our model are not learned based on synchronous grammars and so non-terminals are disallowed.

Algorithm 1: Algorithm for extracting translation rules from a graph-string pair.

Data: A word-aligned graph–string pair $(G(S), T, A)$
Result: A set of translation pairs R

```
1  for each phrase t̄ in T: | t̄ | ≤ L do
2      find the minimal (may be discontinuous) phrase s̃ in S so that | s̃ | ≤ L and ⟨s̃, t̄⟩ is consistent with A ;
3      Queue Q = {s̃};
4      while Q is not empty do
5          pop an element s̃ off;
6          if G(s̃) is connected then
7              add ⟨G(s̃), t⟩ to R;
8          end
9          if | s̃ | < L then
10             for each unaligned word s_i adjacent to s̃ do
11                 s̃′ = extend s̃ with s_i;
12                 add s̃′ to Q;
13             end
14         end
15     end
16 end
```

(lines 1–2), and outputs $\langle G(\tilde{s}), \bar{t} \rangle$ if $\tilde{s}$ is covered by a connected subgraph $G(\tilde{s})$ (lines 6–8). A source phrase can be extended with unaligned source words which are adjacent to the phrase (lines 9–14). We use a queue Q to store all phrases which are consistently aligned to the same target phrase (line 3).

3.3 Model and Decoding

We define our model in the log-linear framework (Och and Ney, 2002) over a derivation $D = r_1 r_2 \cdots r_N$, as in Equation (3):

$$p(D) \propto \prod_i \phi_i(D)^{\lambda_i} \tag{3}$$

where r_i are translation rules, ϕ_i are features defined on derivations and λ_i are feature weights. In our experiments, we use the standard 9 features: two translation probabilities $p(G(s)|t)$ and $p(t|G(s))$, two lexical translation probabilities $p_{lex}(s|t)$ and $p_{lex}(t|s)$, a language model $lm(t)$ over a translation t, a rule penalty, a word penalty, an unknown word penalty and a distortion feature d for distance-based reordering.

The calculation of the distortion feature d in our

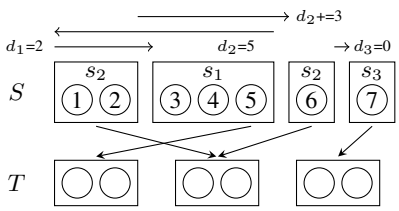


Figure 3: Distortion calculation for both continuous and discontinuous phrases in a derivation.

model is different from the one used in conventional PB models, as we need to take discontinuity into consideration. In this paper, we use a distortion function defined in Galley and Manning (2010) to penalize discontinuous phrases that have relatively long gaps. Figure 3 shows an example of calculating distortion for discontinuous phrases.

Our graph-based decoder is very similar to the PB decoder except that, in our decoder, each hypothesis is extended by translating an uncovered subgraph instead of a phrase. Positions covered by the subgraph are then marked as translated.

4 Graph Segmentation Model

Each derivation in our graph-based translation model implies a sequence of subgraphs (also called a segmentation). By default, similar to PB translation, our model treats each segmentation equally as shown in Equation (2). However, previous work on PB translation has suggested that such segmentations provide useful information which can improve translation performance. For example, boundary information in a phrase segmentation can be used for reordering models (Xiong et al., 2006; Cherry, 2013).

In this paper, we are interested in directly modeling the segmentation using information from graphs. By making the assumption that each subgraph is only dependent on previous subgraphs, we define a generative process over a graph segmentation as in Equation (4):

$$\begin{aligned} &p(G(\tilde{s}_1)\cdots G(\tilde{s}_I)) \\ &= \prod_{i=1}^{I} P(G(\tilde{s}_i)|G(\tilde{s}_1)\cdots G(\tilde{s}_{i-1})) \end{aligned} \quad (4)$$

Instead of training a stand-alone discriminative segmentation model to assign each subgraph a probability given previous subgraphs, we implement the model via sparse features, each of which is extracted at run-time during decoding and then directly added to the log-linear framework, so that these features can be tuned jointly with other features (of Section 3.3) to directly maximize the translation quality.

ZH–EN	#Sents
Train	1.5M+
MT02 (Dev)	878
MT04	1,597
MT05	1,082

DE–EN	#Sents
Train	2M+
WMT11 (Dev)	3,003
WMT12	3,003
WMT13	3,000

Table 2: The number of sentences in our corpora.

Since a segmentation is obtained by breaking up the connectivity of an input graph, it is intuitive to use edges to model the segmentation. According to Equation (4), for a current subgraph G_i, we only consider those edges which are either inside G_i or connect G_i with a previous subgraph. Based on these edges, we extract sparse features for each node in the subgraph. The set of sparse features is defined as follows:

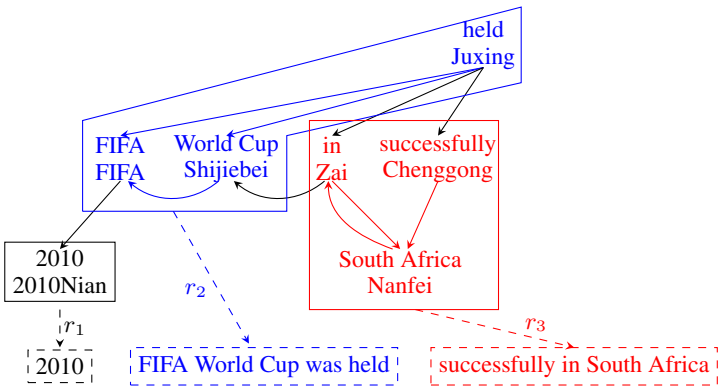
$$\begin{Bmatrix} n.w \\ n.c \end{Bmatrix} \times \begin{Bmatrix} n'.w \\ n'.c \end{Bmatrix} \times \begin{Bmatrix} C \\ P \\ H \end{Bmatrix} \times \begin{Bmatrix} in \\ out \end{Bmatrix}$$

where $n.w$ and $n.c$ are the word and class of the current node n, and $n'.w$ and $n'.c$ are the word and class of a node n' connected to n. C, P, and H denote that the node n' is in the current subgraph G_i or the adjacent previous subgraph G_{i-1} or other previous subgraphs, respectively. Note that we treat the adjacent previous subgraph differently from others since information from the last previous unit is quite useful (Xiong et al., 2006; Cherry, 2013). *in* and *out* denote that the edge is an incoming edge or outgoing edge for the current node n. Figure 4 shows an example of extracting sparse features for a subgraph.

Inspired by success in using sparse features in SMT (Cherry, 2013), in this paper we lexicalize only on the top-100 most frequent words. In addition, we group source words into 50 classes by using *mkcls* which should provide useful generalization (Cherry, 2013) for our model.

5 Experiment

We conduct experiments on Chinese–English (ZH–EN) and German–English (DE–EN) translation tasks. Table 2 provides a summary of our corpra. Our ZH–EN training corpus contains 1.5M+ sentences from LDC. NIST 2002 (MT02) is taken as a development set to tune weights, and NIST



Sparse features for r_3:

W:Zai_W:Nanfei_C_in	C:4_W:Nanfei_C_in	W:Zai_C:5_C_in	C:4_C:5_C_in
W:Zai_W:Nanfei_C_out	C:4_W:Nanfei_C_out	W:Zai_C:5_C_out	C:4_C:5_C_out
W:Zai_W:Shijiebei_P_out	C:4_W:Shijiebei_P_out	W:Zai_C:3_P_out	C:4_C:3_P_out
W:Zai_W:Juxing_P_in	C:4_W:Juxing_P_in	W:Zai_C:7_P_in	C:4_C:7_P_in
W:Nanfei_W:Zai_C_in	C:5_W:Zai_C_in	W:Nanfei_C:4_C_in	C:5_C:4_C_in
W:Nanfei_W:Zai_C_out	C:5_W:Zai_C_out	W:Nanfei_C:4_C_out	C:5_C:4_C_out
W:Nanfei_W:Chenggong_C_in	C:5_W:Chenggong_C_in	W:Nanfei_C:6_C_in	C:5_C:6_C_in
W:Chenggong_W:Nanfei_C_out	C:6_W:Nanfei_C_out	W:Chenggong_C:5_C_out	C:6_C:5_C_out
W:Chenggong_W:Juxing_P_in	C:6_W:Juxing_P_in	W:Chenggong_C:7_P_in	C:6_C:7_P_in

Figure 4: An illustration of extracting sparse features for each node in a subgraph during decoding. The decoder segments the graph in Figure 2 into three subgraphs (solid rectangles) and produces a complete translation by combining translations of each subgraph (dashed rectangles). In this figure, the class of a word is randomly assigned.

2004 (MT04) and NIST 2005 (MT05) are two test sets used to evaluate the systems. The Stanford Chinese word segmenter (Chang et al., 2008) is used to segment Chinese sentences. The Stanford dependency parser (Chang et al., 2009) parses a Chinese sentence into a projective dependency tree which is then converted to a graph by adding bigram relations.

The DE–EN training corpus is from WMT 2014, including Europarl V7 and News Commentary. News-Test 2011 (WMT11) is taken as a development set while News-Test 2012 (WMT12) and News-Test 2013 (WMT13) are test sets. We use mate-tools[2] to perform morphological analysis and parse German sentences (Bohnet, 2010). Then, MaltParser[3] converts a parse result into a projective dependency tree (Nivre and Nilsson, 2005).

5.1 Settings

In this paper, we mainly report results from five systems under the same configuration. **PBMT** is built by the PB model in Moses (Koehn et al., 2007). **Treelet** extends PBMT by taking treelets as the basic translation units (Quirk et al., 2005; Menezes and Quirk, 2005). We implement a Treelet model in Moses which produces translations from left to right and uses beam search for decoding. **DTU** extends the PB model by allowing discontinuous phrases (Galley and Manning, 2010). We implement DTU with source discontinuity in Moses.[4] **GBMT** is our basic graph-based translation system while **GSM** adds the graph segmentation model into GBMT. Both systems are implemented in Moses.

Word alignment is performed by GIZA++ (Och and Ney, 2003) with the heuristic function *grow-diag-final-and*. We use SRILM (Stolcke, 2002) to train a 5-gram language model on the Xinhua portion of the English Gigaword corpus 5th edition with modified Kneser-Ney discounting (Chen and Goodman, 1996). Batch MIRA (Cherry and Foster, 2012) is used to tune weights. BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2011), and TER (Snover et al., 2006) are used for evaluation.

[2] http://code.google.com/p/mate-tools/
[3] http://www.maltparser.org/

[4] The re-implementation of DTU in Moses makes it easier to meaningfully compare systems under the same settings.

Metric	System	ZH–EN		DE–EN	
		MT04	MT05	WMT12	WMT13
BLEU ↑	PBMT	33.2	31.8	19.5	21.9
	Treelet	33.8*	31.7	19.6	22.1*
	DTU	**34.5***	32.3*	**19.8***	**22.3***
	GBMT	**34.7***	**32.4***	**19.8***	**22.4***
	GSM	**34.9**$^{*+}$	**32.7**$^{*+}$	**20.3**$^{*+}$	**22.9**$^{*+}$
METEOR ↑	PBMT	32.1	32.3	28.0	29.2
	Treelet	31.9	31.8	28.0	29.1
	DTU	**32.3***	**32.4**	**28.2***	**29.5***
	GBMT	**32.4**$^{*+}$	**32.5***	**28.2***	**29.4***
	GSM	**32.7**$^{*+}$	**32.6**$^{*+}$	**28.5**$^{*+}$	**29.8**$^{*+}$
TER ↓	PBMT	60.6	61.6	63.7	60.2
	Treelet	60.1*	61.4	63.2*	59.6*
	DTU	60.0*	61.5	63.5*	59.8*
	GBMT	59.8^{*+}	61.3*	63.5*	59.8*
	GSM	60.5	62.1	63.1^{*+}	**59.3**$^{*+}$

Table 3: Metric scores for all systems on Chinese–English (ZH–EN) and German–English (DE–EN). Each score is an average over three MIRA runs (Clark et al., 2011). $*$ means a system is significantly better than PBMT at $p \leq 0.01$. Bold figures mean a system is significantly better than Treelet at $p \leq 0.01$. $+$ means a system is significantly better than DTU at $p \leq 0.01$. In this table, we mark a system by comparing it with previous ones.

5.2 Results and Discussion

Table 3 shows our evaluation results. We find that our GBMT system is significantly better than PBMT as measured by all three metrics across all test sets. Specifically, the improvements are up to +1.5/+0.5 BLEU, +0.3/+0.2 METEOR, and -0.8/-0.4 TER on ZH–EN and DE–EN, respectively. This improvement is reasonable as our system allows discontinuous phrases which can reduce data sparsity and handle long-distance relations (Galley and Manning, 2010). Another argument for discontinuous phrases is that they allow the decoder to use larger translation units which tend to produce better translations (Galley and Manning, 2010). However, this argument was only verified on ZH–EN. Therefore, we are interested in seeing whether we have the same observation in our experiments on both language pairs.

We count the used translation rules in MT02 and WMT11 based on different target lengths. The results are shown in Figure 5. We find that both DTU and GBMT indeed tend to use larger translation units on ZH–EN. However, more smaller translation units are used on DE–EN.[5] We presume this is because long-distance reordering is performed more often on ZH–EN than on DE–EN. Based on the fact that the distortion function d measures the reordering distance, we find that the average distortion value in PB on ZH–EN MT02 is 18.4 and 3.5 on DE–EN WMT11. Our observations suggest that the argument that discontinuous phrases allow decoders to use larger translation units should be considered with caution when we explain the benefit of discontinuity on different language pairs.

[5]We have the same finding on all test sets.

System	# Rules	
	ZH–EN	DE–EN
DTU	224M+	352M+
GBMT	99M+	153M+

Table 4: The number of rules in DTU and GBMT.

Compared to PBMT, the Treelet system does not show consistent improvements. Our system achieves significantly better BLEU and METEOR scores than Treelet on both ZH–EN and DE–EN, and a better TER score on DE–EN. This suggests that continuous phrases are essential for system robustness since it helps to improve phrase coverage (Hanneman and Lavie, 2009). Lower phrase coverage in Treelet results in more short phrases being used, as shown in Figure 5. In addition, we find that both DTU and our systems do not achieve consistent improvements over Treelet in terms of TER. We observed that both DTU and our systems tend to produce longer translations than Treelet, which might cause unreliable TER evaluation in our experiments as TER favours shorter sentences (He and Way, 2010).

Since discontinuous phrases produced by using syntactic information are fewer in number but

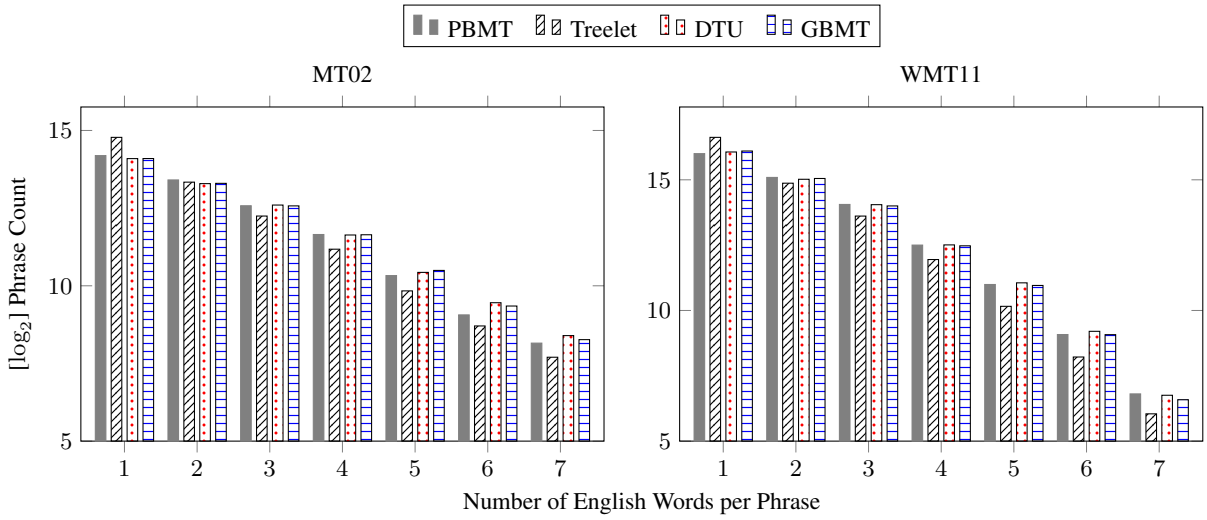


Figure 5: Phrase Length Histogram for MT02 and WMT11.

more reliable (Koehn et al., 2003), our GBMT system achieves comparable performance with DTU but uses significantly fewer rules, as shown in Table 4. After integrating the graph segmentation model to help subgraph selection, GBMT is further improved and the resulted system G2S has significantly better evaluation scores than DTU on both language pairs. However, our segmentation model is more helpful on DE–EN than ZH–EN. We find that the number of features learned on ZH–EN (25K+) is much less than on DE–EN (49K+). This may result in a lower feature coverage during decoding. The lower number of features in ZH–EN could be caused by the fact that the development set MT02 has many fewer sentences than WMT11. Accordingly, we suggest to use a larger development set during tuning to achieve better translation performance when the segmentation model is integrated.

Our current model is more akin to addressing problems in phrase-based and treelet-based models by segmenting graphs into pieces rather than extracting a recursive grammar. Therefore, similar to those models, our model is weak at phrase reordering as well. However, we are interesting in the potential power of our model by incorporating lexical reordering (LR) models and comparing it with syntax-based models.

Table 5 shows BLEU scores of the hierarchical phrase-based (HPB) system (Chiang, 2005) in Moses[6] and GBMT combined with a word-based LR model (Koehn et al., 2005). We find that the LR model significantly improves our system. GBMT+LR is comparable with the Moses HPB model on Chinese–English and better than HPB on German–English.

System	ZH–EN		DE–EN	
	MT04	MT05	WMT12	WMT13
GBMT+LR	36.0	33.9	20.6	23.6
HPB	36.1	34.1	20.3	22.8

Table 5: BLEU scores of a Moses hierarchical phrase-based system (HPB) and our system (GBMT) with a word-based lexical reordering model (LR).

5.3 Examples

Figure 6 shows three examples from MT04 to better explain the differences of each system. Example 1 shows that systems which allow discontinuous phrases (namely Treelet, DTU, GBMT, and GSM) successfully translate a Chinese collocation "*Yu . . . Wuguan*" to "*have nothing to do with*" while PBMT fails to catch the generalization since it only allows continuous phrases.

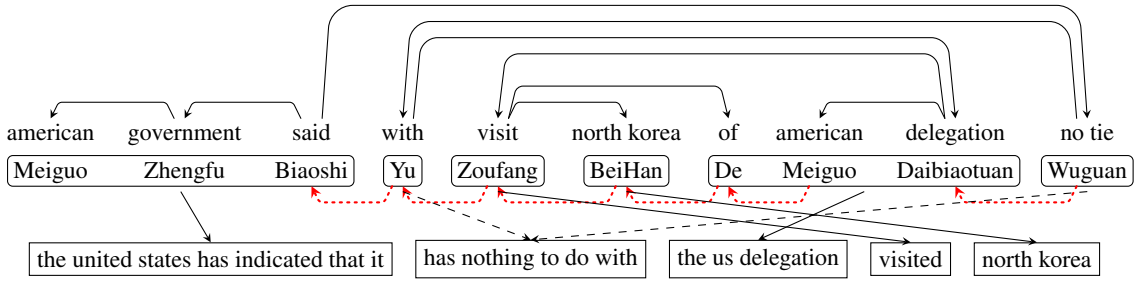
In Example 2, Treelet translates a discontinuous phrase "*Dui . . . Zuofa*" (to . . . practice) only as "*to*" where an important target word "*practice*" is dropped. By contrast, bigram relations allow our systems (GBMT and GSM) to find a better phrase to translate: "*De Zuofa*" to "*of practice*". In addition, DTU translates a discontinuous phrase "*De Zuofa . . . Buman*" to "*dissatisfaction with the approach of*". However, the phrase is actually not

[6]For a fairer comparison, we disallow target discontinuity in HPB rules. This means that a non-terminal on the target side is either the first symbol or the last symbol.

Example 1

PBMT: the united states has indicated that the united states and north korea delegation has visited	*Treelet*: the united states has indicated that it has nothing to do with the us delegation visited the north korea	*DTU*: the united states has indicated that it has nothing to do with the us delegation visited north korea	*GBMT*: the united states has indicated that it has nothing to do with the us delegation visited north korea	*GSM*: the united states has indicated that it has nothing to do with the us delegation visited north korea

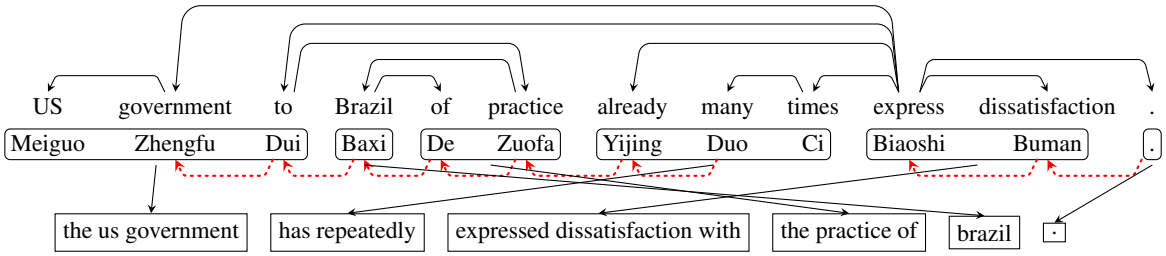
REF: the american government said that it has nothing to do with the american delegation to visit north korea



Example 2

PBMT: the united states government to brazil has repeatedly expressed its dissatisfaction .	*Treelet*: the government of brazil to the united states has on many occasions expressed their discontent .	*DTU*: the united states has repeatedly expressed its dissatisfaction with the approach of the government to brazil .	*GBMT*: the us government has repeatedly expressed dissatisfaction with the practice of brazil .	*GSM*: the us government has repeatedly expressed dissatisfaction with the practice of brazil .

REF: the us government has expressed their resentment against this practice of brazil on many occasions .



Example 3

PBMT: the government and all sectors of society should continue to explore in depth and draw on collective wisdom .	*Treelet*: the government must continue to make in-depth discussions with various sectors of the community and the collective wisdom .	*DTU*: the government must continue to work together with various sectors of the community to make an in-depth study and draw on collective wisdom .	*GBMT*: the government must continue to work together with various sectors of the community in-depth study and draw on collective wisdom .	*GSM*: the government must continue to make in-depth discussions with various sectors of the community and draw on collective wisdom .

REF: the government must continue to hold thorough discussions with all walks of life to pool the wisdom of the masses .

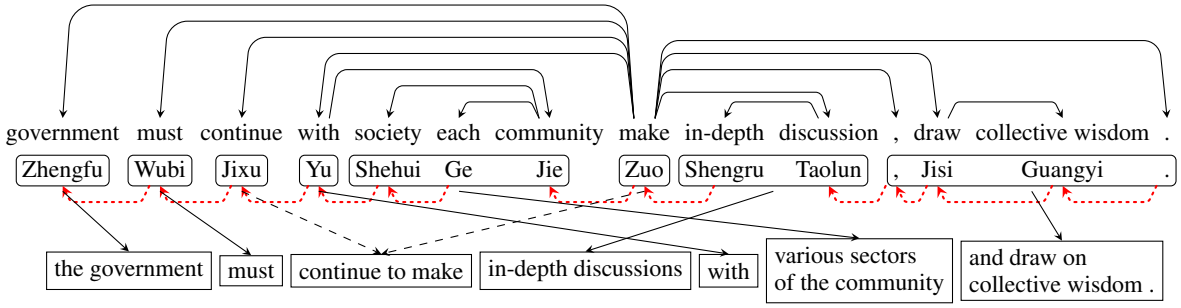


Figure 6: Translation examples from MT04 produced by different systems. Each source sentence is annotated by dependency relations and additional bigram relations (dotted red edges). We also annotate phrase alignments produced by our system GSM.

linguistically motivated and could be unreliable. By disallowing phrases which are not connected in the input graph, GBMT and GSM produce better translations.

Example 3 illustrates that our graph segmentation model helps to select better subgraphs. After obtaining a partial translation "*the government must*", GSM chooses to translate a subgraph which covers a discontinuous phrase "*Jixu ... Zuo*" to "*continue to make*" while GBMT translates "*Jixu Yu*" (continue ... with) to "*continue to work together with*". By selecting the proper subgraph to translate, GSM performs a better reordering on the translation.

6 Related Work

Starting from sequence-based models, SMT has been benefiting increasingly from complex structures.

Sequence-based MT: Since the breakthrough made by IBM on word-based models in the 1990s (Brown et al., 1993), SMT has developed rapidly. The PB model (Koehn et al., 2003) advanced the state-of-the-art by translating multi-word units, which makes it better able to capture local phenomena. However, a major drawback in PBMT is that only continuous phrases are considered. Galley and Manning (2010) extend PBMT by allowing discontinuity. However, without linguistic structure information such as syntax trees, sequence-based models can learn a large amount of phrases which may be unreliable.

Tree-based MT: Compared to sequences, trees provide recursive structures over sentences and can handle long-distance relations. Typically, trees used in SMT are either phrasal structures (Galley et al., 2004; Liu et al., 2006; Marcu et al., 2006) or dependency structures (Menezes and Quirk, 2005; Xiong et al., 2007; Xie et al., 2011; Li et al., 2014). However, conventional tree-based models only use linguistically well-formed phrases. Although they are more reliable in theory, discarding all phrase pairs which are not linguistically motivated is an overly harsh decision. Therefore, exploring more translation rules usually can significantly improve translation performance (Marcu et al., 2006; DeNeefe et al., 2007; Wang et al., 2007; Mi et al., 2008).

Graph-based MT: Compared to sequences and trees, graphs are more general and can represent more relations between words. In recent years, graphs have been drawing quite a lot of attention from researchers. Jones et al. (2012) propose a hypergraph-based translation model where hypergraphs are taken as a meaning representation of sentences. However, large corpora with annotated hypergraphs are not readily available for MT. Li et al. (2015) use an edge replacement grammar to translate dependency graphs which are converted from dependency trees by labeling edges. However, their model only focuses on subgraphs which cover continuous phrases.

7 Conclusion

In this paper, we extend the conventional phrase-based translation model by allowing discontinuous phrases. We use graphs which combine bigram and dependency relations together as inputs and present a graph-based translation model. Experiments on Chinese–English and German–English show our model to be significantly better than the phrase-based model as well as other more sophisticated models. In addition, we present a graph segmentation model to explicitly guide the selection of subgraphs. In experiments, this model further improves our system.

In the future, we will extend this model to allow discontinuity on target sides and explore the possibility of directly encoding reordering information in translation rules. We are also interested in using graphs for neural machine translation to see how it can translate and benefit from graphs.

Acknowledgments

This research has received funding from the People Programme (Marie Curie Actions) of the European Union's Framework Programme (FP7/2007-2013) under REA grant agreement n^{o} 317471 and the European Union's Horizon 2020 research and innovation programme under grant agreement n^{o} 645452 (QT21). The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund. The authors thank all anonymous reviewers for their insightful comments and suggestions.

References

Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is Not a Contradiction. In *Proceedings of the 23rd International Conference on*

Computational Linguistics, pages 89–97, Beijing, China, August.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese Word Segmentation for Machine Translation Performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232, Columbus, Ohio, June.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative Reordering with Chinese Grammatical Relations Features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59, Boulder, Colorado, June.

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 310–318, Santa Cruz, California, June.

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436, Montreal, Canada, June.

Colin Cherry. 2013. Improved Reordering for Phrase-Based Translation using Sparse Features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 22–31, Atlanta, Georgia, June.

David Chiang. 2005. A Hierarchical Phrase-based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 263–270, Ann Arbor, Michigan, June.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better Hypothesis Testing for Statistical Machine Translation: Controlling for Optimizer Instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, pages 176–181, Portland, Oregon, June.

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What Can Syntax-Based MT Learn from Phrase-Based MT? In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 755–763, Prague, Czech Republic, June.

Michael Denkowski and Alon Lavie. 2011. Meteor 1.3: Automatic Metric for Reliable Optimization and Evaluation of Machine Translation Systems. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 85–91, Edinburgh, Scotland, July.

Michel Galley and Christopher D. Manning. 2010. Accurate Non-hierarchical Phrase-Based Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 966–974, Los Angeles, California, June.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a Translation Rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, page 273280, Boston, Massachusetts, USA, May.

Greg Hanneman and Alon Lavie. 2009. Decoding with Syntactic and Non-syntactic Phrases in a Syntax-based Machine Translation System. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 1–9, Boulder, Colorado, June.

Yifan He and Andy Way. 2010. Metric and reference factors in minimum error rate training. *Machine Translation*, 24(1):27–38.

Bevan Jones, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, and Kevin Knight. 2012. Semantics-Based Machine Translation with Hyperedge Replacement Grammars. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers*, pages 1359–1376, Mumbai, India, December.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54, Edmonton, Canada, July.

Philipp Koehn, Amittai Axelrod, Alexandra Birch, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh System Description for the 2005 IWSLT Speech Translation Evaluation. In *Proceedings of the International Workshop on Spoken Language Translation 2005*, pages 68–75, Pittsburgh, PA, USA, October.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration*

Sessions, pages 177–180, Prague, Czech Republic, June.

Philipp Koehn. 2010. *Statistical Machine Translation*. Cambridge University Press, New York, NY, USA, 1st edition.

Liangyou Li, Jun Xie, Andy Way, and Qun Liu. 2014. Transformation and Decomposition for Efficiently Implementing and Improving Dependency-to-String Model In Moses. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, October.

Liangyou Li, Andy Way, and Qun Liu. 2015. Dependency Graph-to-String Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal, September.

Lemao Liu and Liang Huang. 2014. Search-Aware Tuning for Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1942–1952, Doha, Qatar, October.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string Alignment Template for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616, Sydney, Australia, July.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical Machine Translation with Syntactified Target Language Phrases. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 44–52, Sydney, Australia.

Arul Menezes and Chris Quirk. 2005. Dependency Treelet Translation: The Convergence of Statistical and Example-Based Machine-translation? In *Proceedings of the Workshop on Example-based Machine Translation at MT Summit X*, September.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-Based Translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 192–199, Columbus, Ohio, USA, June.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106, Ann Arbor, Michigan, June.

Franz Josef Och and Hermann Ney. 2002. Discriminative Training and Maximum Entropy Models for Statistical Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 295–302, Philadelphia, PA, USA, July.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

Franz Josef Och and Hermann Ney. 2004. The Alignment Template Approach to Statistical Machine Translation. *Computational Linguistics*, 30(4):417–449, December.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, July.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, Michigan, June.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A Study of Translation Edit Rate with Targeted Human Annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, Massachusetts, USA, August.

Andreas Stolcke. 2002. SRILM An Extensible Language Modeling Toolkit. In *Proceedings of the International Conference Spoken Language Processing*, pages 901–904, Denver, CO.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing Syntax Trees to Improve Syntax-Based Machine Translation Accuracy. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 746–754, Prague, Czech Republic, June.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A Novel Dependency-to-string Model for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 216–226, Edinburgh, United Kingdom, July.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2006. Maximum Entropy Based Phrase Reordering Model for Statistical Machine Translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 521–528, Sydney, Australia, July.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A Dependency Treelet String Correspondence Model for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47, Prague, June.

Incremental Acquisition of Verb Hypothesis Space towards Physical World Interaction

Lanbo She and **Joyce Y. Chai**
Department of Computer Science and Engineering
Michigan State University
East Lansing, Michigan 48824, USA
{shelanbo, jchai}@cse.msu.edu

Abstract

As a new generation of cognitive robots start to enter our lives, it is important to enable robots to follow human commands and to learn new actions from human language instructions. To address this issue, this paper presents an approach that explicitly represents verb semantics through hypothesis spaces of fluents and automatically acquires these hypothesis spaces by interacting with humans. The learned hypothesis spaces can be used to automatically plan for lower-level primitive actions towards physical world interaction. Our empirical results have shown that the representation of a hypothesis space of fluents, combined with the learned hypothesis selection algorithm, outperforms a previous baseline. In addition, our approach applies incremental learning, which can contribute to life-long learning from humans in the future.

1 Introduction

As a new generation of cognitive robots start to enter our lives, it is important to enable robots to follow human commands (Tellex et al., 2014; Thomason et al., 2015) and to learn new actions from human language instructions (Cantrell et al., 2012; Mohan et al., 2013). To achieve such a capability, one of the fundamental challenges is to link higher-level concepts expressed by human language to lower-level primitive actions the robot is familiar with. While grounding language to perception (Gorniak and Roy, 2007; Chen and Mooney, 2011; Kim and Mooney, 2012; Artzi and Zettlemoyer, 2013; Tellex et al., 2014; Liu et al., 2014; Liu and Chai, 2015) has received much attention in recent years, less work has addressed grounding language to robotic action. Actions are often expressed by verbs or verb phrases. Most semantic representations for verbs are based on argument frames (e.g., thematic roles which capture participants of an action). For example, suppose a human directs a robot to "*fill the cup with milk*". The robot will need to first create a semantic representation for the verb "*fill*" where "*the cup*" and "*milk*" are grounded to the respective objects in the environment (Yang et al., 2016). Suppose the robot is successful in this first step, it still may not be able to execute the action "*fill*" as it does not know how this higher-level action corresponds to its lower-level primitive actions.

In robotic systems, operations usually consist of multiple segments of lower-level primitive actions (e.g., *move to*, *open gripper*, and *close gripper*) which are executed both sequentially and concurrently. Task scheduling provides the order or schedule for executions of different segments of actions and action planning provides the plan for executing each individual segment. Primitive actions are often predefined in terms of how they change the state of the physical world. Given a goal, task scheduling and action planning will derive a sequence of primitive actions that can change the initial environment to the goal state. The goal state of the physical world becomes a driving force for robot actions. Thus, beyond semantic frames, modeling verb semantics through their effects on the state of the world may provide a link to connect higher-level language and lower-level primitive actions.

Motivated by this perspective, we have developed an approach where each verb is explicitly represented by a hypothesis space of fluents (i.e., desired goal states) of the physical world, which is incrementally acquired and updated through interacting with humans. More specifically, given a human command, if there is no knowledge about the

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 108–117,
Berlin, Germany, August 7-12, 2016.

corresponding verb (i.e., no existing hypothesis space for that verb), the robot will initiate a learning process by asking human partners to demonstrate the sequence of actions that is necessary to accomplish this command. Based on this demonstration, a hypothesis space of fluents for that verb frame will be automatically acquired. If there is an existing hypothesis space for the verb, the robot will select the best hypothesis that is most relevant to the current situation and plan for the sequence of lower-level actions. Based on the outcome of the actions (e.g., whether it has successfully executed the command), the corresponding hypothesis space will be updated. Through this fashion, a hypothesis space for each encountered verb frame is incrementally acquired and updated through continuous interactions with human partners. In this paper, to focus our effort on representations and learning algorithms, we adopted an existing benchmark dataset (Misra et al., 2015) to simulate the incremental learning process and interaction with humans.

Compared to previous works (She et al., 2014b; Misra et al., 2015), our approach has three unique characteristics. First, rather than a single goal state associated with a verb, our approach captures a space of hypotheses which can potentially account for a wider range of novel situations when the verb is applied. Second, given a new situation, our approach can automatically identify the best hypothesis that fits the current situation and plan for lower-level actions accordingly. Third, through incremental learning and acquisition, our approach has a potential to contribute to life-long learning from humans. This paper provides details on the hypothesis space representation, the induction and inference algorithms, as well as experiments and evaluation results.

2 Related Work

Our work here is motivated by previous linguistic studies on verbs, action modeling in AI, and recent advances in grounding language to actions.

Previous linguistic studies (Hovav and Levin, 2008; Hovav and Levin, 2010) propose action verbs can be divided into two types: ***manner verbs*** that "specify as part of their meaning a manner of carrying out an action" (e.g., *nibble, rub, laugh, run, swim*), and ***result verbs*** that "specify the coming about of a result state" (e.g., *clean, cover, empty, fill, chop, cut, open, enter*). Recent work has shown that explicitly modeling resulting change of state for action verbs can improve grounded language understanding (Gao et al., 2016). Motivated by these studies, this paper focuses on result verbs and uses hypothesis spaces to explicitly represent the result states associated with these verbs.

In AI literature on action modeling, action schemas are defined with preconditions and effects. Thus, representing verb semantics for action verbs using resulting states can be connected to the agent's underlying planning modules. Different from earlier works in the planning community that learn action models from example plans (Wang, 1995; Yang et al., 2007) and from interactions (Gil, 1994), our goal here is to explore the representation of verb semantics and its acquisition through language and action.

There has been some work in the robotics community to translate natural language to robotic operations (Kress-Gazit et al., 2007; Jia et al., 2014; Sung et al., 2014; Spangenberg and Henrich, 2015), but not for the purpose of learning new actions. To support action learning, previously we have developed a system where the robot can acquire the meaning of a new verb (e.g., *stack*) by following human's step-by-step language instructions (She et al., 2014a; She et al., 2014b). By performing the actions at each step, the robot is able to acquire the desired goal state associated with the new verb. Our empirical results have shown that representing acquired verbs by resulting states allow the robot to plan for primitive actions in novel situations. Moreover, recent work (Misra et al., 2014; Misra et al., 2015) has presented an algorithm for grounding higher-level commands such as "microwave the cup" to lower-level robot operations, where each verb lexicon is represented as the desired resulting states. Their empirical evaluations once again have shown the advantage of representing verbs as desired states in robotic systems. Different from these previous works, we represent verb semantics through a hypothesis space of fluents (rather than a single hypothesis). In addition, we present an incremental learning approach for inducing the hypothesis space and selecting the best hypothesis.

3 An Incremental Learning Framework

An overview of our incremental learning framework is shown in Figure 1. Given a language

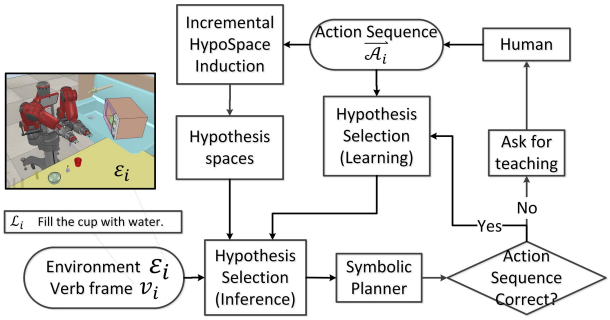


Figure 1: An incremental process of verb acquisition (i.e. learning) and application (i.e. inference).

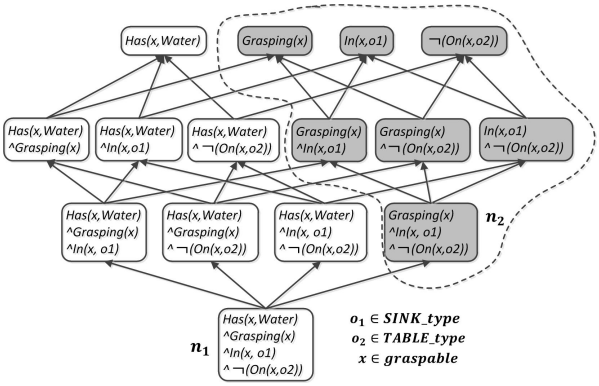


Figure 2: An example hypothesis space for the verb frame `fill(x)`. The bottom node captures the state changes after executing the *fill* command in the environment. Anchored by the bottom node, the hypothesis space is generated in a bottom-up fashion. Each node represents a potential goal state. The highlighted nodes are pruned during induction, as they are not consistent with the bottom node.

command $\mathcal{L}_i$ (e.g. *"fill the cup with water."*) and an environment $\mathcal{E}_i$ (e.g. a simulated environment shown in Figure 1), the goal is to identify a sequence of lower-level robotic actions to perform the command. Similar to previous works (Pasula et al., 2007; Mouro et al., 2012), the environment $\mathcal{E}_i$ is represented by a conjunction of grounded state fluents, where each fluent describes either the property of an object or relations (e.g. spatial) between objects. The language command $\mathcal{L}_i$ is first translated to an intermediate representation of grounded verb frame v_i through semantic parsing and referential grounding (e.g. for "*fill the cup*", the argument *the cup* is grounded to `Cup1` in the scene). The system knowledge of each verb frame (e.g., `fill(x)`) is represented by a *Hypothesis Space* $\mathcal{H}$, where each hypothesis (i.e. a node) is a description of possible fluents - or, in other words, resulting states - that are attributed to executing the verb command. Given a verb frame v_i and an environment $\mathcal{E}_i$, a *Hypothesis Selector* will choose an optimal hypothesis from space $\mathcal{H}$ to describe the expected resulting state of executing v_i in $\mathcal{E}_i$. Given this goal state and the current environment, a symbolic planner such as the STRIPS planner (Fikes and Nilsson, 1971) is used to generate an action sequence for the agent to execute. If the action sequence correctly performs the command (e.g. as evaluated by a human partner), the hypothesis selector will be updated with the success of its prediction. On the other hand, if the action has never been encountered (i.e., the system has no knowledge about this verb and thus the corresponding space is empty) or the predicted action sequence is incorrect, the human partner will provide an action sequence $\vec{A}_i$ that can correctly perform command v_i in the current environment. Using $\vec{A}_i$ as the ground truth information, the system will not only update the hypothesis selector, but will also update the existing space of v_i. The updated hypothesis space is treated as system knowledge of v_i, which will be used in future interaction. Through this procedure, a hypothesis space for each verb frame v_i is continually and incrementally updated through human-robot interaction.

4 State Hypothesis Space

To bridge human language and robotic actions, previous works have studied representing the semantics of a verb with a single resulting state (She et al., 2014b; Misra et al., 2015). One problem of this representation is that when the verb is applied in a new situation, if any part of the resulting state cannot be satisfied, the symbolic planner will not be able to generate a plan for lower-level actions to execute this verb command. The planner is also not able to determine whether the failed part of state representation is even necessary. In fact, this effect is similar to the over-fitting problem. For example, given a sequence of actions of performing `fill(x)`, the induced hypothesis could be "$Has(x, Water) \wedge Grasping(x) \wedge In(x, o_1) \wedge \neg(On(x, o_2))$", where x is a graspable object (e.g. a cup or bowl), o_1 is any type of sink, and o_2 is any table. However, during inference, when applied to a new situation that does not have any type of sink or table, this hypothesis will not

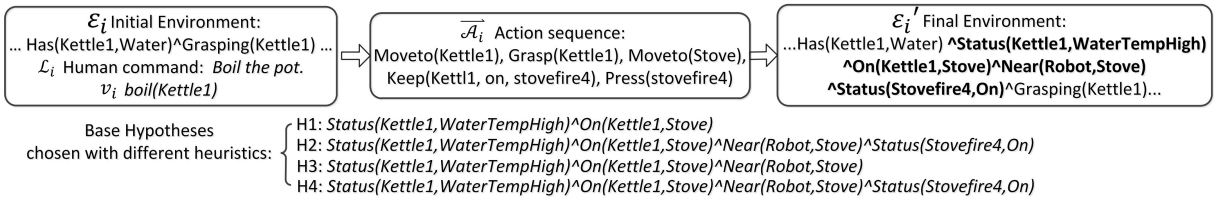


Figure 3: A training instance $\{\mathcal{E}_i, v_i, \vec{\mathcal{A}}_i\}$ for hypothesis space induction. $\mathcal{E}_i'$ is the resulting environment of executing $\vec{\mathcal{A}}_i$ in $\mathcal{E}_i$. The change of state in $\mathcal{E}_i'$ compared to $\mathcal{E}_i$ is highlighted in bold. Different heuristics generate different Base Hypotheses as shown at the bottom.

be applicable. Nevertheless, the first two terms $Has(x, Water) \wedge Grasping(x)$ may already be sufficient to generate a plan for completing the verb command.

To handle this over-fitting problem, we propose a hierarchical hypothesis space to represent verb semantics, as shown in Figure 2. The space is organized based on a specific-to-general hierarchical structure. Formally, a hypothesis space $\mathcal{H}$ for a verb frame is defined as: $\langle \mathbb{N}, \mathbb{E} \rangle$, where each $n_i \in \mathbb{N}$ is a hypothesis node and each $e_{ij} \in \mathbb{E}$ is a directed edge pointing from parent n_i to child n_j, meaning node n_j is more general than n_i and has one less constraint.

In Figure 2, the bottom hypothesis (n_1) is $Has(x, Water) \wedge Grasping(x) \wedge In(x, o1) \wedge \neg(On(x, o2))$. A hypothesis n_i represents a conjunction of parameterized state fluents l_k:

$$n_i := \wedge l_k, and\, l_k := [\neg]\, pred_k(x_{k_1}[, x_{k_2}])$$

A fluent l_k is composed of a predicate (e.g. object status: *Has,* or spatial relation: *On*) and a set of argument variables. It can be positive or negative. Take the bottom node in Figure 2 as an example, it contains four fluents including one negative term (i.e. $\neg(On(x, o_2))$) and three positive terms. During inference, the parameters will be grounded to the environment to check whether this hypothesis is applicable.

5 Hypothesis Space Induction

Given an initial environment $\mathcal{E}_i$, a language command which contains the verb frame v_i, and a corresponding action sequence $\vec{\mathcal{A}}_i$, $\{\mathcal{E}_i, v_i, \vec{\mathcal{A}}_i\}$ forms a training instance for hypothesis space induction. First, based on different heuristics, a base hypothesis is generated by comparing the state difference between the final and the initial environment. Second, a hypothesis space $\mathcal{H}$ is induced on top of this *Base Hypothesis* in a bottom-up fashion. And during induction some nodes are pruned. Third, if the system has existing knowledge for the same verb frame (i.e. an existing hypothesis space $\mathcal{H}_t$ for the same verb frame), this newly induced space will be merged with previous knowledge. Next we explain each step in detail.

5.1 Base Hypothesis Induction

One key concept in the space induction is the *Base Hypothesis* (e.g. the bottom node in Figure 2), which provides a foundation for building a space. As shown in Figure 3, given a verb frame v_i and a working environment $\mathcal{E}_i$, the action sequence $\vec{\mathcal{A}}_i$ given by a human will change the initial environment $\mathcal{E}_i$ to a final environment $\mathcal{E}_i'$. The state changes are highlighted in Figure 3. Suppose a state change can be described by n fluents. Then the first question is which of these n fluents should be included in the base hypothesis. To gain some understanding on what would be a good representation, we applied different heuristics of choosing fluents to form a base hypothesis as shown in Figure 3:

- $H1_{argonly}$: only includes the changed states associated with the argument objects specified in the frame (e.g., in Figure 3, `Kettle1` is the only argument).
- $H2_{manip}$: includes the changed states of all the objects that have been manipulated in the action sequence taught by the human.
- $H3_{argrelated}$: includes the changed states of all the objects related to the argument objects in the final environment. An object o is considered as "related to" an argument object if there is a state fluent that includes both o and an argument object in one predicate. (e.g. `Stove` is related to the argument object `Kettle1` through $On(Kettle1, Stove)$).

Input: A Base Hypothesis h
Initialization: Set initial space $\mathcal{H}$: $\langle \mathbb{N}, \mathbb{E} \rangle$ with $\mathbb{N}$:[h] and $\mathbb{E}$:[ ],
Set a set of temporary hypotheses T:[h]
while T is not empty **do**
  Pop an element t from T
  Generate children [$t^{(0)}$,...,$t^{(k)}$] from t by removing each single fluent
  foreach i = 0 ... k **do**
    if $t^{(i)}$ is consistent with t **then**
      Append $t^{(i)}$ to T;
      Add $t^{(i)}$ to $\mathbb{N}$ if not already in;
      Add link $t \rightarrow t^{(i)}$ to $\mathbb{E}$ if not already in;
    else
      Prune $t^{(i)}$ and any node that can be generalized from $t^{(i)}$
    end
  end
end
Output: Hypothesis space $\mathcal{H}$

Algorithm 1: A single hypothesis space induction algorithm. $\mathcal{H}$ is a space initialized with a base hypothesis and an empty set of links. T is a temporary container of candidate hypotheses.

- $H4_{all}$: includes all the fluents whose values are changed from $\mathcal{E}_i$ to $\mathcal{E}_i'$ (e.g. all the four highlighted state fluents in $\mathcal{E}_i'$).

5.2 Single Space Induction

First we define the ***consistency*** between two hypotheses:

Definition. Hypotheses h_1 and h_2 are ***consistent***, if and only if the action sequence $\vec{\mathcal{A}}_1$ generated from a symbolic planner based on goal state h_1 is exactly the same as the action sequence $\vec{\mathcal{A}}_2$ generated based on goal state h_2.

Given a base hypothesis, the space induction process is a while-loop generalizing hypotheses in a bottom-up fashion, which stops when no hypotheses can be further generalized. As shown in Algorithm 1, a hypothesis node t can firstly be generalized to a set of immediate children [$t^{(0)}$,...,$t^{(k)}$] by removing a single fluent from t. For example, the base hypothesis n_1 in Figure 2 is composed of 4 fluents, such that 4 immediate children nodes can potentially be generated. If a child node $t^{(i)}$ is consistent with its parent t (i.e. determined based on the *consistency* defined previously), node $t^{(i)}$ and a link $t \rightarrow t^{(i)}$ are added to the space $\mathcal{H}$. The node $t^{(i)}$ is also added to a temporary hypothesis container waiting to be further generalized. On the other hand, some children hypotheses can be inconsistent with their parents. For example, the gray node (n_2) in Figure 2 is a child node that is inconsistent with its parent (n_1). As n_2 does not explicitly specify $Has(x, Water)$ as part of its goal state, the symbolic planner generates less steps to achieve goal state n_2 than goal state n_1. This implies that the semantics of achieving n_2 may be different than those for achieving n_1. Such hypotheses that are inconsistent with their parents are pruned. In addition, if $t^{(i)}$ is inconsistent with its parent t, any children of $t^{(i)}$ are also inconsistent with t (e.g. children of n_2 in Figure 2 are also gray nodes, meaning they are inconsistent with the base hypothesis). Through pruning, the size of entire space can be greatly reduced.

In the resulting hypothesis space, every single hypothesis is consistent with the base hypothesis. By only keeping consistent hypotheses via pruning, we can remove fluents that are not representative of the main goal associated with the verb.

5.3 Space Merging

If the robot has existing knowledge (i.e. hypothesis space $\mathcal{H}_t$) for a verb frame, the induced hypothesis space $\mathcal{H}$ from a new instance of the same verb will be merged with the existing space $\mathcal{H}_t$. Currently, a new space $\mathcal{H}_{t+1}$ is generated where the nodes of $\mathcal{H}_{t+1}$ are the union of $\mathcal{H}$ and $\mathcal{H}_t$, and links in $\mathcal{H}_{t+1}$ are generated by checking the parent-child relationship between nodes. In future work, more space merging operations will be explored, and human feedback will be incorporated into the induction process.

6 Hypothesis Selection

Hypothesis selection is applied when the agent intends to execute a command. Given a verb frame extracted from the language command, the agent will first select the best hypothesis (describing the goal state) from the existing knowledge base, and then apply a symbolic planner to generate an action sequence to achieve the goal. In our framework, the model of selecting the best hypothesis is incrementally learned throughout continuous interaction with humans. More specifically, given a correct action sequence (whether performed by the robot or provided by the human), a regression model is trained to capture the fitness of a hypothesis given a particular situation.

Inference: Given a verb frame v_i and a working environment $\mathcal{E}_i$, the goal of inference is to estimate how well each hypothesis h_k from a space $\mathcal{H}_t$ describes the expected result of performing v_i

in $\mathcal{E}_i$. The best fit hypothesis will be used as the goal state to generate the action sequence. Specifically, the "goodness" of describing command v_i with hypothesis h_k in environment $\mathcal{E}_i$ is formulated as follows:

$$f(h_k \mid v_i; \mathcal{E}_i; \mathcal{H}_t) = W^T \cdot \Phi(h_k, v_i, \mathcal{E}_i, \mathcal{H}_t) \quad (1)$$

where $\Phi(h_k, v_i, \mathcal{E}_i, \mathcal{H}_t)$ is a feature vector capturing multiple aspects of relations between $h_k, v_i, \mathcal{E}_i$ and $\mathcal{H}_t$ as shown in Table 1; and W captures the weight associated with each feature. Example global features include whether the candidate goal h_k is in the top level of entire space $\mathcal{H}_t$ and whether h_k has the highest frequency. Example local features include if most of the fluents in h_k are already satisfied in current scene $\mathcal{E}_i$ (as this h_k is unlikely to be a desired goal state). The features also include whether the same verb frame v_i has been performed in a similar scene during previous interactions, as the corresponding hypotheses induced during that experience are more likely to be relevant and are thus preferred.

Parameter Estimation: Given an action sequence $\vec{\mathcal{A}}_i$ that illustrates how to correctly perform command v_i in environment $\mathcal{E}_i$ during interaction, the model weights will be incrementally updated with[1]:

$$W_{t+1} = W_t - \eta(\alpha \frac{\partial R(W_t)}{\partial W_t} + \frac{\partial L(J_{ki}, f_{ki})}{\partial W_t})$$

where $f_{ki} := f(h_k|v_i; \mathcal{E}_i; \mathcal{H}_t)$ is defined in Equation 1. J_{ki} is the dependent variable the model should approximate, where $J_{ki} := J(s_i, h_k)$ is the Jaccard Index (details in Section 7) between hypothesis h_k and a set of changed states s_i (i.e. the changed states of executing the illustration action sequence $\vec{\mathcal{A}}_i$ in current environment). $L(J_{ki}, f_{ki})$ is a squared loss function. $\alpha R(W_t)$ is the penalty term, and η is the constant learning rate.

7 Experiment Setup

Dataset Description. To evaluate our approach, we applied the dataset made available by (Misra et al., 2015). To support incremental learning, each utterance from every original paragraph is extracted so that each command/utterance only contains one verb and its arguments. The corresponding initial environment and an action sequence taught by a human for each command are also extracted. An example is shown in Figure 3, where $\mathcal{L}_i$ is a language command, $\mathcal{E}_i$ is the initial working environment, and $\vec{\mathcal{A}}_i$ is a sequence of primitive actions to complete the command given by the human. In the original data, some sentences are not aligned with any actions, and thus cannot be used for either the learning or the evaluation. Removing these unaligned sentences resulted in a total of 991 data instances, including 165 different verb frames.

Features on candidate hypothesis h_k and the space $\mathcal{H}_t$ 1. If h_k belongs to the top level of $\mathcal{H}_t$. 2. If h_k has the highest frequency in $\mathcal{H}_t$.
Features on h_k and current situation $\mathcal{E}_i$ 3. Portion of fluents in h_k that are already satisfied by $\mathcal{E}_i$. 4. Portion of non-argument objects in h_k. Examples of non-argument objects are o_1 and o_2 in Figure 2.
Features on relations between a testing verb frame v_i and previous interaction experience 5. Whether the same verb frame v_i has been executed previously with the same argument objects. 6. Similarities between noun phrase descriptions used in current command and commands from interaction history.

Table 1: Current features used for incremental learning of the regression model. The first two are binary features and the rest are real-valued features.

Among the 991 data instances, 793 were used for incremental learning (i.e., space induction and hypothesis selector learning). Specifically, given a command, if the robot correctly predicts an action sequence[2], this correct prediction is used to update the hypothesis selector. Otherwise, the agent will require a correct action sequence from the human, which is used for hypothesis space induction as well as updating the hypothesis selector.

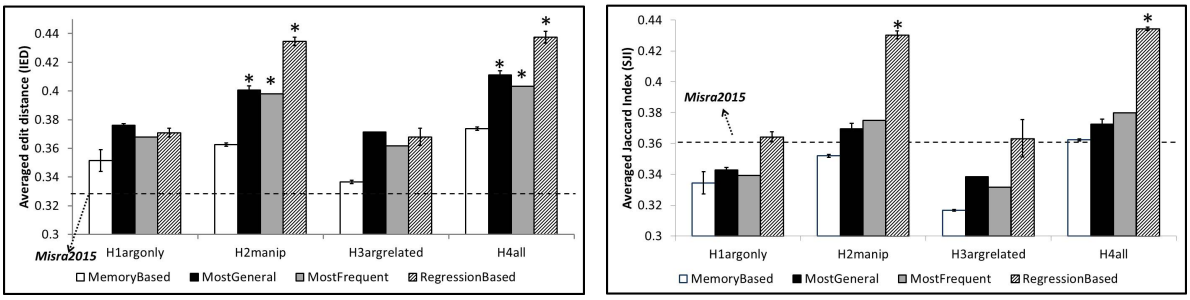
The hypothesis spaces and regression based selectors acquired at each run were evaluated on the other 20% (198) testing instances. Specifically, for each testing instance, the induced space and the hypothesis selector were applied to identify a desired goal state. Then a symbolic planner[3] was applied to predict an action sequence $\vec{\mathcal{A}}^{(p)}$ based on this predicted goal state. We then compared $\vec{\mathcal{A}}^{(p)}$ with the ground truth action sequence $\vec{\mathcal{A}}^{(g)}$ using the following two metrics.

- IED (*Instruction Edit Distance*) measures

[1]The SGD regressor in the scikit-learn (Pedregosa et al., 2011) is used to perform the linear regression with L2 regularization.

[2]Currently, a prediction is considered correct if the predicted result ($c^{(p)}$) is similar to a human labeled action sequence ($c^{(g)}$) (i.e., $SJI(c^{(g)}, c^{(p)}) > 0.5$).

[3]The symbolic planner implemented by (Rintanen, 2012) was utilized to generate action sequences.



(a) IED results for different configurations

(b) SJI results for different configurations

Figure 4: The overall performance on the testing set with different configurations in generating the base hypothesis and in hypothesis selection. Each configuration runs five times by randomly shuffling the order of learning instances, and the averaged performance is reported. The result from *Misra2015* is shown as a line. Results that are statistically significant better than *Misra2015* are marked with $*$ (*paired t-test,* $p< 0.05$).

similarity between the ground truth action sequence $\vec{\mathcal{A}}^{(g)}$ and the predicted sequence $\vec{\mathcal{A}}^{(p)}$. Specifically, the edit distance d between two action sequences $\vec{\mathcal{A}}^{(g)}$ and $\vec{\mathcal{A}}^{(p)}$ is first calculated. Then d is rescaled as $IED = 1 - d/max(\vec{\mathcal{A}}^{(g)}, \vec{\mathcal{A}}^{(p)})$, such that IED ranges from 0 to 1 and a larger IED means the two sequences are more similar.

- SJI (*State Jaccard Index*). Because different action sequences could lead to a same goal state, we also use Jaccard Index to check the overlap between the changed states. Specifically, executing the ground truth action sequence $\vec{\mathcal{A}}^{(g)}$ in the initial scene $\mathcal{E}_i$ results in a final environment $\mathcal{E}_i'$. Suppose the changed states between $\mathcal{E}_i$ and $\mathcal{E}_i'$ is $c^{(g)}$. For the predicted action sequence, we can calculate another set of changed states $c^{(p)}$. The Jaccard Index between $c^{(g)}$ and $c^{(p)}$ is evaluated, which also ranges from 0 to 1 and a larger SJI means the predicted state changes are more similar to the ground truth.

Configurations. We also compared the results of using the regression based selector to select a hypothesis (i.e., *RegressionBased*) with the following different strategies for selecting the hypothesis:

- *Misra2015*: The state of the art system reported in (Misra et al., 2015) on the command/utterance level evaluation[4].

- *MemoryBased*: Given the induced space, only the base hypotheses h_ks from each learning instances are used. Because these h_ks don't have any relaxation, they represent purely learning from memorization.

- *MostGeneral*: In this case, only those hypotheses from the top level of the hypothesis space are used, which contain the least number of fluents. These nodes are the most relaxed hypotheses in the space.

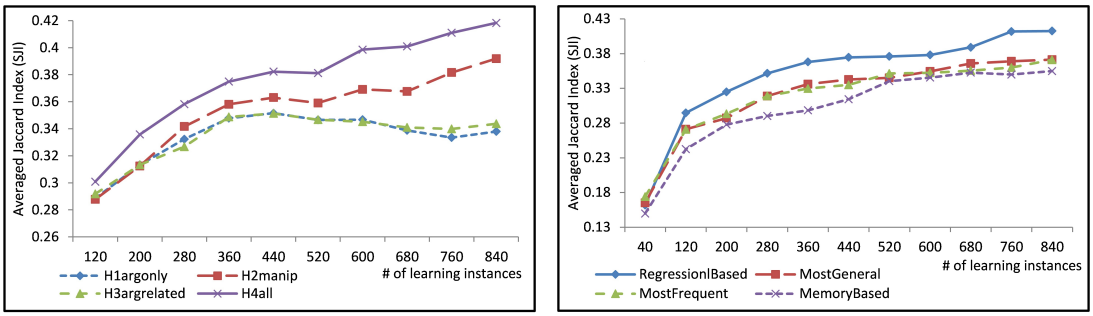
- *MostFrequent*: In this setting, the hypotheses that are most frequently observed in the learning instances are used.

8 Results

8.1 Overall performance

The results of the overall performance across different configurations are shown in Figure 4. For both of the IED and SJI (i.e. Figure 4(a) and Figure 4(b)), the hypothesis spaces with the regression model based hypothesis selector always achieve the best performance across different configurations, and outperforms the previous approach (Misra et al., 2015). For different base hypothesis induction strategies, the $H4_{all}$ considering all the changed states achieves the best performance across all configurations. This is because $H4_{all}$ keeps all of the state change information compared with other heuristics. The performance of $H2_{manip}$ is similar to $H4_{all}$. The reason is that, when all the manipulated objects are considered, the resulted set of changed states will cover most of the fluents in $H4_{all}$. On the other dimension,

[4]We applied the same system described in (Misra et al., 2015) to predict action sequences. The only difference is here we report the performance at the command level, not at the paragraph level.



(a) Use regression based selector to select hypothesis, and compare each base hypothesis induction heuristics.

(b) Induce the base hypothesis with $H4_{all}$, and compare different hypothesis selection strategies.

Figure 5: Incremental learning results. The spaces and regression models acquired at different incremental learning cycles are evaluated on testing set. The averaged Jaccard Index is reported.

the regression based hypothesis selector achieves the best performance and the *MemoryBased* strategy has the lowest performance. Results for *MostGeneral* and *MostFrequent* are between the regression based selector and *MemoryBased*.

8.2 Incremental Learning Results

Figure 5 presents the incremental learning results on the testing set. To better present the results, we show the performance based on each learning cycle of 40 instances. The averaged Jaccard Index (SJI) is reported. Specifically, Figure 5(a) shows the results of configurations comparing different base hypothesis induction heuristics using regression model based hypothesis selection. After using 200 out of 840 (23.8%) learning instances, all the four curves achieve more than 80% of the overall performance. For example, for the heuristic $H4_{all}$, the final average Jaccard Index is 0.418. When 200 instances are used, the score is 0.340 ($0.340/0.418 \approx 81\%$). The same number holds for the other heuristics. After 200 instances, $H4_{all}$ and $H2_{manip}$ consistently achieve better performance than $H1_{argonly}$ and $H3_{argrelated}$. This result indicates that while change of states mostly affect the arguments of the verbs, other state changes in the environment cannot be ignored. Modeling them actually leads to better performance. Using $H4_{all}$ for base hypothesis induction, Figure 5(b) shows the results of comparing different hypothesis selection strategies. The regression model based selector always outperforms other selection strategies.

8.3 Results on Frequently Used Verb Frames

Beside overall evaluation, we have also taken a closer look at individual verb frames. Most of the verb frames in the data have a very low frequency, which cannot produce statistically significant results. So we only selected verb frames with frequency larger than 40 in this evaluation. For each verb frame, 60% data are used for incremental learning and 40% are for testing. For each frame, a regression based selector is trained separately. The resulting SJI curves are shown in Figure 6.

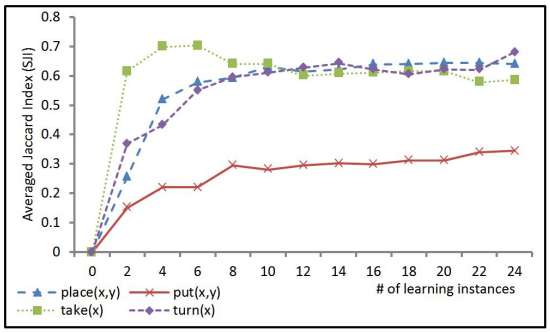


Figure 6: Incremental evaluation for individual verb frames. Four frequently used verb frames are examined: $place(x, y)$, $put(x, y)$, $take(x)$, and $turn(x)$. X-axis is the number of incremental learning instances, and Y-axis is the averaged SJI computed with $H4_{all}$ base hypothesis induction and regression based hypothesis selector.

As shown in Figure 6, all the four curves become steady after 8 learning instances are used. However, while some verb frames have final SJIs of more than 0.55 (i.e. $take(x)$ and $turn(x)$), others have relatively lower results (e.g. results for $put(x, y)$ are lower than 0.4). After examining the learning instances for $put(x, y)$, we found these data are more noisy than the training data for other frames. One source of errors is the incorrect object grounding results. For example, a problematic

training instance is "*put the pillow on the couch*", where the object grounding module cannot correctly ground the "*couch*" to the target object. As a result, the changed states of the second argument (i.e. the "*couch*") are incorrectly identified, which leads to incorrect prediction of desired states during inference. Another common error source is from automated parsing of utterances. The action frames generated from the parsing results could be incorrect in the first place, which would contribute to a hypothesis space for a wrong frame. These different types of errors are difficult to be recognized by the system itself. This points to the future direction of involving humans in a dialogue to learn a more reliable hypothesis space for verb semantics.

9 Conclusion

This paper presents an incremental learning approach that represents and acquires semantics of action verbs based on state changes of the environment. Specifically, we propose a hierarchical hypothesis space, where each node in the space describes a possible effect on the world from the verb. Given a language command, the induced hypothesis space, together with a learned hypothesis selector, can be applied by the agent to plan for lower-level actions. Our empirical results have demonstrated a significant improvement in performance compared to a previous leading approach. More importantly, as our approach is based on incremental learning, it can be potentially integrated in a dialogue system to support life-long learning from humans. Our future work will extend the current approach with dialogue modeling to learn more reliable hypothesis spaces of resulting states for verb semantics.

Acknowledgments

This work was supported by IIS-1208390 and IIS-1617682 from the National Science Foundation. The authors would like to thank Dipendra K. Misra and colleagues for providing the evaluation data, and the anonymous reviewers for valuable comments.

References

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, Volume1(1):49–62.

R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz. 2012. Tell me when and why to do it! run-time planner model updates via natural language instruction. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI'12)*, pages 471–478, Boston, Massachusetts, USA, March.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-2011)*, pages 859–865, San Francisco, California, USA, August.

Richard E. Fikes and Nils J. Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2nd International Joint Conference on Artificial Intelligence (IJCAI'71)*, pages 608–620, London, England.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Y. Chai. 2016. Physical causality of action verbs in grounded language understanding. In *In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.

Yolanda Gil. 1994. Learning by experimentation: incremental refinement of incomplete planning domains. In *Prococeedings of the Eleventh International Conference on Machine Learning (ICML'94)*, pages 87–95, New Brunswick, NJ, USA.

P. Gorniak and D. Roy. 2007. Situated language understanding as filtering perceived affordances. *Cognitive Science*, Volume31(2):197–231.

Malka Rappaport Hovav and Beth Levin. 2008. Reflections on manner/result complementarity. *Lecture notes*.

Malka Rappaport Hovav and Beth Levin. 2010. Reflections on Manner / Result Complementarity. *Lexical Semantics, Syntax, and Event Structure*, pages 21–38.

Yunyi Jia, Ning Xi, Joyce Y. Chai, Yu Cheng, Rui Fang, and Lanbo She. 2014. Perceptive feedback for natural language control of robotic operations. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, pages 6673–6678.

Joohyun Kim and Raymond J. Mooney. 2012. Unsupervised pcfg induction for grounded language learning with highly ambiguous supervision. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL '12)*, pages 433–444, Jeju Island, Korea.

Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. 2007. From structured english to robot motion. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 2717–2722.

Changsong Liu and Joyce Y. Chai. 2015. Learning to mediate perceptual differences in situated human-robot dialogue. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI'15)*, pages 2288–2294, Austin, Texas, USA.

Changsong Liu, Lanbo She, Rui Fang, and Joyce Y. Chai. 2014. Probabilistic labeling for efficient referential grounding based on collaborative discourse. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics ACL'14 (Volume 2: Short Papers)*, pages 13–18, Baltimore, MD, USA.

Dipendra Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2014. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *Proceedings of Robotics: Science and Systems (RSS'14)*, Berkeley, US.

Dipendra Kumar Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing ACL-IJCNLP'15 (Volume 1: Long Papers)*, pages 992–1002, Beijing, China.

Shiwali Mohan, James Kirk, and John Laird. 2013. A computational model for situated task learning with interactive instruction. In *Proceedings of the International conference on cognitive modeling (ICCM'13)*.

Kira Mouro, Luke S. Zettlemoyer, Ronald P. A. Petrick, and Mark Steedman. 2012. Learning strips operators from noisy and incomplete observations. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI'12)*, pages 614–623, Catalina Island, CA, USA.

Hanna M Pasula, Luke S Zettlemoyer, and Leslie Pack Kaelbling. 2007. Learning symbolic models of stochastic domains. *Journal of Artificial Intelligence Research*, Volume29:309–352.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, Volume12:2825–2830.

Jussi Rintanen. 2012. Planning as satisfiability: Heuristics. *Artificial Intelligence*, Volume193:45–86.

Lanbo She, Yu Cheng, Joyce Y. Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. 2014a. Teaching robots new actions through natural language instructions. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication, IEEE RO-MAN'14*, pages 868–873, Edinburgh, UK.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Y. Chai, and Ning Xi. 2014b. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 89–97, Philadelphia, PA, U.S.A., June. Association for Computational Linguistics.

M. Spangenberg and D. Henrich. 2015. Grounding of actions based on verbalized physical effects and manipulation primitives. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 844–851, Hamburg, Germany.

Jaeyong Sung, Bart Selman, and Ashutosh Saxena. 2014. Synthesizing manipulation sequences for under-specified tasks using unrolled markov random fields. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'14)*, pages 2970–2977, Chicago, IL, USA.

Stefanie Tellex, Pratiksha Thaker, Joshua Joseph, and Nicholas Roy. 2014. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, Volume94(2):151–167.

Jesse Thomason, Shiqi Zhang, Raymond Mooney, and Peter Stone. 2015. Learning to interpret natural language commands through human-robot dialog. In *Proceedings of the 2015 International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1923–1929, Buenos Aires, Argentina.

Xuemei Wang. 1995. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML'95)*, pages 549–557, Tahoe City, California, USA.

Qiang Yang, Kangheng Wu, and Yunfei Jiang. 2007. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, Volume171(23):107 – 143.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y. Chai. 2016. Grounded semantic role labeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'16)*, San Diego, California.

Language Transfer Learning for Supervised Lexical Substitution

Gerold Hintz and **Chris Biemann**
Research Training Group AIPHES / FG Language Technology
Computer Science Department, Technische Universität Darmstadt
{hintz,biem}@lt.informatik.tu-darmstadt.de

Abstract

We propose a framework for lexical substitution that is able to perform transfer learning across languages. Datasets for this task are available in at least three languages (English, Italian, and German). Previous work has addressed each of these tasks in isolation. In contrast, we regard the union of three shared tasks as a combined multilingual dataset. We show that a supervised system can be trained effectively, even if training and evaluation data are from different languages. Successful transfer learning between languages suggests that the learned model is in fact independent of the underlying language. We combine state-of-the-art unsupervised features obtained from syntactic word embeddings and distributional thesauri in a supervised delexicalized ranking system. Our system improves over state of the art in the full lexical substitution task in all three languages.

1 Introduction

The lexical substitution task is defined as replacing a target word in a sentence context with a synonym, which does not alter the meaning of the utterance. Although this appears easy to humans, automatically performing such a substitution is challenging, as it implicitly addresses the problem of both determining semantically similar substitutes, as well as resolving the ambiguity of polysemous words. In fact, lexical substitution was originally conceived as an extrinsic evaluation of Word Sense Disambiguation (WSD) when first proposed by McCarthy & Navigli (2007). However, a system capable of replacing words by appropriate meaning-preserving substitutes can be utilized in downstream tasks that require paraphrasing of input text. Examples of such use cases include text simplification, text shortening, and summarization. Furthermore, lexical substitution can be regarded as an alternative to WSD in downstream tasks requiring word disambiguation. For example, it was successfully applied in Semantic Textual Similarity (Bär et al., 2012). A given list of substitution words can be regarded as a vector representation modeling the meaning of a word in context. As opposed to WSD systems, this is not reliant on a predefined sense inventory, and therefore does not have to deal with issues of coverage, or sense granularity. On the other hand, performing lexical substitution is more complex than WSD, as a system has to both generate and rank a list of substitution candidates per instance.

Over the last decade, a number of shared tasks in lexical substitution has been organized and a wide range of methods have been proposed. Although many approaches are in fact language-independent, most existing work is tailored to a single language and dataset. In this work, we investigate lexical substitution as a multilingual task, and report experimental results for English, German and Italian datasets. We consider a supervised approach to lexical substitution, which casts the task as a ranking problem (Szarvas et al., 2013b). We adapt state-of-the-art unsupervised features (Biemann and Riedl, 2013; Melamud et al., 2015a) in a delexicalized ranking framework and perform transfer learning experiments by training a ranker model from a different language. Finally, we demonstrate the utility of aggregating data from different languages and train our model on this single multilingual dataset. We are able to improve the state of the art for the full task on all datasets.

The remainder of this paper is structured as follows. In Section 2 we elaborate on the lexical sub-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 118–129,
Berlin, Germany, August 7-12, 2016.

stitution task and datasets. Section 3 shows related work of systems addressing each of these tasks. In Section 4 we describe our method for building a supervised system capable of transfer learning. Section 5 shows our experimental results and discussion. Finally in Section 6 we give a conclusion and outlook to future work.

2 Lexical substitution datasets and evaluation

The lexical substitution task was first defined at *SemEval 2007* (McCarthy and Navigli, 2007, "SE07"). A lexical sample of target word is selected from different word classes (nouns, verbs, and adjectives). Through annotation, a set of valid substitutes was collected for 10-20 contexts per target. Whereas in the original SE07 task, annotators were free to provide "up to three, but all equally good" substitutes, later tasks dropped this restriction. Substitutes were subsequently aggregated by annotator frequency, creating a ranking of substitutes. The use of SE07 has become a de-facto standard for system comparison, however equivalent datasets have been produced for other languages. *Evalita 2009* posed a lexical substitution task for Italian (Toral, 2009, "EL09"). Participants were free to obtain a list of substitution candidates in any way, most commonly *Italian WordNet*[1] was used. A *WeightedSense* baseline provided by the organizers proved very strong, as all systems scored below it. This baseline is obtained by aggregating differently weighted semantic relations from multiple human-created lexical resources (Ruimy et al., 2002). A German version of the lexical substitution task was organized at *GermEval 2015* (Cholakov et al., 2014; Miller et al., 2015, "GE15"). Likewise, *WeightedSense* was able to beat both of two participating systems in *oot* evaluations (Miller et al., 2015).

A variation for *cross-lingual lexical substitution* was proposed by Mihalcea et al. (2010), in which substitute words are required in a different language than the source sentence. The sentence context as well as the target word were given in English, whereas the substitute words should be provided in Spanish (annotators were fluent in both languages). This variant is motivated by direct application in Machine Translation systems, or as an aid for human-based translation. There also exists a larger crowd-sourced dataset of 1012 nouns (Biemann, 2013, "TWSI"), as well as an all-words dataset in which all words in each sentence are annotated with lexical expansions (Kremer et al., 2014). Evaluation of lexical substitution adheres to metrics defined by SE07 (McCarthy and Navigli, 2007), who provide two evaluation settings[2]; *best* evaluating only a system's "best guess" of a single target substitute and *oot,* an unordered evaluation of up to ten substitutes. Thater et. al (2009) proposed to use *Generalized Average Precision* (GAP), to compare an output *ranking* rather than unordered sets of substitutes.

Dataset comparison The proposed lexical substitution datasets (SE07, EL09, GE15) differ in their degree of ambiguity of target items. If a dataset contains mostly target words that are unambiguous, substitution lists of different instances of the same target are similar, despite occurring in different context. We can quantify this degree of variation by measuring the overlap of gold substitutes of each target across all contexts. For this, we adapt the *pairwise agreement* (PA) metric defined by McCarthy & Navigli (2009). Instead of inter-annotator agreement we measure agreement across different context instances. Let T be a set of lexical target words, and D dataset of instances $(t_i, S_i) \in D$, in which target $t_i \in T$ is annotated with a set of substitutes S_i. Then we regard for each target word t the substitute sets $S_t \subset D$ for t. We define a *substitute agreement* as $SA(t)$ as the mean pairwise dice coefficient between all $s_1, s_2 \in S_t$ where $s_1 \neq s_2$. For each dataset D we list the substitute variance $SV = 1 - \frac{1}{|T|} \sum_{t \in T} SA(t)$. Table 1 shows this metric for the three datasets, as well as for subsets of the dataset according to target part of speech. It can be seen that the variance in gold substitutes differs substantially between datasets, but not much between target word type within a dataset. *EL09* has the highest degree of variance, suggesting that targets tend to be more ambiguous, whereas *GE15* has the lowest degree of variance, suggesting less ambiguity.

3 Related Work

Lexical substitution has been addressed extensively in recent years. Early systems, having only very few training instances available, use un-

[1] Italian WordNet has later been migrated into MultiWordNet (MWN), which is used in this work.

[2] The original SE07 task had a third evaluation setting MWE, in which systems had to correctly identify which target words were part of a multiword expression.

dataset	substitute variance (*SV*)				
	noun	verb	adj	adv	all
SemEval-2007	0.78	0.79	0.72	0.66	0.75
Evalita-2009	0.84	0.82	0.83	0.82	0.83
GermEval-2015	0.59	0.67	0.60	-	0.66
all	0.75	0.72	0.73	0.69	0.73

Table 1: Degree of variation in gold answers

supervised approaches for determining appropriate substitutes. For the English SE07 task, systems mostly consider substitution candidates from *WordNet* (Fellbaum, 1998) and cast lexical substitution into a ranking task. Experiments may also be performed by pooling the set of candidates from the gold data, evaluating a pure ranking variant. Early approaches use a contextualized word instance representation and rank candidates according to their similarity to this representation. Effective representations are *syntactic vector space models* (Erk and Padó, 2008; Thater et al., 2011), which use distributional sparse vector representations based on the syntactic context of words. Performance improvement could be shown for different models, including the use of graph centrality algorithms on directional word similarity graphs (Sinha and Mihalcea, 2011), and clustering approaches on word instance representations (Erk and Padó, 2010). Multiple systems have built upon the distributional approach. Extensions include the use of LDA topic models (Ó Séaghdha and Korhonen, 2014), and probabilistic graphical models (Moon and Erk, 2013). The current state of the art combines a distributional model with the use of n-gram language models (Melamud et al., 2015a). They define the context vector of each word in a background corpus as a *substitute vector*, which is a vector of suitable filler words for the current n-gram context. They then obtain a contextualized *paraphrase vector* by computing a weighted average of substitute vectors in the background corpus, based on their similarity to the current target instance. In contrast to traditional sparse vector representations obtained through distributional methods, a recent trend is the use of low-dimensional dense vector representations. The use of such vector representations or word embeddings has been popularized by the *continuous bag-of-words* (CBOW) and *Skip-gram* model (Mikolov et al., 2013a). Melamud et al. (2015b) show a simple and knowledge-lean model for lexical substitution based solely on syntactic word embeddings. As we leverage this model as a feature in our approach, we will elaborate on this in Section 4. Another approach for applying word embeddings to lexical substitution is their direct extension with multiple word senses, which can be weighted according to target context (Neelakantan et al., 2014).

Biemann (2013) first showed that the lexical substitution task can be solved very well when sufficient amount of training data is collected per target. An approach based on crowdsourcing human judgments achieved the best performance on the S07 dataset to day. However, judgments had to be collected for each lexical item, and as a consequence the approach can not scale to an open vocabulary. As an alternative to *per-word* supervised systems trained on target instances per lexeme, *all-words* systems aim to generalize over all lexical items. Szarvas et al. (2013a) proposed such a system by using *delexicalization*: features are generalized in such a way that they are independent of lexical items, and thus generalize beyond the training set and across targets. Originally, a maximum entropy classifier was trained on target-substitute instances and used for pointwise ranking of substitution candidates. In a follow-up work it was shown that learning-to-rank methods could drastically improve this approach, achieving state-of-the-art performance with a *LambdaMART* ranker (Szarvas et al., 2013b). In this work we will build upon this model and further generalize not only across lexical items but across different languages.

For both EL09 and GE15, existing approaches have been adapted. For the Italian dataset, a distributional method was combined with LSA (De Cao and Basili, 2009). The best performing system applied a WSD system and language models (Basile and Semeraro, 2009). For the German dataset, Hintz and Biemann (2015) adapted the supervised approach by (Szarvas et al., 2013a), achieving best performance for nouns and adjectives. Jackov (2015) used a deep semantic analysis framework employing an internal dependency relation knowledge base, which achieved the best performance for verbs.

4 Method description

We subdivide lexical substitution into two subtasks; *candidate selection* and *ranking*. For a given target t, we consider a list of possible substi-

tutes $s \in C_t$, where C_t is a static per-target candidate list. Our method is agnostic to the creation of this static resource, which can be obtained either by an unsupervised similarity-based approach, or from a lexical resource. In particular, candidates obtained at this stage do not disambiguate possible multiple senses of t, and are filtered and ordered in the ranking stage by a supervised model.

In modeling a supervised system, we have experimented with two learning setups. The first is applying a standard classification / regression learner. Here, lexical substitution is cast into a pointwise ranking task by training on target-substitute pairs generated from the gold standard. For each sentence context c, target word t and substitute s, we regard the tuple (c,t,s) as a training instance. We obtain these training instances for each lexsub instance (c,t) by considering all substitutes $s \in G_t \cup C_t$ where G_t are all candidates for target t pooled from the gold data and C_t are obtained from lexical resources. We then experiment with two labeling alternatives for a binary classification and a regression setup, respectively. For binary classification we label each instance (c,t,s) as *positive* if s has been suggested as a substitute for t by at least one annotator, and as *negative* otherwise. For regression, we normalize the annotation counts for each substitute to obtain a score in $(0,1]$ if a substitute s occurs in the gold data, 0 otherwise. The ranking of substitutes per target is obtained by considering the posterior likelihood of the *positive* label as yielded by a classifier model. We have tried multiple classifiers but have found no significant improvement over a *maximum entropy* baseline[3]. Our second setup is a learning-to-rank framework, adapted from (Szarvas et al., 2013b). Here, we are not restricted to a pointwise ranking model, but consider pairwise and listwise models[4].

We base our feature model on existing research. In addition to basic syntactic and frequency-based features, we obtained sophisticated features from trigram and syntactic thesauri, motivated by the findings of Biemann and Riedl (2013), as well as syntactic embedding features motivated by Melamud et al. (2015b).

[3]For classification setup we use *Mallet*: `http://mallet.cs.umass.edu/`

[4]For learning-to-rank we use *RankLib*: `http://mallet.cs.umass.edu/`

dataset	maximum recall	
	w/ MWE	w/o MWE
SemEval-2007	0.459	0.404
Evalita-2009	0.369	0.337
GermEval-2015	0.192	0.178
all	0.242	0.223

Table 2: Upper bound for substitute recall based on lexical resources *WordNet*, *MultiWordNet*, *GermaNet*

4.1 Candidate selection

We confirm earlier research (Sinha and Mihalcea, 2009) on the high quality of selecting candidates from lexical resources. We thus base our candidate selection on prevalently used resources: *WordNet* (Fellbaum, 1998) for English, *GermaNet* (Hamp and Feldweg, 1997) for German and *MultiWordNet* (Pianta et al., 2002) for Italian. For all resources, we consider all possible senses for a given target word and obtain all *synonyms*, *hypernyms* and *hyponyms* and their transitive hull. Thus, for the *hypernymy* and *hyponymy* relation, we follow the respective edges in the graph collecting all nodes (synsets) along the path. For each synset, we extract all lemmas as substitution candidates. Although restricting candidates incurs a relatively low upper bound on system recall, we still obtain best results using this rather conservative filter. Table 2 shows the upper bound for system recall for each of the datasets, evaluated with and without removing all multiword expressions from both candidate lists and gold data. A higher coverage of *WordNet* is a plausible explanation for the much higher recall on the English data.

4.2 Supervised ranking

Learning-to-rank methods train a supervised model for ranking a list of items by *relevance*. A basic *pointwise* approach applies regression techniques to obtain a relevance scores for each item in isolation. More advanced models are based on *pairwise* preference information for instance pairs, and *listwise* approaches, which are optimized on a global metric of a given ranking output. An extensive overview of learning-to-rank models can be found in (Burges, 2010). For lexical substitution, *LambdaMART* (Wu et al., 2010) has been found to be particularly effective. *LambdaMART* is a listwise method based on gradient boosting of regression trees. Its two main hyperparameters are

the number of leaves in each regression tree and the number of iterations and trees. We have not performed extensive tuning of these hyperparameters and used default settings, an ensemble of 1000 trees with 10 leaves.

4.3 Delexicalized features

The idea of delexicalization has been proposed, for instance, by Bergsma et al. (2007). They propose to use statistical measures based solely on the frequency of different expansions of the same target term. Their feature set has motivated a large subset of the feature model, which we adapt in this work. The idea of generalizing features for lexical substitution in such a way that they work across lexical items has been shown by Moon and Erk (2013), and made explicit by Szarvas et al. (2013a). Instances are characterized using non-lexical features from heterogeneous evidence. The intuition of this feature model is to exploit redundant signals of substitutability from different sources and methods.

In cases where background corpora are required, the following data is used throughout all features: For English, a newspaper corpus compiled from 105 million sentences from the Leipzig Corpora Collection (Richter et al., 2006) and the Gigaword corpus (Parker et al., 2011) was used. For German a 70M sentence newswire corpus (Biemann et al., 2007) was used. For Italian, a subset of 40M sentences of *itWac*, a large web-crawl, was used (Baroni et al., 2009).

Shallow syntactic features We apply a part-of-speech tagger trained on universal POS tags (Petrov et al., 2012), which we simplify into the classes *noun*, *verb*, *adjective* and *adverb*. Using these simplified tags we construct an n-gram sliding window, with $n \in [1..5]$, of POS around the target. We could also reduce window sizes drastically to $n = 1{,}2$ without sacrificing performance.

Frequency features We use language models for each of the languages to obtain *frequency ratio* features. An n-gram sliding window around a target t is used to generate a set of features $\frac{freq(c_l,s,c_r)}{freq(c_l,t,c_r)}$, where c_l and c_r are the left and right context words around t. Here, we normalize the frequency of the substitute with the frequency of the n-gram with original target t. As a variant, we further normalize frequencies by the set of all substitutes, to obtain frequencies features $\frac{freq(c_l,s,c_r)}{\sum_{s' \in C_t} freq(c_l,s',c_r)}$ where C_t is the set of candidate substitutes for t. In our experiments we used sliding windows of size $[1..5]$. We obtain 5-gram counts from *web1t* (Brants and Franz, 2009).

Conjunction ratio features Based on the n-gram resources above, we further define a conjunctive phrase ratio feature, which measures how often the construct $(c_l, t, conjunction, s, c_r)$ occurs in a background corpus; i.e. how often t and s co-occur with a *conjunction* word ("*and*", "*or*", ","), within the context of the sentence. As there is a different set of conjunction words for each language, we first aggregate the mean over all conjunction words:

$$conj_{l,r}(t,s) = \frac{1}{|CONJ|} \sum_{con \in CONJ} freq(c_l,t,con,s,c_r)$$

where l and r is the size of the left and right context window, and $CONJ$ is a set of conjunction words per-language[5]. For left and right context size $l = r = 0$ this feature also captures a context-independent conjunction co-occurrence between only t and s. Again, we normalize this feature over the set of all candidates:

$$\frac{conj_{l,r}(t,s)}{\sum_{s' \in C_t} conj_{l,r}(t,s)}$$

Distributional features We construct a *distributional thesaurus* (DT) for each of the languages by following Biemann and Riedl (2013) and obtain first-order word-to-context measures, as well as second-order word-to-word similarity measures. As context features we have experimented with both syntactic dependencies as well as left and right neighboring words, and have found them to perform equivalently. As a salience measure we use *Lexicographer's Mutual Information* (Bordag, 2008) and prune the data, keeping only the 1000 most salient features per word. Word similarity is obtained from an overlap count in the pruned context features. We model features for the *contextualized* distributional similarity between t and s as

- percentage of shared context features for the top-k context features of t and s, globally and restricted to sentence context (k =5, 20, 50, 100, 200)

[5] Conjunction words used are *and*, *or*, (*comma*), for English; *und*, *oder*, (*comma*) for German and *e*, *ed*, *o*, *od*, (*comma*) for Italian.

- percentage of shared words for the top-k similar words of t and s (k =200)
- sum of salience score of context features of s overlapping with the sentence context
- binary occurrence of s in top-k similar words of t (k =100, 200)

With the exception of the last feature, these measures are scaled to $[0,1]$ over the set of all substitute candidates.

Syntactic word embeddings We adapt the unsupervised approach by (Melamud et al., 2015a) as a set of features. We follow (Levy and Goldberg, 2014) to construct dependency-based word embeddings; we obtain syntactic contexts by running a syntactic dependency parser[6], and computing word embeddings using dependency edges as context features[7]. The resulting dense vector representations for words and context live within the same vector space. We compute the semantic similarity between a target and a substitute word from the cosine similarity in the word embedding space, as well as the first-order target-to-context similarity. For a given target word t and substitute s, let C_t be the syntactic context of t and $c \in C_t$ a single context – i.e. a dependency edge attached to t; let v_t, v_s be the vector representations of t and s in the word embedding space, and v_c the vector representation of c in the context embedding space. Then $Sim_1 = \cos(v_s, v_c)$ and $Sim_2 = \cos(v_s, v_t)$ are the first-order and second-order substitutability measures considered by Melamud et al. (2015a). In contrast to their approach, we do not just consider an unsupervised combination of these two measures, but instead use both Sim_1 and Sim_2 as single features. We also use their combinations of a *balanced* / *unbalanced*, *arithmetic* / *geometrical* mean, to obtain six numeric features in total. Importantly, these features are independent of the underlying embedding vectors and can therefore generalize across arbitrary embeddings between languages.

Semantic resource features To generalize across multiple languages we minimize the complexity of features obtained from semantic resources – which may differ notably in size and structure. From the resources listed in Section 4.1 we extract binary features for the semantic relations *synonymy*, *hypernymy* and *hyponymy*, occurring between t and s. We have also experimented with graded variants for transitive relations, such as encoding n-th level hypernymy, but have not observed any gain from this feature variation.

[6] We trained models for *Mate* (`https://code.google.com/p/mate-tools/`) based on *universal dependencies* (`http://universaldependencies.org/`)

[7] We used *word2vecf* (`https://bitbucket.org/yoavgo/word2vecf`) for computing syntactic word embeddings

4.4 Transfer learning

Transfer learning is made feasible by a fully lexeme-independent and language-independent feature space. Language-specific knowledge resides only within the respective resources for each language, and gets abstracted in feature extraction. Figure 1 illustrates this process at the example of two entirely unrelated sentences in different languages (English and German). A further mediator for transfer learning is a model based on boosted decision trees. As opposed to linear models, which could not be reasonably learned across languages, a *LambdaMART* ranker is able to learn feature interaction across languages. To give an example of what the resulting model can pick up on, we can regard conditionally strong features. Consider the n-gram pair frequency ratio feature of window size $(l,r) = (1,0)$, which compares the frequency ratio of the target and substitute including a single left context word. Depending on the POS window, this feature can be highly informative in some cases, where it is less informative in others. For *adjective-noun* pairs, in which the noun is the substitution target, the model can learn that this frequency ratio is strongly positively correlated; in this case, the substitute frequently occurs with the same adjective than the original target. For other POS windows, for example *determiner-noun* pairs, the same frequency ratio may be less indicative, as most nouns frequently occur with a determiner. This property works across languages, as long as as attributive adjectives are prepositioned. In our subset of languages, this is the case for English and German, but not for Italian, which uses postpositive adjectives. Nevertheless, we are able to learn such universal feature interactions.

5 Results and discussion

Evaluation of lexical substitution requires special care, as different evaluation settings are used

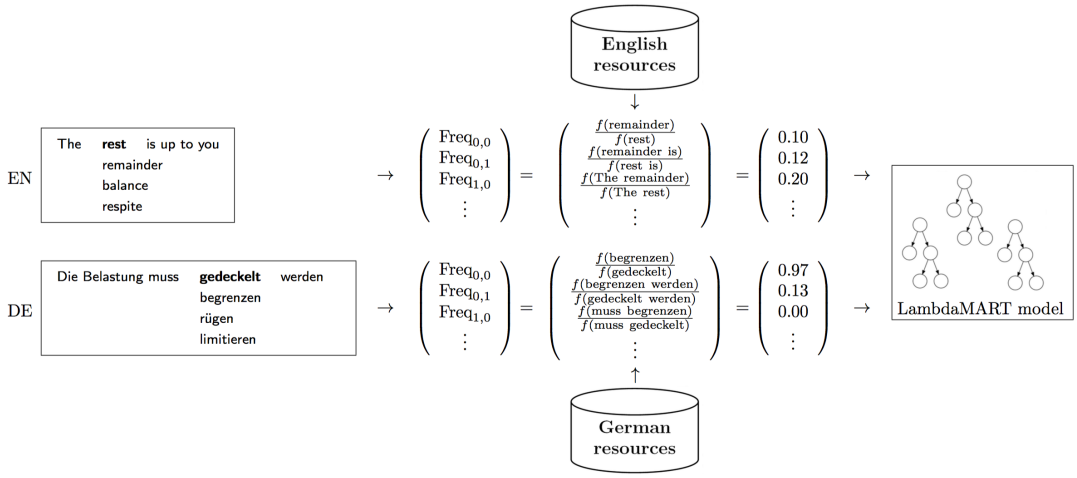


Figure 1: Visualization of feature extraction and delexicalization. Two unrelated sentences in English and German (translation: "*the strain has to be **limited***") are shown. Language-specific knowledge is obtained from resources for each language respectively. The resulting feature space is delexicalized and language independent.

throughout previous work and comparability is not always guaranteed. We follow the convention of reporting the full lexical substitution task (both generating and ranking candidates) with the metrics *P-best* and *P-oot* and report the ranking-only task (candidates pooled from the gold standard) with the *GAP* score. We further observe that previous work commonly discards multiword expressions from both the candidate lists as well as the gold data[8]. We follow this convention, but note that our system is in fact capable of successfully ranking multiword expansions out of the box. System performance slightly decreases when including MWE, as there is virtually no overlap between those provided by the system and those in the gold standard.

For ranking we experiment with different pointwise classifiers as provided by *Mallet* (MaxEnt classification and regression) as well learning-to-rank models provided by *RankLib* (RankBoost, RankNet, LambdaMART). In line with findings in (Szarvas et al., 2013b), we observe that learning-to-rank approaches work better than a pointwise classification / regression setup throughout all languages and feature subsets. Among different rankers, we confirm LambdaMART to yield the best performance, and will only report numbers using this model. As optimization metric we have explored both *NDCG@10* and *MAP*. The *NDCG* metric can incorporate different scoring weights based on annotator overlap, however MAP directly correlates with evaluation score. We have found optimizing on MAP to yield slightly better results, even if this disregards the relative score weights between gold substitutes. For the ranking-only task, we also extended the pooled training data with additional negative examples (i.e. adding all candidates as for the full task) but observed a minor decrease in system performance.

[8]The omission of MWE by multiple authors has been confirmed by the authors of (Melamud et al., 2015a).

	Open evaluation (best-P / oot-P)					
Training	English		German		Italian	
English	16.63	48.16	7.43	26.79	8.57	31.94
German	13.20	44.61	11.97	38.45	7.05	28.75
Italian	13.91	39.72	4.25	22.66	15.19	40.37
others	17.19	46.79	8.15	27.33	10.04	30.82
all	**17.23**	**48.83**	**12.94**	**41.32**	**16.15**	**41.29**
SOA[9]	15.94	36.37	11.20	20.14	10.86	41.46

Table 3: Transfer learning results for the open candidate task (candidates from lexical resources)

	Ranking evaluation (GAP)		
Training	English	German	Italian
English	51.0	26.9	44.5
German	44.3	**56.2**	42.9
Italian	36.7	22.2	48.0
others	43.7	26.7	43.9
all	**51.9**	51.3	**50.0**

Table 4: Transfer learning results on the ranking-only task (candidates pooled from gold)

We report transfer learning results across all three datasets. Table 3 shows a transfer-learning matrix for the full lexical substitution task, whereas Table 4 shows results for the ranking-only task. For evaluation, we consistently use the complete datasets, which are roughly equal in size for all languages (~ 2000 instances). For the identity entries in this matrix, as well as training on the complete dataset ("all") we follow previous supervised work and perform 10-fold cross-validation. Splits are based on the target lexeme, so that no two instances for the same target word are in different sets. Tables 3 and 4 suggest the feasibility of transfer learning. Although models trained on the original language (identity entries of the matrix) perform best, training on a different language still yields reasonable results. Training only on a single other language, not surprisingly, yields worse results for each dataset, however combining the data from the two remaining languages ("others") can mitigate this issue to some degree. Importantly, adding the data from two additional languages consistently improves system performance throughout all datasets for the open candidate task (Table 3). It is interesting to note that in case of SE07, training on only other languages performs surprisingly well for the *best-P* score, beating even a model trained on English. A possible explanation for this is that the SE07 dataset appears to be somewhere in the middle between EL09 and GE15 in terms of substitute variance. For the ranking-only task, transfer learning seems to work a little less effectively. In case of German, adding foreign language data in fact hurts *GAP* performance. This potentially originates from a much smaller set of training instances and inconsistency of the amount and overlap of pooled candidates across different tasks (as described in Table 1). We also observe that a learning-to-rank model is essential for performing transfer learning. In case of LambdaMART, an ensemble of decision trees is constructed, which is well suited to exploit redundant signals across multiple features. Linear models resulted in worse performance for transfer learning, as the resulting weights seem to be language-specific.

Feature ablation experiments are performed for various feature groups in the full and ranking-only task (Table 5). The ablation groups correspond to the feature categories defined in Section 4.3. The *frequency* group includes plain frequency features as well as conjunction ratio features. We consider only our universal model trained on all language data (with 10-fold CV for each dataset). In case of English, the full system performs best and all feature groups improve overall performance. For other languages these results are mixed. In case of the German data, embedding features and semantic relation features seem to work well on their own, so that results for other ablation groups are slightly better. For ranking-only, embedding features seem to be largely subsumed by the combination of the other groups. Ablation of embeddings differs vastly between the full and ranking-only task; they seem to more more crucial for the full task. For all languages, semantic relations are the best feature in the full task, acting as a strong filter for candidates; in ranking-only they are more dispensable.

In summary, we observe that delexicalized transfer learning for lexical substitution is possible. Existing supervised approaches can be extended to generalize across multiple languages without much effort. Training a supervised system on different language data emphasizes that the learned model is sufficiently generic to be language independent. Our feature space constructed from heterogeneous evidence consists of many features that perform relatively weakly on their own. The resulting ranking model captures redundancy between these signals. Finally, Table 6 shows our results in comparison to previous work. Note that we omit some participating systems from the original SE07 task. The reason we did not list IRST2 (Giuliano et al., 2007) is that for out-of-ten results, the system outputs the same substitute multiple times and the evaluation scheme gives credit for each copy of the substitute. Our (and other) systems do not tamper with the metric in this way, and only yield a set of substitutes. UNT (Hassan et al., 2007) uses a much richer set of knowledge bases, not all of them easily available, to achieve slightly better *oot* scores. From our experiments, we list both a model trained per language, as well as a universal model trained on all data. The latter beats nearly all other approaches on the full lexical substitution task, despite not being optimized for a single language. Although omission of MWEs is common practice for SE07, it is unclear if this was

[9] State of the art baseline, according to previous reported results, c.f. Table 6

	English		German		Italian	
	best-P	GAP	best-P	GAP	best-P	GAP
w/o syntax	15.35	49.5	12.33	42.1	15.70	50.3
w/o frequency	17.04	48.6	13.30	54.6	15.78	51.5
w/o DT	16.88	48.8	12.18	54.6	17.65	51.8
w/o sem. relation	11.51	49.9	6.82	33.9	8.06	49.7
w/o embedding	10.05	51.5	11.51	47.1	7.17	54.4
full system	17.23	51.9	12.94	51.3	16.15	50.0

Table 5: Feature ablation results for the full and ranking-only task (universal model trained on all data)

done for EL09 and GE15. However, re-inclusion of MWE does not drastically alter results[10]. In the ranking-only variant, we are not able to beat the learning-to-rank approach by Szarvas et. al (2013b), we note however that they have performed extensive hyperparameter optimization of their ranker, which we have omitted. We are also not able to achieve *GAP* scores reported by Melamud at al. (2015b). Although we used their exact embeddings, we could not reproduce their results[11].

6 Conclusion

We are the first to model lexical substitution as a language-independent task by considering not just a single-language dataset, but by merging data from distinct tasks in English, German and Italian. We have shown that a supervised, delexicalized approach can successfully learn a single model across languages – and thus perform transfer learning for lexical substitution. We observe that a listwise ranker model such as LambdaMART facilitates this transfer learning. We have further shown that incorporating more data helps training a more robust model and can consistently improve system performance by adding foreign language training data. We extended an existing supervised learning-to-rank approach for lexical substitution (Szarvas et al., 2013b) with state-of-the-art embedding features (Melamud et al., 2015b). In our experiments, a single model trained on all data performed best on each language. In all three datasets we were able to improve the current state of the art for the full lexical substitution task. The resulting model can be regarded as language-independent; given an unannotated background corpus for computing language-specific resources and a source of substitution candidates, the system can be used almost out of the box. For obtaining substitution candidates, we still rely on lexical resources such as *WordNet*, which have to be available for each language. As future work we aim to make our approach completely knowledge-free by eliminating this dependency. We can consider substitution candidates based on their distributional similarity. First experiments confirm that this already yields a much better coverage, i.e. upper bound on recall, while introducing more noise. The remaining key challenge is to better characterize possible substitutes from bad substitutes in ranked lists of distributionally similar words, which frequently contain antonyms and co-hyponyms. We will explore unsupervised acquisition of relational similarity (Mikolov et al., 2013b) for this task.

SemEval '07			
method	**best-P**	**oot-P**	**GAP**
(Erk and Padó, 2010)	-	-	38.6
(Thater et al., 2011)	-	-	51.7
(Szarvas et al., 2013a)	15.94	-	52.4
(Szarvas et al., 2013b)	-	-	55.0
(Melamud et al., 2015b)	08.09	27.65	52.9
(Melamud et al., 2015a)	12.72	36.37	**55.2**
our method (English only)	16.63	48.16	51.0
our method (all data)	**17.23**	**48.83**	51.9
Evalita '09			
method	**best-P**	**oot-P**	**GAP**
(Basile and Semeraro, 2009)	08.16	**41.46**	-
(Toral, 2009)[12]	10.86	27.52	-
our method (Italian only)	15.19	40.37	48.0
our method (all data)	**16.15**	31.18	**50.0**
GermEval '15			
method	**best-P**	**oot-P**	**GAP**
(Hintz and Biemann, 2015)	11.20	19.49	-
(Jackov, 2015)	06.73	20.14	-
our method (German only)	11.97	38.45	56.2
our method (all data)	**12.94**	**41.32**	51.3

Table 6: Experimental results of our method compared to related work for all three lexical substitution tasks

[10]For comparison, our scores including MWE for the "all data" model are as follows (*best-P*, *oot-P*, *GAP*). EL09: 15.12, 33.92, 45.8; GE15: 12.20, 41.15, 50.0

[11]Our evaluation of (Melamud et al., 2015b), *balAdd* yields a *GAP* score of 48.8, which is likely related to different evaluation settings.

[12]baseline by task organizer

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1 and the SEMSCH project, grant No. BI 1544.

References

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. UKP: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 435–440, Montreal, Canada.

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

Pierpaolo Basile and Giovanni Semeraro. 2009. UNIBA @ EVALITA 2009 lexical substitution task. In *Proceedings of EVALITA workshop, 11th Congress of Italian Association for Artificial Intelligence*, Reggio Emilia, Italy.

Shane Bergsma and Qin Iris Wang. 2007. Learning noun phrase query segmentation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, volume 7, pages 819–826, Prague, Czech Republic.

Chris Biemann and Martin Riedl. 2013. Text: Now in 2D! A framework for lexical expansion with contextual similarity. *Journal of Language Modelling*, 1(1):55–95.

Chris Biemann, Gerhard Heyer, Uwe Quasthoff, and Matthias Richter. 2007. The Leipzig corpora collection: monolingual corpora of standard size. In *Proceedings of Corpus Linguistics*, Birmingham, UK.

Chris Biemann. 2013. Creating a system for lexical substitutions from scratch using crowdsourcing. *Language Resources and Evaluation*, 47(1):97–122.

Stefan Bordag. 2008. A Comparison of Co-occurrence and Similarity Measures As Simulations of Context. In *Proceedings of the 9th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing 2008, pages 52–63, Haifa, Israel.

Thorsten Brants and Alex Franz. 2009. Web 1T 5-gram, 10 European languages version 1. *Linguistic Data Consortium*.

Christopher J.C. Burges. 2010. From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research.

Kostadin Cholakov, Chris Biemann, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Lexical substitution dataset for German. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*, pages 1406–1411, Reykjavik, Iceland.

Diego De Cao and Roberto Basili. 2009. Combining distributional and paradigmatic information in a lexical substitution task. In *Proceedings of EVALITA workshop, 11th Congress of Italian Association for Artificial Intelligence*, Reggio Emilia, Italy.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 897–906, Waikiki, HI, USA.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 conference short papers*, pages 92–97, Uppsala, Sweden.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Claudio Giuliano, Alfio Gliozzo, and Carlo Strapparava. 2007. FBK-irst: Lexical substitution task exploiting domain and syntagmatic coherence. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 145–148, Prague, Czech Republic.

Birgit Hamp and Helmut Feldweg. 1997. GermaNet - a lexical-semantic net for German. In *In Proceedings of ACL workshop Automatic Information Extraction and Building of Lexical Semantic Resources for NLP Applications*, pages 9–15, Madrid, Spain.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. UNT: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 410–413, Prague, Czech Republic.

Gerold Hintz and Chris Biemann. 2015. Delexicalized supervised German lexical substitution. In *Proceedings of GermEval 2015: LexSub*, pages 11–16, Essen, Germany.

Luchezar Jackov. 2015. Lexical substitution using deep syntactic and semantic analysis. In *Proceedings of GermEval 2015: LexSub*, pages 17–20, Essen, Germany.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 540–549, Gothenburg, Sweden.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, Baltimore, MD, USA.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 48–53, Prague, Czech Republic.

Diana McCarthy and Roberto Navigli. 2009. The English lexical substitution task. *Language resources and evaluation*, 43(2):139–159.

Oren Melamud, Ido Dagan, and Jacob Goldberger. 2015a. Modeling word meaning in context with substitute vectors. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 472–482, Denver, CO, USA.

Oren Melamud, Omer Levy, and Ido Dagan. 2015b. A simple word embedding model for lexical substitution. *VSM Workshop*. Denver, CO, USA.

Rada Mihalcea, Ravi Sinha, and Diana McCarthy. 2010. Semeval-2010 task 2: Cross-lingual lexical substitution. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 9–14, Uppsala, Sweden.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, Lake Tahoe, NV, USA.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, GA, USA.

Tristan Miller, Darina Benikova, and Sallam Abualhaija. 2015. GermEval 2015: LexSub – A shared task for German-language lexical substitution. In *Proceedings of GermEval 2015: LexSub*, pages 1–9, Essen, Germany.

Taesun Moon and Katrin Erk. 2013. An inference-based model of word meaning in context as a paraphrase distribution. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(3):42.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1059–1069, Doha, Qatar.

Diarmuid Ó Séaghdha and Anna Korhonen. 2014. Probabilistic distributional semantics with latent variable models. *Computational Linguistics*, 40(3):587–631.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. Technical report, Linguistic Data Consortium, Philadelphia, PA, USA.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of the Eight International Conference on Language Resources and Evaluation*, pages 2089–2096, Istanbul, Turkey.

Emanuele Pianta, Luisa Bentivogli, and Christian Girardi. 2002. MultiWordNet: developing an aligned multilingual database. In *Proceedings of the 1st International WordNet Conference*, pages 293–302, Mysore, India.

Matthias Richter, Uwe Quasthoff, Erla Hallsteinsdóttir, and Chris Biemann. 2006. Exploiting the Leipzig Corpora Collection. In *Proceedings of the Information Society Language Technologies Conference 2006*, pages 68–73. Ljubljana, Slovenia.

Nilda Ruimy, Monica Monachini, Raffaella Distante, Elisabetta Guazzini, Stefano Molino, Marisa Ulivieri, Nicoletta Calzolari, and Antonio Zampolli. 2002. CLIPS, a multi-level Italian computational lexicon: a glimpse to data. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation*, pages 792–799, Las Palmas, Spain.

Ravi Sinha and Rada Mihalcea. 2009. Combining lexical resources for contextual synonym expansion. In *Proceedings of the Conference in Recent Advances in Natural Language Processing*, pages 404–410, Borovets, Bulgaria.

Ravi Som Sinha and Rada Flavia Mihalcea. 2011. Using centrality algorithms on directed graphs for synonym expansion. In *FLAIRS Conference*, pages 311–316, Palm Beach, FL, USA.

György Szarvas, Chris Biemann, Iryna Gurevych, et al. 2013a. Supervised all-words lexical substitution using delexicalized features. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1131–1141, Atlanta, GA, USA.

György Szarvas, Róbert Busa-Fekete, and Eyke Hüllermeier. 2013b. Learning to rank lexical substitutions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1926–1932, Seattle, WA, USA.

Stefan Thater, Georgiana Dinu, and Manfred Pinkal. 2009. Ranking paraphrases in context. In *Proceedings of the 2009 Workshop on Applied Textual Inference*, pages 44–47, Singapore, Republic of Singapore.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2011. Word meaning in context: A simple and effective vector model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1134–1143, Edinburgh, UK.

Antonio Toral. 2009. The lexical substitution task at EVALITA 2009. In *Proceedings of EVALITA Workshop, 11th Congress of Italian Association for Artificial Intelligence*, Reggio Emilia, Italy.

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270.

Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning

Yulia Tsvetkov[♠] **Manaal Faruqui**[♠] **Wang Ling**[♣] **Brian MacWhinney**[♠] **Chris Dyer**[♣♠]
[♠]Carnegie Mellon University [♣]Google DeepMind
{ytsvetko,mfaruqui,cdyer}@cs.cmu.edu,
lingwang@google.com, macw@cmu.edu

Abstract

We use Bayesian optimization to learn curricula for word representation learning, optimizing performance on downstream tasks that depend on the learned representations as features. The curricula are modeled by a linear ranking function which is the scalar product of a learned weight vector and an engineered feature vector that characterizes the different aspects of the complexity of each instance in the training corpus. We show that learning the curriculum improves performance on a variety of downstream tasks over random orders and in comparison to the natural corpus order.

1 Introduction

It is well established that in language acquisition, there are robust patterns in the order by which phenomena are acquired. For example, prototypical concepts are acquired earlier; concrete words tend to be learned before abstract ones (Rosch, 1978). The acquisition of lexical knowledge in artificial systems proceeds differently. In general, models will improve during the course of parameter learning, but the time course of acquisition is not generally studied beyond generalization error as a function of training time or data size. We revisit this issue of choosing the order of learning—**curriculum learning**—framing it as an optimization problem so that a rich array of factors—including nuanced measures of difficulty, as well as prototypicality and diversity—can be exploited.

Prior research focusing on curriculum strategies in NLP is scarce, and has conventionally been following a paradigm of "starting small" (Elman, 1993), i.e., initializing the learner with "simple" examples first, and then gradually increasing data complexity (Bengio et al., 2009; Spitkovsky et al., 2010). In language modeling, this preference for increasing complexity has been realized by curricula that increase the entropy of training data by growing the size of the training vocabulary from frequent to less frequent words (Bengio et al., 2009). In unsupervised grammar induction, an effective curriculum comes from increasing length of training sentences as training progresses (Spitkovsky et al., 2010). These case studies have demonstrated that carefully designed curricula can lead to better results. However, they have relied on heuristics in selecting curricula or have followed the intuitions of human and animal learning (Kail, 1990; Skinner, 1938). Had different heuristics been chosen, the results would have been different. In this paper, we use curriculum learning to create improved word representations. However, rather than testing a small number of curricula, we search for an optimal curriculum using Bayesian optimization. A curriculum is defined to be the ordering of the training instances, in our case it is the ordering of paragraphs in which the representation learning model reads the corpus. We use a linear ranking function to conduct a systematic exploration of interacting factors that affect curricula of representation learning models. We then analyze our findings, and compare them to human intuitions and learning principles.

We treat curriculum learning as an outer loop in the process of learning and evaluation of vector-space representations of words; the iterative procedure is (1) predict a curriculum; (2) train word embeddings; (3) evaluate the embeddings on tasks that use word embeddings as the sole features. Through this model we analyze the impact of curriculum on word representation models and on extrinsic tasks. To quantify curriculum properties, we define three groups of features aimed at analyzing statistical and linguistic content and structure

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 130–139,
Berlin, Germany, August 7-12, 2016.

of training data: (1) diversity, (2) simplicity, and (3) prototypicality. A function of these features is computed to score each paragraph in the training data, and the curriculum is determined by sorting corpus paragraphs by the paragraph scores. We detail the model in §2. Word vectors are learned from the sorted corpus, and then evaluated on part-of-speech tagging, parsing, named entity recognition, and sentiment analysis (§3). Our experiments confirm that training data curriculum affects model performance, and that models with optimized curriculum consistently outperform baselines trained on shuffled corpora (§4). We analyze our findings in §5.

The contributions of this work are twofold. First, this is the first framework that formulates curriculum learning as an optimization problem, rather then shuffling data or relying on human intuitions. We experiment with optimizing the curriculum of word embeddings, but in principle the curriculum of other models can be optimized in a similar way. Second, to the best of our knowledge, this study is the first to analyze the impact of distributional and linguistic properties of training texts on the quality of task-specific word embeddings.

2 Curriculum Learning Model

We are considering the problem of maximizing a performance of an NLP task through sequentially optimizing the curriculum of training data of word vector representations that are used as features in the task.

Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be the training corpus with n lines (sentences or paragraphs). The curriculum of word representations is quantified by scoring each of the paragraphs according to the linear function $\mathbf{w}^\intercal \boldsymbol{\phi}(\mathcal{X})$, where $\boldsymbol{\phi}(\mathcal{X}) \in \mathbb{R}^{\ell \times 1}$ is a real-valued vector containing ℓ linguistic features extracted for each paragraph, and $\mathbf{w} \in \mathbb{R}^{\ell \times 1}$ denote the weights learned for these features. The feature values $\boldsymbol{\phi}(\mathcal{X})$ are z-normalized across all paragraphs. These scores are used to specify the order of the paragraphs in the corpus—the curriculum: we sort the paragraphs by their scores.

After the paragraphs are curriculum-ordered, the reordered training corpus is used to generate word representations. These word representations are then used as features in a subsequent NLP task. We define the objective function $eval : \mathcal{X} \rightarrow \mathbb{R}$, which is the quality estimation metric for this NLP task performed on a held-out dataset (e.g., correlation, accuracy, F_1 score, BLEU). Our goal is to define the features $\boldsymbol{\phi}(\mathcal{X})$ and to find the optimal weights $\mathbf{w}$ that maximize $eval$.

We optimize the feature weights using Bayesian optimization; we detail the model in §2.1. Distributional and linguistic features inspired by prior research in language acquisition and second language learning are described in §2.2. Figure 1 shows the computation flow diagram.

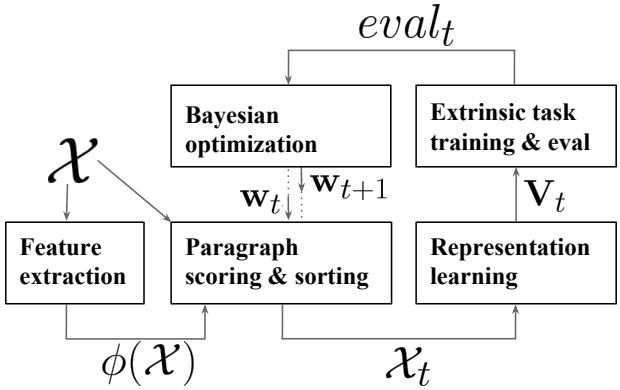


Figure 1: Curriculum optimization framework.

2.1 Bayesian Optimization for Curriculum Learning

As no assumptions are made regarding the form of $eval(\mathbf{w})$, gradient-based methods cannot be applied, and performing a grid search over parameterizations of $\mathbf{w}$ would require a exponentially growing number of parameterizations to be traversed. Thus, we propose to use Bayesian Optimization (BayesOpt) as the means to maximize $eval(\mathbf{w})$. BayesOpt is a methodology to globally optimize *expensive*, multimodal black-box functions (Shahriari et al., 2016; Bergstra et al., 2011; Snoek et al., 2012). It can be viewed as a sequential approach to performing a regression from high-level model parameters (e.g., learning rate, number of layers in a neural network, and in our model–curriculum weights $\mathbf{w}$) to the loss function or the performance measure ($eval$).

An arbitrary objective function, $eval$, is treated as a black-box, and BayesOpt uses Bayesian inference to characterize a posterior distribution over functions that approximate $eval$. This model of $eval$ is called the **surrogate model**. Then, the BayesOpt exploits this model to make decisions about $eval$, e.g., where is the expected maximum of the function, and what is the expected improvement that can be obtained over the best iteration so far. The strategy function, estimating the next set

of parameters to explore given the current beliefs about $eval$ is called the **acquisition function**. The surrogate model and the acquisition function are the two key components in the BayesOpt framework; their interaction is shown in Algorithm 1.

The surrogate model allows us to cheaply approximate the quality of a set of parameters $\mathbf{w}$ without running $eval(\mathbf{w})$, and the acquisition function uses this surrogate to choose a new value of $\mathbf{w}$. However, a trade-off must be made: should the acquisition function move $\mathbf{w}$ into a region where the surrogate believes an optimal value will be found, or should it explore regions of the space that reveal more about how $eval$ behaves, perhaps discovering even better values? That is, acquisition functions balance a tradeoff between exploration—by selecting $\mathbf{w}$ in the regions where the uncertainty of the surrogate model is high, and exploitation—by querying the regions where the model prediction is high.

Popular choices for the surrogate model are Gaussian Processes (Rasmussen, 2006; Snoek et al., 2012, GP), providing convenient and powerful prior distribution on functions, and tree-structured Parzen estimators (Bergstra et al., 2011, TPE), tailored to handle conditional spaces. Choices of the acquisition functions include probability of improvement (Kushner, 1964), expected improvement (EI) (Močkus et al., 1978; Jones, 2001), GP upper confidence bound (Srinivas et al., 2010), Thompson sampling (Thompson, 1933), entropy search (Hennig and Schuler, 2012), and dynamic combinations of the above functions (Hoffman et al., 2011); see Shahriari et al. (2016) for an extensive comparison. Yogatama et al. (2015) found that the combination of EI as the acquisition function and TPE as the surrogate model performed favorably in Bayesian optimization of text representations; we follow this choice in our model.

Algorithm 1 Bayesian optimization

$\mathcal{H} \leftarrow \emptyset$ ▷ Initialize observation history
$\mathcal{A} \leftarrow EI$ ▷ Initialize acquisition function
$\mathcal{S}_0 \leftarrow TPE$ ▷ Initialize surrogate model
for t $\leftarrow$ 1 to T **do**
  $\mathbf{w}_t \leftarrow \text{argmax}_{\mathbf{w}} \mathcal{A}(\mathbf{w}; \mathcal{S}_{t-1}, \mathcal{H})$ ▷ Predict $\mathbf{w}_t$ by optimizing acquisition function
  $eval(\mathbf{w}_t)$ ▷ Evaluate $\mathbf{w}_t$ on extrinsic task
  $\mathcal{H} \leftarrow \mathcal{H} \cup (\mathbf{w}_t, eval(\mathbf{w}_t))$ ▷ Update observation history
  Estimate $\mathcal{S}_t$ given $\mathcal{H}$
end for
return $\mathcal{H}$

2.2 Distributional and Linguistic Features

To characterize and quantify a curriculum, we define three categories of features, focusing on various distributional, syntactic, and semantic aspects of training data. We now detail the feature categories along with motivations for feature selection.

DIVERSITY. Diversity measures capture the distributions of types in data. Entropy is the best-known measure of diversity in statistical research, but there are many others (Tang et al., 2006; Gimpel et al., 2013). Common measures of diversity are used in many contrasting fields, from ecology and biology (Rosenzweig, 1995; Magurran, 2013), to economics and social studies (Stirling, 2007). Diversity has been shown effective in related research on curriculum learning in language modeling, vision, and multimedia analysis (Bengio et al., 2009; Jiang et al., 2014).

Let p_i and p_j correspond to empirical frequencies of word types t_i and t_j in the training data. Let d_{ij} correspond to their semantic similarity, calculated as the cosine similarity between embeddings of t_i and t_j learned from the training data. We annotate each paragraph with the following diversity features:

- Number of word types: $\#types$
- Type-token ratio: $\frac{\#types}{\#tokens}$
- Entropy: $-\sum_i p_i \ln(p_i)$
- Simpson's index (Simpson, 1949): $\sum_i {p_i}^2$
- Quadratic entropy (Rao, 1982):[1] $\sum_{i,j} d_{ij} p_i p_j$

SIMPLICITY. Spitkovsky et al. (2010) have validated the utility of syntactic simplicity in curriculum learning for unsupervised grammar induction by showing that training on sentences in order of increasing lengths outperformed other orderings. We explore the simplicity hypothesis, albeit without prior assumptions on specific ordering of data, and extend it to additional simplicity/complexity measures of training data. Our features are inspired by prior research in second language acquisition, text simplification, and readability assessment (Schwarm and Ostendorf, 2005; Heilman et al., 2007; Pitler and Nenkova, 2008; Vajjala and

[1] Intuitively, this feature promotes paragraphs that contain semantically similar high-probability words.

Meurers, 2012). We use an off-the-shelf syntactic parser[2] (Zhang and Clark, 2011) to parse our training corpus. Then, the following features are used to measure phonological, lexical, and syntactic complexity of training paragraphs:

- Language model score
- Character language model score
- Average sentence length
- Verb-token ratio
- Noun-token ratio
- Parse tree depth
- Number of noun phrases: $\#NPs$
- Number of verb phrases: $\#VBs$
- Number of prepositional phrases: $\#PPs$

PROTOTYPICALITY. This is a group of semantic features that use insights from cognitive linguistics and child language acquisition. The goal is to characterize the curriculum of representation learning in terms of the curriculum of human language learning. We resort to the Prototype theory (Rosch, 1978), which posits that semantic categories include more central (or prototypical) as well as less prototypical words. For example, in the ANIMAL category, *dog* is more prototypical than *sloth* (because *dog* is more frequent); *dog* is more prototypical than *canine* (because *dog* is more concrete); and *dog* is more prototypical than *bull terrier* (because *dog* is less specific). According to the theory, more prototypical words are acquired earlier. We use lexical semantic databases to operationalize insights from the prototype theory in the following semantic features; the features are computed on token level and averaged over paragraphs:

- Age of acquisition (AoA) of words was extracted from the crowd-sourced database, containing over 50 thousand English words (Kuperman et al., 2012). For example, the AoA of *run* is 4.47 (years), of *flee* is 8.33, and of *abscond* is 13.36. If a word was not found in the database it was assigned the maximal age of 25.
- Concreteness ratings on the scale of 1–5 (1 is most abstract) for 40 thousand English lemmas (Brysbaert et al., 2014). For example, *cookie* is rated as 5, and *spirituality* as 1.07.
- Imageability ratings are taken from the MRC psycholinguistic database (Wilson, 1988). Following Tsvetkov et al. (2014), we used the MRC annotations as seed, and propagated the ratings to all vocabulary words using the word embeddings as features in an ℓ_2-regularized logistic regression classifier.
- Conventionalization features count the number of "conventional" words and phrases in a paragraph. Assuming that a Wikipedia title is a proxy to a conventionalized concept, we counted the number of existing titles (from a database of over 4.5 million titles) in the paragraph.
- Number of syllables scores are also extracted from the AoA database; out-of-database words were annotated as 5-syllable words.
- Relative frequency in a supersense was computed by marginalizing the word frequencies in the training corpus over coarse semantic categories defined in the WordNet (Fellbaum, 1998; Ciaramita and Altun, 2006). There are 41 supersense types: 26 for nouns and 15 for verbs, e.g., NOUN.ANIMAL and VERB.MOTION. For example, in NOUN.ANIMAL the relative frequency of *human* is 0.06, of *dog* is 0.01, of *bird* is 0.01, of *cattle* is 0.009, and of *bumblebee* is 0.0002.
- Relative frequency in a synset was calculated similarly to the previous feature category, but word frequencies were marginalized over WordNet synsets (more fine-grained synonym sets). For example, in the synset {*vet, warhorse, veteran, oldtimer, seasoned stager*}, *veteran* is the most prototypical word, scoring 0.87.

3 Evaluation Benchmarks

We evaluate the utility of the pretrained word embeddings as features in downstream NLP tasks. We choose the following off-the-shelf models that utilize pretrained word embeddings as features:

Sentiment Analysis (Senti). Socher et al. (2013) created a treebank of sentences annotated with fine-grained sentiment labels on phrases and sentences from movie review excerpts. The coarse-grained treebank of positive and negative classes has been split into training, development, and test datasets containing 6,920, 872, and 1,821 sentences, respectively. We use the average of the word vectors of a given sentence as a feature vector for classification (Faruqui et al., 2015; Sedoc

[2] http://http://people.sutd.edu.sg/~yue_zhang/doc

et al., 2016). The ℓ_2-regularized logistic regression classifier is tuned on the development set and accuracy is reported on the test set.

Named Entity Recognition (NER). Named entity recognition is the task of identifying proper names in a sentence, such as names of persons, locations etc. We use the recently proposed LSTM-CRF NER model (Lample et al., 2016) which trains a forward-backward LSTM on a given sequence of words (represented as word vectors), the hidden units of which are then used as (the only) features in a CRF model (Lafferty et al., 2001) to predict the output label sequence. We use the CoNLL 2003 English NER dataset (Tjong Kim Sang and De Meulder, 2003) to train our models and present results on the test set.

Part of Speech Tagging (POS). For POS tagging, we again use the LSTM-CRF model (Lample et al., 2016), but instead of predicting the named entity tag for every word in a sentence, we train the tagger to predict the POS tag of the word. The tagger is trained and evaluated with the standard Penn TreeBank (PTB) (Marcus et al., 1993) training, development and test set splits as described in Collins (2002).

Dependency Parsing (Parse). Dependency parsing is the task of identifying syntactic relations between the words of a sentence. For dependency parsing, we train the stack-LSTM parser of Dyer et al. (2015) for English on the universal dependencies v1.1 treebank (Agić et al., 2015) with the standard development and test splits, reporting unlabeled attachment scores (UAS) on the test data. We remove all part-of-speech and morphology features from the data, and prevent the model from optimizing the word embeddings used to represent each word in the corpus, thereby forcing the parser to rely completely on the pretrained embeddings.

4 Experiments

Data. All models were trained on Wikipedia articles, split to paragraph-per-line. Texts were cleaned, tokenized, numbers were normalized by replacing each digit with "DG", all types that occur less than 10 times were replaces by the "UNK" token, the data was not lowercased. We list data sizes in table 1.

# paragraphs	# tokens	# types
2,532,361	100,872,713	156,663

Table 1: Training data sizes.

Setup. 100-dimensional word embeddings were trained using the `cbow` model implemented in the word2vec toolkit (Mikolov et al., 2013).[3] All training data was used, either shuffled or ordered by a curriculum. As described in §3, we modified the extrinsic tasks to learn solely from word embeddings, without additional features. All models were learned under same conditions, across curricula: in Parse, NER, and POS we limited the number of training iterations to 3, 3, and 1, respectively. This setup allowed us to evaluate the effect of curriculum without additional interacting factors.

Experiments. In all the experiments we first train word embedding models, then the word embeddings are used as features in four extrinsic tasks (§3). We tune the tasks on development data, and report results on the test data. The only component that varies across the experiments is order of paragraphs in the training corpus—the curriculum. We compare the following experimental setups:

- **Shuffled** baselines: the curriculum is defined by random shuffling the training data. We shuffled the data 10 times, and trained 10 word embeddings models, each model was then evaluated on downstream tasks. Following Bengio et al. (2009), we report test results for the system that is closest to the median in dev scores. To evaluate variability and a range of scores that can be obtained from shuffling the data, we also report test results for systems that obtained the highest dev scores.
- **Sorted** baselines: the curriculum is defined by sorting the training data by sentence length in increasing/decreasing order, similarly to (Spitkovsky et al., 2010).
- **Coherent** baselines: the curriculum is defined by just concatenating Wikipedia articles. The goal of this experiment is to evaluate the importance of semantic coherence in training data.

[3]To evaluate the impact of curriculum learning, we enforced sequential processing of data organized in a predefined order of training examples. To control for sequential processing, word embedding were learned by running the `cbow` using a single thread for one iteration.

Our intuition is that a coherent curriculum can improve models, since words with similar meanings and similar contexts are grouped when presented to the learner.

- **Optimized curriculum** models: the curriculum is optimized using the BayesOpt. We evaluate and compare models optimized using features from one of the three feature groups (§2.2). As in the shuffled baselines, we fix the number of trials (here, BayesOpt iterations) to 10, and we report test results of systems that obtained best dev scores.

Results. Experimental results are listed in table 2. Most systems trained with curriculum substantially outperform the strongest of all baselines. These results are encouraging, given that all word embedding models were trained on the same set of examples, only in different order, and display the indirect influence of the data curriculum on downstream tasks. These results support our assumption that curriculum matters. Albeit not as pronounced as with optimized curriculum, sorting paragraphs by length can also lead to substantial improvements over random baselines, but there is no clear recipe on whether the models prefer curricula sorted in an increasing or decreasing order. These results also support the advantage of a task-specific optimization framework over a general, intuition-guided recipe. An interesting result, also, that shuffling is not essential: systems trained on coherent data are on par (or better) than the shuffled systems.[4] In the next section, we analyze these results qualitatively.

5 Analysis

What are task-specific curriculum preferences? We manually inspect learned features and curriculum-sorted corpora, and find that best systems are obtained when their embeddings are learned from curricula appropriate to the downstream tasks. We discuss below several examples.

POS and Parse systems converge to the same set of weights, when trained on features that provide various measures of syntactic simplicity. The features with highest coefficients (and thus the most important features in sorting) are $\#NPs$, Parse tree depth, $\#VPs$, and $\#PPs$ (in this order). The sign in the $\#NPs$ feature weight, however, is the opposite from the other three feature weights (i.e., sorted in different order). $\#NPs$ is sorted in the increasing order of the number of noun phrases in a paragraph, and the other features are sorted in the decreasing order. Since Wikipedia corpus contains a lot of partial phrases (titles and headings), such curriculum promotes more complex, full sentences, and demotes partial sentences.

Best Senti system is sorted by prototypicality features. Most important features (with the highest coefficients) are Concreteness, Relative frequency in a supersense, and the Number of syllables. First two are sorted in decreasing order (i.e. paragraphs are sorted from more to less concrete, and from more to less prototypical words), and the Number of syllables is sorted in increasing order (this also promotes simpler, shorter words which are more prototypical). We hypothesize that this sorting reflects the type of data that Sentiment analysis task is trained on: it is trained on movie reviews, that are usually written in a simple, colloquial language.

Unlike POS, Parse, and Senti systems, all NER systems prefer curricula in which texts are sorted from short to long paragraphs. The most important features in the best (simplicity-sorted) system are $\#PPs$ and Verb-token ratio, both sorted from less to more occurrences of prepositional and verb phrases. Interestingly, most of the top lines in the NER system curricula contain named entities, although none of our features mark named entities explicitly. We show top lines in the simplicity-optimized system in figure 2.

Finally, in all systems sorted by prototypicality, the last line is indeed not a prototypical word *Donaudampfschiffahrtselektrizitätenhauptbetriebswerkbauunterbeamtengesellschaft*, which is an actual word in German, frequently used as an example of compounding in synthetic languages, but rarely (or never?) used by German speakers.

Weighting examples according to curriculum. Another way to integrate curriculum in word em-

[4]Note that in the shuffled NER baselines, best dev results yield lower performance on the test data. This implies that in the standard development/test splits the development and test sets are not fully compatible or not large enough. We also observe this problem in the curriculum-optimized Parse-prototypicality and Senti-diversity systems. The dev scores for the Parse systems are 76.99, 76.47, 76.47 for diversity, prototypicality, and simplicity, respectively, but the prototypicality-sorted parser performs poorly on test data. Similarly in the sentiment analysis task, the dev scores are 69.15, 69.04, 69.49 for diversity, prototypicality, and simplicity feature groups. Senti-diversity scores, however, are lower on the test data, although the dev results are better than in Senti-simplicity. This limitation of the standard dev/test splits is beyond the scope of this paper.

		Senti	NER	POS	Parse
Shuffled	median	66.01	85.88	96.35	75.08
	best	66.61	85.50	96.38	76.40
Sorted	long→short	66.78	85.22	96.47	75.85
	short→long	66.12	85.49	96.20	75.31
Coherent	original order	66.23	85.99	96.47	76.08
Optimized curriculum	diversity	66.06	86.09	96.59	**76.63**
	prototypicality	**67.44**	85.96	96.53	75.81
	simplicity	67.11	**86.42**	**96.62**	76.54

Table 2: Evaluation of the impact of the curriculum of word embeddings on the downstream tasks.

Trimingham " Golf " ball .
Adélie penguin
" **Atriplex** " leaf UNK UNK
Hồng Lĩnh mountain
Anneli Jäätteenmäki UNK cabinet
Gävle goat
Early telescope observations .
Scioptric ball
Matryoshka doll
Luxembourgian passport
Australian Cashmere goat
Plumbeous water redstart
Dagebüll lighthouse
Vecom FollowUs . tv
Syracuse Junction railroad .
San Clemente Island goat
Tychonoff plank

Figure 2: Most of the top lines in best-scoring NER system contain named entities, although our features do not annotate named entities explicitly.

bedding training is to weight training examples according to curriculum during word representation training. We modify the `cbow` objective $\sum_{t=1}^{T} \log p(w_t|w_{t-c}..w_{t+c})$ as follows:[5]

$$\sum_{t=1}^{T}(\frac{1}{1+e^{-weight(w_t)}}+\lambda)\log p(w_t|w_{t-c}..w_{t+c})$$

Here, $weight(w_t)$ denotes the score attributed to the token w_t, which is the z-normalized score of the paragraph; λ=0.5 is determined empirically. $\log p(w_t)|w_{t-c}..w_{t+c})$ computes the probability of predicting word w_t, using the context of c words to the left and right of w_t. Notice that this quantity is no longer a proper probability, as we are not normalizing over the weights $weight(w_t)$ over all tokens. However, the optimization in word2vec is performed using stochastic gradient descent, optimizing for a single token at each iteration. This yields a normalizer of 1 for each iteration, yielding the same gradient as the original `cbow` model.

We retrain our best curriculum-sorted systems with the modified objective, also controlling for curriculum. The results are shown in table 3. We find that the benefit of integrating curriculum in training objective of word representations is not evident across tasks: Senti and NER systems trained on vectors with the modified objective substantially outperform best results in table 2; POS and Parse perform better than the baselines but worse than the systems with the original objective.

	Senti	NER	POS	Parse
curriculum	67.44	86.42	**96.62**	**76.63**
`cbow`+curric	**68.26**	**86.49**	96.48	76.54

Table 3: Evaluation of the impact of curriculum integrated in the `cbow` objective.

Are we learning task-specific curricula? One way to assess whether we learn meaningful task-specific curriculum preferences is to compare curricula learned by one downstream task across different feature groups. If learned curricula are similar in, say, NER system, despite being optimized once using diversity features and once using prototypicality features—two disjoint feature sets—we can infer that the NER task prefers word embeddings learned from examples presented in a certain order, regardless of specific optimization features. For each downstream task, we thus measure Spearman's rank correlation between the curricula optimized using diversity (D), or prototypicality (P), or simplicity (S) feature sets. Prior to measuring correlations, we remove duplicate lines from

[5]The modified word2vec tool is located at `https://github.com/wlin12/wang2vec`.

the training corpora. Correlation results across tasks and across feature sets are shown in table 4.

The general pattern of results is that if two systems score higher than baselines, training sentences of their feature embeddings have similar curricula (i.e., the Spearman's ρ is positive), and if two systems disagree (one is above and one is below the baseline), then their curricula also disagree (i.e., the Spearman's ρ is negative or close to zero). NER systems all outperform the baselines and their curricula have high correlations. Moreover, NER sorted by diversity and simplicity have better scores than NER sorted by prototypicality, and in line with these results $\rho(\text{S,D})_{NER} > \rho(\text{P,S})_{NER}$ and $\rho(\text{S,D})_{NER} > \rho(\text{D,P})_{NER}$. Similar pattern of results is in POS correlations. In Parse systems, also, diversity and simplicity features yielded best parsing results, and $\rho(\text{S,D})_{Parse}$ has high positive correlation. The prototypicality-optimized parser performed poorly, and its correlations with better systems are negative. The best parser was trained using the diversity-optimized curriculum, and thus $\rho(\text{D,P})_{Parse}$ is the lowest. Senti results follow similar pattern of curricula correlations.

	Senti	**NER**	**POS**	**Parse**
ρ(D, P)	-0.68	0.76	0.66	-0.76
ρ(P, S)	0.33	0.75	0.75	-0.45
ρ(S, D)	-0.16	0.81	0.51	0.67

Table 4: Curricula correlations across feature groups.

Curriculum learning vs. data selection. We compare the task of curriculum learning to the task of data selection (reducing the set of training instances to more important or cleaner examples). We reduce the training data to the subset of 10% of tokens, and train downstream tasks on the reduced training sets. We compare system performance trained using the top 10% of tokens in the best curriculum-sorted systems (Senti-prototypicality, NER-implicity, POS-simplicity, Parse-diversity) to the systems trained using the top 10% of tokens in a corpus with randomly shuffled paragraphs.[6] The results are listed in table 5.

The curriculum-based systems are better in POS and in Parse systems, mainly because these tasks prefer vectors trained on curricula that promote well-formed sentences (as discussed above). Conversely, NER prefers vectors trained on corpora that begin with named entities, so most of the tokens in the reduced training data are constituents in short noun phrases. These results suggest that the tasks of data selection and curriculum learning are different. Curriculum is about strong initialization of the models and time-course learning, which is not necessarily sufficient for data reduction.

	Senti	**NER**	**POS**	**Parse**
random	63.97	**82.35**	96.22	69.11
curriculum	**64.47**	76.96	**96.55**	**72.93**

Table 5: Data selection results.

6 Related Work

Two prior studies on curriculum learning in NLP are discussed in the paper (Bengio et al., 2009; Spitkovsky et al., 2010). Curriculum learning and related research on self-paced learning has been explored more deeply in computer vision (Bengio et al., 2009; Kumar et al., 2010; Lee and Grauman, 2011) and in multimedia analysis (Jiang et al., 2015). Bayesian optimization has also received little attention in NLP. GPs were used in the task of machine translation quality estimation (Cohn and Specia, 2013) and in temporal analysis of social media texts (Preotiuc-Pietro and Cohn, 2013); TPEs were used by Yogatama et al. (2015) for optimizing choices of feature representations—n-gram size, regularization choice, etc.—in supervised classifiers.

7 Conclusion

We used Bayesian optimization to optimize curricula for training dense distributed word representations, which, in turn, were used as the sole features in NLP tasks. Our experiments confirmed that better curricula yield stronger models. We also conducted an extensive analysis, which sheds better light on understanding of text properties that are beneficial for model initialization. The proposed novel technique for finding an optimal curriculum is general, and can be used with other datasets and models.

[6] Top $n\%$ tokens are used rather than top $n\%$ paragraphs because in all tasks except NER curriculum-sorted corpora begin with longer paragraphs. Thus, with top $n\%$ paragraphs our systems would have an advantage over random systems due to larger vocabulary sizes and not necessarily due to a better subset of data.

Acknowledgments

This work was supported by the National Science Foundation through award IIS-1526745. We are grateful to Nathan Schneider, Guillaume Lample, Waleed Ammar, Austin Matthews, and the anonymous reviewers for their insightful comments.

References

Željko Agić, Maria Jesus Aranzabe, Aitziber Atutxa, Cristina Bosco, Jinho Choi, Marie-Catherine de Marneffe, Timothy Dozat, Richárd Farkas, Jennifer Foster, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Jan Hajič, Anders Trærup Johannsen, Jenna Kanerva, Juha Kuokkala, Veronika Laippala, Alessandro Lenci, Krister Lindén, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Héctor Alonso Martínez, Ryan McDonald, Anna Missilä, Simonetta Montemagni, Joakim Nivre, Hanna Nurmi, Petya Osenova, Slav Petrov, Jussi Piitulainen, Barbara Plank, Prokopis Prokopidis, Sampo Pyysalo, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Kiril Simov, Aaron Smith, Reut Tsarfaty, Veronika Vincze, and Daniel Zeman. 2015. Universal dependencies 1.1. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proc. ICML*, pages 41–48.

James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proc. NIPS*, pages 2546–2554.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proc. EMNLP*, pages 594–602.

Trevor Cohn and Lucia Specia. 2013. Modelling annotator bias with multi-task Gaussian processes: An application to machine translation quality estimation. In *Proc. ACL*, pages 32–42.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. ACL*.

Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. NAACL*.

Christiane Fellbaum, editor. 1998. *WordNet: an electronic lexical database*. MIT Press.

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proc. EMNLP*, pages 1100–1111.

Michael J. Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language texts. In *Proc. NAACL*, pages 460–467.

Philipp Hennig and Christian J Schuler. 2012. Entropy search for information-efficient global optimization. *The Journal of Machine Learning Research*, 13(1):1809–1837.

Matthew D Hoffman, Eric Brochu, and Nando de Freitas. 2011. Portfolio allocation for Bayesian optimization. In *Proc. UAI*, pages 327–336.

Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014. Self-paced learning with diversity. In *Proc. NIPS*, pages 2078–2086.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. 2015. Self-paced curriculum learning. In *Proc. AAAI*, volume 2, page 6.

Donald R Jones. 2001. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, 21(4):345–383.

Robert Kail. 1990. *The development of memory in children*. W. H. Freeman and Company, 3rd edition.

M. P. Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Proc. NIPS*, pages 1189–1197.

Victor Kuperman, Hans Stadthagen-Gonzalez, and Marc Brysbaert. 2012. Age-of-acquisition ratings for 30,000 english words. *Behavior Research Methods*, 44(4):978–990.

Harold J Kushner. 1964. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. NAACL*.

Yong Jae Lee and Kristen Grauman. 2011. Learning the easy things first: Self-paced visual category discovery. In *Proc. CVPR*, pages 1721–1728.

Anne E Magurran. 2013. *Measuring biological diversity*. John Wiley & Sons.

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

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

Jonas Močkus, Vytautas Tiesis, and Antanas Žilinskas. 1978. On Bayesian methods for seeking the extremum. *Towards global optimization*, 2(117-129):2.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proc. EMNLP*, pages 186–195.

Daniel Preotiuc-Pietro and Trevor Cohn. 2013. A temporal model of text periodicities using gaussian processes. In *Proc. EMNLP*, pages 977–988.

C Radhakrishna Rao. 1982. Diversity and dissimilarity coefficients: a unified approach. *Theoretical population biology*, 21(1):24–43.

Carl Edward Rasmussen. 2006. Gaussian Processes for machine learning.

Eleanor Rosch. 1978. Principles of categorization. In Eleanor Rosch and Barbara B. Lloyd, editors, *Cognition and categorization*, pages 28–71.

Michael L Rosenzweig. 1995. *Species diversity in space and time*. Cambridge University Press.

Sarah E. Schwarm and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *Proc. ACL*, pages 523–530.

João Sedoc, Jean Gallier, Lyle Ungar, and Dean Foster. 2016. Semantic word clusters using signed normalized graph cuts. *arXiv preprint arXiv:1601.05403*.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. 2016. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE*, 104(1):148–175.

Edward H Simpson. 1949. Measurement of diversity. *Nature*.

Burrhus Frederic Skinner. 1938. The behavior of organisms: an experimental analysis. *An Experimental Analysis*.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Proc. NIPS*, pages 2951–2959.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP*.

Valentin I Spitkovsky, Hiyan Alshawi, and Dan Jurafsky. 2010. From baby steps to leapfrog: How less is more in unsupervised dependency parsing. In *Proc. NAACL*, pages 751–759.

Niranjan Srinivas, Andreas Krause, Sham M Kakade, and Matthias Seeger. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, pages 1015–1022.

Andy Stirling. 2007. A general framework for analysing diversity in science, technology and society. *Journal of the Royal Society Interface*, 4(15):707–719.

E Ke Tang, Ponnuthurai N Suganthan, and Xin Yao. 2006. An analysis of diversity measures. *Machine Learning*, 65(1):247–271.

William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. CoNLL*.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proc. ACL*.

Sowmya Vajjala and Detmar Meurers. 2012. On improving the accuracy of readability classification using insights from second language acquisition. In *Proc. BEA*, pages 163–173.

Michael Wilson. 1988. MRC psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–10.

Dani Yogatama, Lingpeng Kong, and Noah A Smith. 2015. Bayesian optimization of text representations. In *Proc. EMNLP*, pages 2100–2105.

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

Pointing the Unknown Words

Caglar Gulcehre
Université de Montréal

Sungjin Ahn
Université de Montréal

Ramesh Nallapati
IBM T.J. Watson Research

Bowen Zhou
IBM T.J. Watson Research

Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Abstract

The problem of rare and unknown words is an important issue that can potentially effect the performance of many NLP systems, including traditional count-based and deep learning models. We propose a novel way to deal with the rare and unseen words for the neural network models using attention. Our model uses two softmax layers in order to predict the next word in conditional language models: one predicts the location of a word in the source sentence, and the other predicts a word in the shortlist vocabulary. At each timestep, the decision of which softmax layer to use is adaptively made by an MLP which is conditioned on the context. We motivate this work from a psychological evidence that humans naturally have a tendency to point towards objects in the context or the environment when the name of an object is not known. Using our proposed model, we observe improvements on two tasks, neural machine translation on the Europarl English to French parallel corpora and text summarization on the Gigaword dataset.

1 Introduction

Words are the basic input/output units in most of the NLP systems, and thus the ability to cover a large number of words is a key to building a robust NLP system. However, considering that (i) the number of all words in a language including named entities is very large and that (ii) language itself is an evolving system (people create new words), this can be a challenging problem.

A common approach followed by the recent neural network based NLP systems is to use a softmax output layer where each of the output dimension corresponds to a word in a predefined word-shortlist. Because computing high dimensional softmax is computationally expensive, in practice the shortlist is limited to have only top-K most frequent words in the training corpus. All other words are then replaced by a special word, called the *unknown word (UNK)*.

The shortlist approach has two fundamental problems. The first problem, which is known as the *rare word* problem, is that some of the words in the shortlist occur less frequently in the training set and thus are difficult to learn a good representation, resulting in poor performance. Second, it is obvious that we can lose some important information by mapping different words to a single dummy token UNK. Even if we have a very large shortlist including all unique words in the training set, it does not necessarily improve the test performance, because there still exists a chance to see an unknown word at test time. This is known as the *unknown word* problem. In addition, increasing the shortlist size mostly leads to increasing rare words due to Zipf's Law.

These two problems are particularly critical in language understanding tasks such as factoid question answering (Bordes et al., 2015) where the words that we are interested in are often named entities which are usually unknown or rare words.

In a similar situation, where we have a limited information on how to call an object of interest, it seems that humans (and also some primates) have an efficient behavioral mechanism of drawing attention to the object: *pointing* (Matthews et al., 2012). Pointing makes it possible to deliver information and to associate context to a particular object without knowing how to call it. In particular, human infants use pointing as a fundamental communication tool (Tomasello et al., 2007).

In this paper, inspired by the pointing behavior of humans and recent advances in the atten-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 140–149,
Berlin, Germany, August 7-12, 2016.

tion mechanism (Bahdanau et al., 2014) and the pointer networks (Vinyals et al., 2015), we propose a novel method to deal with the rare or unknown word problem. The basic idea is that we can see many NLP problems as a task of predicting target text given context text, where some of the target words appear in the context as well. We observe that in this case we can make the model *learn to point* a word in the context and copy it to the target text, as well as *when to point*. For example, in machine translation, we can see the source sentence as the context, and the target sentence as what we need to predict. In Figure 1, we show an example depiction of how words can be copied from source to target in machine translation. Although the source and target languages are different, many of the words such as named entities are usually represented by the same characters in both languages, making it possible to copy. Similarly, in text summarization, it is natural to use some words in the original text in the summarized text as well.

Specifically, to predict a target word at each timestep, our model first determines the source of the word generation, that is, whether to take one from a predefined shortlist or to copy one from the context. For the former, we apply the typical softmax operation, and for the latter, we use the attention mechanism to obtain the pointing softmax probability over the context words and pick the one of high probability. The model learns this decision so as to use the pointing only when the context includes a word that can be copied to the target. This way, our model can predict even the words which are not in the shortlist, as long as it appears in the context. Although some of the words still need to be labeled as UNK, i.e., if it is neither in the shortlist nor in the context, in experiments we show that this *learning when and where to point* improves the performance in machine translation and text summarization.

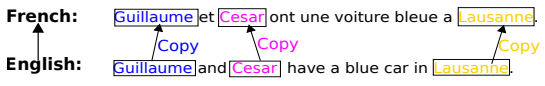


Figure 1: An example of how copying can happen for machine translation. Common words that appear both in source and the target can directly be copied from input to source. The rest of the unknown in the target can be copied from the input after being translated with a dictionary.

The rest of the paper is organized as follows. In the next section, we review the related works including pointer networks and previous approaches to the rare/unknown problem. In Section 3, we review the neural machine translation with attention mechanism which is the baseline in our experiments. Then, in Section 4, we propose our method dealing with the rare/unknown word problem, called the **Pointer Softmax (PS)**. The experimental results are provided in the Section 5 and we conclude our work in Section 6.

2 Related Work

The attention-based pointing mechanism is introduced first in the pointer networks (Vinyals et al., 2015). In the pointer networks, the output space of the target sequence is constrained to be the observations in the input sequence (not the input space). Instead of having a fixed dimension softmax output layer, softmax outputs of varying dimension is dynamically computed for each input sequence in such a way to maximize the attention probability of the target input. However, its applicability is rather limited because, unlike our model, there is no option to choose whether to point or not; it always points. In this sense, we can see the pointer networks as a special case of our model where we always choose to point a context word.

Several approaches have been proposed towards solving the rare words/unknown words problem, which can be broadly divided into three categories. The first category of the approaches focuses on improving the computation speed of the softmax output so that it can maintain a very large vocabulary. Because this only increases the shortlist size, it helps to mitigate the unknown word problem, but still suffers from the rare word problem. The hierarchical softmax (Morin and Bengio, 2005), importance sampling (Bengio and Senécal, 2008; Jean et al., 2014), and the noise contrastive estimation (Gutmann and Hyvärinen, 2012; Mnih and Kavukcuoglu, 2013) methods are in the class.

The second category, where our proposed method also belongs to, uses information from the context. Notable works are (Luong et al., 2015) and (Hermann et al., 2015). In particular, applying to machine translation task, (Luong et al., 2015) learns to point some words in source sentence and copy it to the target sentence, similarly to our method. However, it does not use attention mechanism, and by having fixed sized soft-

max output over the relative pointing range (e.g., -7, ..., -1, 0, 1, ..., 7), their model (the Positional All model) has a limitation in applying to more general problems such as summarization and question answering, where, unlike machine translation, the length of the context and the pointing locations in the context can vary dramatically. In question answering setting, (Hermann et al., 2015) have used placeholders on named entities in the context. However, the placeholder id is directly predicted in the softmax output rather than predicting its location in the context.

The third category of the approaches changes the unit of input/output itself from words to a smaller resolution such as characters (Graves, 2013) or bytecodes (Sennrich et al., 2015; Gillick et al., 2015). Although this approach has the main advantage that it could suffer less from the rare/unknown word problem, the training usually becomes much harder because the length of sequences significantly increases.

Simultaneously to our work, (Gu et al., 2016) and (Cheng and Lapata, 2016) proposed models that learn to copy from source to target and both papers analyzed their models on summarization tasks.

3 Neural Machine Translation Model with Attention

As the baseline neural machine translation system, we use the model proposed by (Bahdanau et al., 2014) that learns to (soft-)align and translate jointly. We refer this model as NMT.

The encoder of the NMT is a bidirectional RNN (Schuster and Paliwal, 1997). The forward RNN reads input sequence $\mathbf{x} = (x_1, \ldots, x_T)$ in left-to-right direction, resulting in a sequence of hidden states $(\overrightarrow{\mathbf{h}}_1, \ldots, \overrightarrow{\mathbf{h}}_T)$. The backward RNN reads $\mathbf{x}$ in the reversed direction and outputs $(\overleftarrow{\mathbf{h}}_1, \ldots, \overleftarrow{\mathbf{h}}_T)$. We then concatenate the hidden states of forward and backward RNNs at each time step and obtain a sequence of *annotation* vectors $(\mathbf{h}_1, \ldots, \mathbf{h}_T)$ where $\mathbf{h}_j = \left[\overrightarrow{\mathbf{h}}_j || \overleftarrow{\mathbf{h}}_j\right]$. Here, $||$ denotes the concatenation operator. Thus, each annotation vector $\mathbf{h}_j$ encodes information about the j-th word with respect to all the other surrounding words in both directions.

In the decoder, we usually use gated recurrent unit (GRU) (Cho et al., 2014; Chung et al., 2014). Specifically, at each time-step t, the soft-alignment mechanism first computes the relevance weight e_{tj} which determines the contribution of annotation vector $\mathbf{h}_j$ to the t-th target word. We use a non-linear mapping f (e.g., MLP) which takes $\mathbf{h}_j$, the previous decoder's hidden state $\mathbf{s}_{t-1}$ and the previous output y_{t-1} as input:

$$e_{tj} = f(\mathbf{s}_{t-1}, \mathbf{h}_j, y_{t-1}).$$

The outputs e_{tj} are then normalized as follows:

$$l_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T} \exp(e_{tk})}. \quad (1)$$

We call l_{tj} as the relevance score, or the alignment weight, of the j-th annotation vector.

The relevance scores are used to get the *context vector* $\mathbf{c}_t$ of the t-th target word in the translation:

$$\mathbf{c}_t = \sum_{j=1}^{T} l_{tj} \mathbf{h}_j \ ,$$

The hidden state of the decoder $\mathbf{s}_t$ is computed based on the previous hidden state $\mathbf{s}_{t-1}$, the context vector $\mathbf{c}_t$ and the output word of the previous time-step y_{t-1}:

$$\mathbf{s}_t = f_r(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t), \quad (2)$$

where f_r is GRU.

We use a deep output layer (Pascanu et al., 2013) to compute the conditional distribution over words:

$$p(y_t = a | y_{<t}, \mathbf{x}) \propto \exp\left(\psi^a_{(\mathbf{W}_o, \mathbf{b}_o)} f_o(\mathbf{s}_t, y_{t-1}, \mathbf{c}_t)\right), \quad (3)$$

where $\mathbf{W}$ is a learned weight matrix and $\mathbf{b}$ is a bias of the output layer. f_o is a single-layer feed-forward neural network. $\psi_{(\mathbf{W}_o, \mathbf{b}_o)}(\cdot)$ is a function that performs an affine transformation on its input. And the superscript a in ψ^a indicates the a-th column vector of ψ.

The whole model, including both the encoder and the decoder, is jointly trained to maximize the (conditional) log-likelihood of target sequences given input sequences, where the training corpus is a set of $(\mathbf{x}_n, \mathbf{y}_n)$'s. Figure 2 illustrates the architecture of the NMT.

4 The Pointer Softmax

In this section, we introduce our method, called as the pointer softmax (PS), to deal with the rare and unknown words. The pointer softmax can be

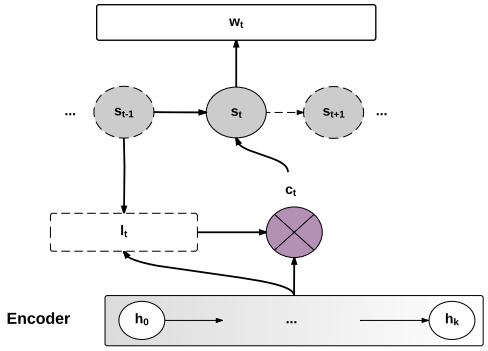


Figure 2: A depiction of neural machine translation architecture with attention. At each timestep, the model generates the attention weights $\mathbf{l}_t$. We use $\mathbf{l}_t$ the encoder's hidden state to obtain the context $\mathbf{c}_t$. The decoder uses $\mathbf{c}_t$ to predict a vector of probabilities for the words $\mathbf{w}_t$ by using softmax.

applicable approach to many NLP tasks, because it resolves the limitations about unknown words for neural networks. It can be used in parallel with other existing techniques such as the large vocabulary trick (Jean et al., 2014). Our model learns two key abilities jointly to make the pointing mechanism applicable in more general settings: (i) to predict whether it is required to use the pointing or not at each time step and (ii) to point any location of the context sequence whose length can vary widely over examples. Note that the pointer networks (Vinyals et al., 2015) are in lack of the ability (i), and the ability (ii) is not achieved in the models by (Luong et al., 2015).

To achieve this, our model uses two softmax output layers, the *shortlist* softmax and the *location* softmax. The shortlist softmax is the same as the typical softmax output layer where each dimension corresponds a word in the predefined word shortlist. The location softmax is a pointer network where each of the output dimension corresponds to the location of a word in the context sequence. Thus, the output dimension of the location softmax varies according to the length of the given context sequence.

At each time-step, if the model decides to use the shortlist softmax, we generate a word w_t from the shortlist. Otherwise, if it is expected that the context sequence contains a word which needs to be generated at the time step, we obtain the location of the context word l_t from the location softmax. The key to making this possible is deciding when to use the shortlist softmax or the location softmax at each time step. In order to accomplish this, we introduce a switching network to the model. The switching network, which is a multilayer perceptron in our experiments, takes the representation of the context sequence (similar to the input annotation in NMT) and the previous hidden state of the output RNN as its input. It outputs a binary variable z_t which indicates whether to use the shortlist softmax (when $z_t = 1$) or the location softmax (when $z_t = 0$). Note that if the word that is expected to be generated at each timestep is neither in the shortlist nor in the context sequence, the switching network selects the shortlist softmax, and then the shortlist softmax predicts UNK. The details of the pointer softmax model can be seen in Figure 3 as well.

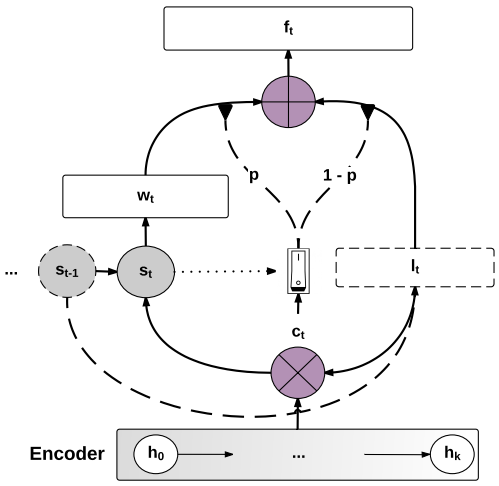


Figure 3: A simple depiction of the Pointer Softmax(PS) architecture. At each timestep as usual$\mathbf{l}_t$, $\mathbf{c}_t$ and the $\mathbf{w}_t$ for the words over the limited vocabulary(shortlist) is being generated. We have an additional switching variable z_t that decides whether to use $\mathbf{w}_t$ or copy the word from the input via $\mathbf{l}_t$. The final word prediction will be performed via pointer softmax $\mathbf{f}_t$ which can either copy the word from the source or predict the word from the shortlist vocabulary.

More specifically, our goal is to maximize the probability of observing the target word sequence $\mathbf{y} = (y_1, y_2, \ldots, y_{T_y})$ and the word generation source $\mathbf{z} = (z_1, z_2, \ldots, z_{T_y})$, given the context sequence $\mathbf{x} = (x_1, x_2, \ldots, x_{T_x})$:

$$p_\theta(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \prod_{t=1}^{T_y} p_\theta(y_t, z_t|y_{<t}, z_{<t}, \mathbf{x}). \quad (4)$$

Note that the word observation y_t can be either a word w_t from the shortlist softmax or a location l_t from the location softmax, depending on the switching variable z_t.

Considering this, we can factorize the above equation further

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \prod_{t \in \mathcal{T}_w} p(w_t, z_t|(y,z)_{<t}, \mathbf{x}) \times \prod_{t' \in \mathcal{T}_l} p(l_{t'}, z_{t'}|(y,z)_{<t'}, \mathbf{x}). \quad (5)$$

Here, $\mathcal{T}_w$ is a set of time steps where $z_t = 1$, and $\mathcal{T}_l$ is a set of time-steps where $z_t = 0$. And, $\mathcal{T}_w \cup \mathcal{T}_l = \{1, 2, \ldots, T_y\}$ and $\mathcal{T}_w \cap \mathcal{T}_l = \emptyset$. We denote all previous observations at step t by $(y,z)_{<t}$. Note also that $\mathbf{h}_t = f((y,z)_{<t})$.

Then, the joint probabilities inside each product can be further factorized as follows:

$$p(w_t, z_t|(y,z)_{<t}) = p(w_t|z_t = 1, (y,z)_{<t}) \times p(z_t = 1|(y,z)_{<t}) \quad (6)$$

$$p(l_t, z_t|(y,z)_{<t}) = p(l_t|z_t = 0, (y,z)_{<t}) \times p(z_t = 0|(y,z)_{<t}) \quad (7)$$

here, we omitted $\mathbf{x}$ which is conditioned on all probabilities in the above.

The switch probability is modeled as a multilayer perceptron with binary output:

$$p(z_t = 1|(y,z)_{<t}, \mathbf{x}) = \sigma(f(\mathbf{x}, \mathbf{h}_{t-1}; \theta)) \quad (8)$$

$$p(z_t = 0|(y,z)_{<t}, \mathbf{x}) = 1 - \sigma(f(\mathbf{x}, \mathbf{h}_{t-1}; \theta)). \quad (9)$$

And $p(w_t|z_t = 1, (y,z)_{<t}, \mathbf{x})$ is the shortlist softmax and $p(l_t|z_t = 0, (y,z)_{<t}, \mathbf{x})$ is the location softmax which can be a pointer network. $\sigma(\cdot)$ stands for the sigmoid function, $\sigma(x) = \frac{1}{\exp(-x)+1}$.

Given N such context and target sequence pairs, our training objective is to maximize the following log likelihood w.r.t. the model parameter θ

$$\arg\max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_\theta(y_n, z_n|x_n). \quad (10)$$

4.1 Basic Components of the Pointer Softmax

In this section, we discuss practical details of the three fundamental components of the pointer softmax. The interactions between these components and the model is depicted in Figure 3.

Location Softmax $\mathbf{l}_t$: The location of the word to copy from source text to the target is predicted by the location softmax $\mathbf{l}_t$. The location softmax outputs the conditional probability distribution $p(l_t|z_t = 0, (y,z)_{<t}, \mathbf{x})$. For models using the attention mechanism such as NMT, we can reuse the probability distributions over the source words in order to predict the location of the word to point. Otherwise we can simply use a pointer network of the model to predict the location.

Shortlist Softmax $\mathbf{w}_t$: The subset of the words in the vocabulary V is being predicted by the shortlist softmax $\mathbf{w}_t$.

Switching network d_t : The switching network d_t is an MLP with sigmoid output function that outputs a scalar probability of switching between $\mathbf{l}_t$ and $\mathbf{w}_t$, and represents the conditional probability distribution $p(z_t|(y,z)_{<t}, \mathbf{x})$. For NMT model, we condition the MLP that outputs the switching probability on the representation of the context of the source text $\mathbf{c}_t$ and the hidden state of the decoder $\mathbf{h}_t$. Note that, during the training, d_t is observed, and thus we do not have to sample.

The output of the pointer softmax, $\mathbf{f}_t$ will be the concatenation of the the two vectors, $d_t \times \mathbf{w}_t$ and $(1 - d_t) \times \mathbf{l}_t$.

At test time, we compute Eqn. (6) and (7) for all shortlist word w_t and all location l_t, and pick the word or location of the highest probability.

5 Experiments

In this section, we provide our main experimental results with the pointer softmax on machine translation and summarization tasks. In our experiments, we have used the same baseline model and just replaced the softmax layer with pointer softmax layer at the language model. We use the Adadelta (Zeiler, 2012) learning rule for the training of NMT models. The code for pointer softmax model is available at `https://github.com/caglar/pointer_softmax`.

5.1 The Rarest Word Detection

We construct a synthetic task and run some preliminary experiments in order to compare the results with the pointer softmax and the regular softmax's performance for the rare-words. The vocabulary size of our synthetic task is $|V| = 600$ using sequences of length 7. The words in the sequences are sampled according to their unigram distribu-

tion which has the form of a geometric distribution. The task is to predict the least frequent word in the sequence according to unigram distribution of the words. During the training, the sequences are generated randomly. Before the training, validation and test sets are constructed with a fixed seed.

We use a GRU layer over the input sequence and take the last-hidden state, in order to get the summary $\mathbf{c}_t$ of the input sequence. The $\mathbf{w}_t$, $\mathbf{l}_t$ are only conditioned on $\mathbf{c}_t$, and the MLP predicting the d_t is conditioned on the latent representations of $\mathbf{w}_t$ and $\mathbf{l}_t$. We use minibatches of size 250 using adam adaptive learning rate algorithm (Kingma and Adam, 2015) using the learning rate of 8×10^{-4} and hidden layers with 1000 units.

We train a model with pointer softmax where we assign pointers for the rarest 60 words and the rest of the words are predicted from the shortlist softmax of size 540. We observe that increasing the inverse temperature of the sigmoid output of d_t to 2, in other words making the decisions of d_t to become sharper, works better, i.e. $d_t = \sigma(2x)$.

At the end of training with pointer softmax we obtain the error rate of 17.4% and by using softmax over all 600 tokens, we obtain the error-rate of 48.2%.

5.2 Summarization

In these series of experiments, we use the annotated Gigaword corpus as described in (Rush et al., 2015). Moreover, we use the scripts that are made available by the authors of (Rush et al., 2015) [1] to preprocess the data, which results to approximately 3.8M training examples. This script generates about 400K validation and an equal number of test examples, but we use a randomly sampled subset of 2000 examples each for validation and testing. We also have made small modifications to the script to extract not only the tokenized words, but also system-generated named-entity tags. We have created two different versions of training data for pointers, which we call UNK-pointers data and entity-pointers data respectively.

For the UNK-pointers data, we trim the vocabulary of the source and target data in the training set and replace a word by the UNK token whenever a word occurs less than 5 times in either source or target data separately. Then, we create pointers from each UNK token in the target data to the position in the corresponding source document where the same word occurs in the source, as seen in the data before UNKs were created. It is possible that the source can have an UNK in the matching position, but we still created a pointer in this scenario as well. The resulting data has 2.7 pointers per 100 examples in the training set and 9.1 pointers rate in the validation set.

In the entity-pointers data, we exploit the named-entity tags in the annotated corpus and first anonymize the entities by replacing them with an integer-id that always starts from 1 for each document and increments from left to right. Entities that occur more than once in a single document share the same id. We create the anonymization at token-level, so as to allow partial entity matches between the source and target for multi-token entities. Next, we create a pointer from the target to source on similar lines as before, but only for exact matches of the anonymized entities. The resulting data has 161 pointers per 100 examples in the training set and 139 pointers per 100 examples in the validation set.

If there are multiple matches in the source, either in the UNK-pointers data or the entity-pointers data, we resolve the conflict in favor of the first occurrence of the matching word in the source document. In the UNK data, we model the UNK tokens on the source side using a single placeholder embedding that is shared across all documents, and in the entity-pointers data, we model each entity-id in the source by a distinct placeholder, each of which is shared across all documents.

In all our experiments, we use a bidirectional GRU-RNN (Chung et al., 2014) for the encoder and a uni-directional RNN for the decoder. To speed-up training, we use the large-vocabulary trick (Jean et al., 2014) where we limit the vocabulary of the softmax layer of the decoder to 2000 words dynamically chosen from the words in the source documents of each batch and the most common words in the target vocabulary. In both experiments, we fix the embedding size to 100 and the hidden state dimension to 200. We use pretrained word2vec vectors trained on the same corpus to initialize the embeddings, but we finetuned them by backpropagating through the embeddings during training. Our vocabulary sizes are fixed to 125K for source and 75K for target for both exper-

[1] https://github.com/facebook/NAMAS

iments.

We use the reference data for pointers for the model only at the training time. During the test time, the switch makes a decision at every timestep on which softmax layer to use.

For evaluation, we use full-length Rouge F1 using the official evaluation tool [2]. In their work, the authors of (Bahdanau et al., 2014) use full-length Rouge Recall on this corpus, since the maximum length of limited-length version of Rouge recall of 75 bytes (intended for DUC data) is already long for Gigaword summaries. However, since full-length Recall can unfairly reward longer summaries, we also use full-length F1 in our experiments for a fair comparison between our models, independent of the summary length.

The experimental results comparing the Pointer Softmax with NMT model are displayed in Table 1 for the UNK pointers data and in Table 2 for the entity pointers data. As the experiments show, pointer softmax improves over the baseline NMT on both UNK data and entities data. Our hope was that the improvement would be larger for the entities data since the incidence of pointers was much greater. However, it turns out this is not the case, and we suspect the main reason is anonymization of entities which removed data-sparsity by converting all entities to integer-ids that are shared across all documents. We believe that on de-anonymized data, our model could help more, since the issue of data-sparsity is more acute in this case.

Table 1: Results on Gigaword Corpus when pointers are used for UNKs in the training data, using Rouge-F1 as the evaluation metric.

	Rouge-1	Rouge-2	Rouge-L
NMT + lvt	34.87	16.54	32.27
NMT + lvt + PS	**35.19**	**16.66**	**32.51**

Table 2: Results on anonymized Gigaword Corpus when pointers are used for entities, using Rouge-F1 as the evaluation metric.

	Rouge-1	Rouge-2	Rouge-L
NMT + lvt	34.89	**16.78**	32.37
NMT + lvt + PS	**35.11**	16.76	**32.55**

[2] http://www.berouge.com/Pages/default.aspx

Table 3: Results on Gigaword Corpus for modeling UNK's with pointers in terms of recall.

	Rouge-1	Rouge-2	Rouge-L
NMT + lvt	36.45	17.41	33.90
NMT + lvt + PS	**37.29**	**17.75**	**34.70**

In Table 3, we provide the results for summarization on Gigaword corpus in terms of recall as also similar comparison done by (Rush et al., 2015). We observe improvements on all the scores with the addition of pointer softmax. Let us note that, since the test set of (Rush et al., 2015) is not publicly available, we sample 2000 texts with their summaries without replacement from the validation set and used those examples as our test set.

In Table 4 we present a few system generated summaries from the Pointer Softmax model trained on the UNK pointers data. From those examples, it is apparent that the model has learned to accurately point to the source positions whenever it needs to generate rare words in the summary.

5.3 Neural Machine Translation

In our neural machine translation (NMT) experiments, we train NMT models with attention over the Europarl corpus (Bahdanau et al., 2014) over the sequences of length up to 50 for English to French translation. [3]. All models are trained with early-stopping which is done based on the negative log-likelihood (NLL) on the development set. Our evaluations to report the performance of our models are done on `newstest2011` by using BLUE score. [4]

We use $30,000$ tokens for both the source and the target language shortlist vocabularies (1 of the token is still reserved for the unknown words). The whole corpus contains $134,831$ unique English words and $153,083$ unique French words. We have created a word-level dictionary from French to English which contains translation of 15,953 words that are neither in shortlist vocabulary nor dictionary of common words for both the source and the target. There are about $49,490$ words shared between English and French parallel corpora of Europarl.

[3] In our experiments, we use an existing code, provided in https://github.com/kyunghyuncho/dl4mt-material, and on the original model we only changed the last softmax layer for our experiments

[4] We compute the BLEU score using the multi-blue.perl script from Moses on tokenized sentence pairs.

Table 4: Generated summaries from NMT with PS. Boldface words are the words copied from the source.

Source #1	china 's tang **gonghong** set a world record with a clean and jerk lift of ### kilograms to win the women 's over-## kilogram weightlifting title at the asian games on tuesday .
Target #1	china 's tang <unk>,sets world weightlifting record
NMT+PS #1	china 's tang **gonghong** wins women 's weightlifting weightlifting title at asian games
Source #2	owing to criticism , nbc said on wednesday that it was ending a **three-month-old** experiment that would have brought the first liquor advertisements onto national broadcast network television .
Target #2	advertising : nbc retreats from liquor commercials
NMT+PS #2	nbc says it is ending a **three-month-old** experiment
Source #3	a senior trade union official here wednesday called on ghana 's government to be " **mindful** of the plight " of the ordinary people in the country in its decisions on tax increases .
Target #3	tuc official,on behalf of ordinary ghanaians
NMT+PS #3	ghana 's government urged to be **mindful** of the plight

During the training, in order to decide whether to pick a word from the source sentence using attention/pointers or to predict the word from the short-list vocabulary, we use a simple heuristic. If the word is not in the short-list vocabulary, we first check if the word $\mathbf{y}_t$ itself appears in the source sentence. If it is not, we check if the word itself is in the source sentence by using the shared words lookup table for the source and the target language. If the word is in the source sentence, we then use the location of the word in the source as the target. Otherwise we check if one of the English senses from the cross-language dictionary of the French word is in the source. If it is in the source sentence, then we use the location of that word as our translation. Otherwise we just use the argmax of $\mathbf{l}_t$ as the target.

For switching network d_t, we observed that using a two-layered MLP with noisy-tanh activation (Gulcehre et al., 2016) function with residual connection from the lower layer (He et al., 2015) activation function to the upper hidden layers improves the BLEU score about 1 points over the d_t using ReLU activation function. We initialized the biases of the last sigmoid layer of d_t to -1 such that if d_t becomes more biased toward choosing the shortlist vocabulary at the beginning of the training. We renormalize the gradients if the norm of the gradients exceed 1 (Pascanu et al., 2012).

In Table 5, we provided the result of NMT with pointer softmax and we observe about 3.6 BLEU score improvement over our baseline.

Table 5: Europarl Dataset (EN-FR)

	BLEU-4
NMT	20.19
NMT + PS	**23.76**

In Figure 4, we show the validation curves of the NMT model with attention and the NMT model with shortlist-softmax layer. Pointer softmax converges faster in terms of number of minibatch updates and achieves a lower validation negative-log-likelihood (NLL) (63.91) after $200k$ updates over the Europarl dataset than the NMT model with shortlist softmax trained for $400k$ minibatch updates (65.26). Pointer softmax converges faster than the model using the shortlist softmax, because the targets provided to the pointer softmax also acts like guiding hints to the attention.

6 Conclusion

In this paper, we propose a simple extension to the traditional soft attention-based shortlist softmax by using pointers over the input sequence. We show that the whole model can be trained jointly with single objective function. We observe noticeable improvements over the baselines on machine translation and summarization tasks by using pointer softmax. By doing a very simple mod-

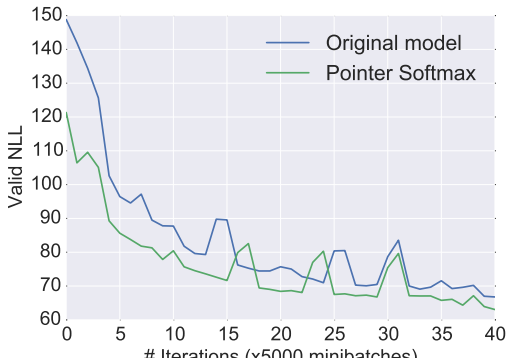


Figure 4: A comparison of the validation learning-curves of the same NMT model trained with pointer softmax and the regular softmax layer. As can be seen from the figures, the model trained with pointer softmax converges faster than the regular softmax layer. Switching network for pointer softmax in this Figure uses ReLU activation function.

ification over the NMT, our model is able to generalize to the unseen words and can deal with rare-words more efficiently. For the summarization task on Gigaword dataset, the pointer softmax was able to improve the results even when it is used together with the large-vocabulary trick. In the case of neural machine translation, we observed that the training with the pointer softmax is also improved the convergence speed of the model as well. For French to English machine translation on Europarl corpora, we observe that using the pointer softmax can also improve the training convergence of the model.

References

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[Bengio and Senécal2008] Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *Neural Networks, IEEE Transactions on*, 19(4):713–722.

[Bordes et al.2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

[Cheng and Lapata2016] Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

[Cho et al.2014] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Chung et al.2014] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

[Gillick et al.2015] Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

[Graves2013] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

[Gu et al.2016] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

[Gulcehre et al.2016] Caglar Gulcehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. 2016. Noisy activation functions. *arXiv preprint arXiv:1603.00391*.

[Gutmann and Hyvärinen2012] Michael U Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361.

[He et al.2015] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for mage recognition. *arXiv preprint arXiv:1512.03385*.

[Hermann et al.2015] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

[Jean et al.2014] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

[Kingma and Adam2015] Diederik P Kingma and Jimmy Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representation*.

[Luong et al.2015] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*.

[Matthews et al.2012] Danielle Matthews, Tanya Behne, Elena Lieven, and Michael Tomasello. 2012. Origins of the human pointing gesture: a training study. *Developmental science*, 15(6):817–829.

[Mnih and Kavukcuoglu2013] Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems*, pages 2265–2273.

[Morin and Bengio2005] Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Aistats*, volume 5, pages 246–252. Citeseer.

[Pascanu et al.2012] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

[Pascanu et al.2013] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

[Rush et al.2015] Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *CoRR*, abs/1509.00685.

[Schuster and Paliwal1997] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681.

[Sennrich et al.2015] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

[Theano Development Team2016] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

[Tomasello et al.2007] Michael Tomasello, Malinda Carpenter, and Ulf Liszkowski. 2007. A new look at infant pointing. *Child development*, 78(3):705–722.

[Vinyals et al.2015] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

7 Acknowledgments

We would also like to thank the developers of Theano [5], for developing such a powerful tool for scientific computing (Theano Development Team, 2016). We acknowledge the support of the following organizations for research funding and computing support: NSERC, Samsung, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. C. G. thanks for IBM T.J. Watson Research for funding this research during his internship between October 2015 and January 2016.

[5] `http://deeplearning.net/software/theano/`

Generalized Transition-based Dependency Parsing via Control Parameters

Bernd Bohnet, Ryan McDonald, Emily Pitler and Ji Ma
Google Inc.
{bohnetbd,ryanmcd,epitler,maji}@google.com

Abstract

In this paper, we present a generalized transition-based parsing framework where parsers are instantiated in terms of a set of control parameters that constrain transitions between parser states. This generalization provides a unified framework to describe and compare various transition-based parsing approaches from both a theoretical and empirical perspective. This includes well-known transition systems, but also previously unstudied systems.

1 Introduction

Transition-based dependency parsing is perhaps the most successful parsing framework in use today (Nivre, 2008). This is due to the fact that it can process sentences in linear time (Nivre, 2003); is highly accurate (Zhang and Nivre, 2011; Bohnet and Nivre, 2012; Weiss et al., 2015); and has elegant mechanisms for parsing non-projective sentences (Nivre, 2009). As a result, there have been numerous studies into different transition systems, each with varying properties and complexities (Nivre, 2003; Attardi, 2006; Nivre, 2008; Nivre, 2009; Gómez-Rodríguez and Nivre, 2010; Choi and Palmer, 2011; Pitler and McDonald, 2015).

While connections between these transition systems have been noted, there has been little work on developing frameworks that generalize the phenomena parsed by these diverse systems. Such a framework would be beneficial for many reasons: It would provide a language from which we can theoretically compare known transition systems; it can give rise to new systems that could have favorable empirical properties; and an implementation of the generalization allows for comprehensive empirical studies.

In this work we provide such a generalized transition-based parsing framework. Our framework can be cast as transition-based parsing as it contains both parser states as well as transitions between these states that construct dependency trees. As in traditional transition-based parsing, the state maintains two data structures: a set of *unprocessed tokens* (normally called the *buffer*); and a set of *operative tokens* (often called the *stack*). Key to our generalization is the notion of *active* tokens, which is the set of tokens in which new arcs can be created and/or removed from consideration. A parser instantiation is defined by a set of control parameters, which dictate: the types of transitions that are permitted and their properties; the capacity of the active token set; and the maximum arc distance.

We show that a number of different transition systems can be described via this framework. Critically the two most common systems are covered – arc-eager and arc-standard (Nivre, 2008). But also Attardi's non-projective (Attardi, 2006), Kuhlmann's hybrid system (Kuhlmann et al., 2011), the directed acyclic graph (DAG) parser of Sagae and Tsujii (2008), and likely others. More interestingly, the easy-first framework of Goldberg and Elhadad (2010) can be described as an arc-standard system with an unbounded active token capacity.

We present a number of experiments with an implementation of our generalized framework. One major advantage of our generalization (and its implementation) is that it allows for easy exploration of novel systems not previously studied. In Section 5 we discuss some possibilities and provide experiments for these in Section 6.

2 Related Work

Transition-based dependency parsing can be characterized as any parsing system that maintains a

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 150–160,
Berlin, Germany, August 7-12, 2016.

state as well as a finite set of operations that move the system from one state to another (Nivre, 2008). In terms of modern statistical models that dominate the discourse today, the starting point is likely the work of Kudo and Matsumoto (2000) and Yamada and Matsumoto (2003), who adopted the idea of cascaded chunking from Abney (1991) in a greedy dependency parsing framework.

From this early work, transition-based parsing quickly grew in scope with the formalization of the arc-eager versus arc-standard paradigms (Nivre, 2003; Nivre, 2008), the latter largely being based on well-known shift-reduce principles in the phrase-structure literature (Ratnaparkhi, 1999). The speed and empirical accuracy of these systems – as evident in the widely used MaltParser software (Nivre et al., 2006a) – led to the study of a number of different transition systems.

Many of these new transition systems attempted to handle phenomena not covered by arc-eager or arc-standard transition systems, which inherently could only produce projective dependency trees. The work of Attardi (2006), Nivre (2009), Gómez-Rodríguez and Nivre (2010), Choi and Palmer (2011), and Pitler and McDonald (2015) derived transition systems that could parse non-projective trees. Each of these systems traded-off complexity for empirical coverage. Additionally, Sagae and Tsujii (2008) developed transition systems that could parse DAGs by augmentating the arc-standard and the arc-eager system. Bohnet and Nivre (2012) derived a system that could produce both labeled dependency trees as well as part-of-speech tags in a joint transition system. Taking this idea further Hatori et al. (2012) defined a transition system that performed joint segmentation, tagging and parsing.

In terms of empirical accuracy, from the early success of Nivre and colleagues (Nivre et al., 2006b; Hall et al., 2007; Nivre, 2008), there has been an succession of improvements in training and decoding, including structured training with beam search (Zhang and Clark, 2008; Zhang and Nivre, 2011), incorporating graph-based rescoring features (Bohnet and Kuhn, 2012), the aformentioned work on joint parsing and tagging (Bohnet and Nivre, 2012), and more recently the adoption of neural networks and feature embeddings (Chen and Manning, 2014; Weiss et al., 2015; Dyer et al., 2015; Alberti et al., 2015).

In terms of abstract generalizations of transition systems, the most relevant work is that of Gómez-Rodríguez and Nivre (2013) – which we abbreviate GR&N13. In that work, a generalized framework is defined by first defining a set of base transitions, and then showing that many transition-based systems can be constructed via composition of these base transitions. Like our framework, this covers common systems such as arc-eager and arc-standard, as well as easy-first parsing. In particular, arc construction in easy-first parsing can be seen as an action composed of a number of shifts, an arc action, and a number of un-shift actions. The primary conceptual difference between that work and the present study is the distinction between complex actions versus control parameters. In terms of theoretical coverage, the frameworks are not equivalent. For instance, our generalization covers the system of Attardi (2006), whereas GR&N13 cover transition systems where multiple arcs can be created in tandem. In Section 7 we compare the two generalizations.

3 Generalized Transition-based Parsing

A transition system must define a parser state as well as a set of transitions that move the system from one state to the next. Correct sequences of transitions create valid parse trees. A parser state is typically a tuple of data structures and variables that represent the dependency tree constructed thus far and, implicitly, possible valid transitions to the next state.

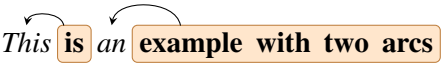
In order to generalize across the parser states of transition-based parsing systems, we preserve their common parts and make the specific parts configurable. In order to generalize across the transitions, we divide the transitions into their basic operations. Each specific transition is then defined by composition of these basic operations. By defining the properties of the data structures and the composition of the basic operations, different transition systems can be defined and configured within one single unified system. As a consequence, we obtain a generalized parser that is capable of executing a wide range of different transition systems by setting a number of control parameters without changing the specific implementation of the generalized parser.

3.1 Basic Notation

In the following, we use a directed unlabeled dependency tree $T = \langle V, A \rangle$ for a sentence $x =$

(a) Arc-standard: *is* and *example* are eligible for arcs.

(b) Arc-eager: *example* and *with* are eligible for arcs.



(c) Easy-first: All unreduced tokens are active (bolded).

Figure 1: A partially processed dependency tree after having just added the arc *(example, an)* in the arc-standard, arc-eager, and easy-first systems. Tokens in the operative token set O are shaded orange, while tokens in the unordered buffer U are in an unshaded box. The bolded tokens are in ACTIVE(O) and eligible for arcs (Section 3.4).

$w_1, ..., w_n$, where $V_x = \{1, ..., n\}$ and $V_x^r = V_x \cup \{r\}$, $A_x \subset V_x^r \times V_x$. r is a placeholder for the root node and set either to 0 (root to the left of the sentence) or $n + 1$ (root to the right). The definition is mostly equivalent to that of Kübler et al. (2009) but deviates in the potential handling of the root on the right (Ballesteros and Nivre, 2013).

The set of nodes V_x index the words of a sentence x and V_x^r includes in addition the artificial root node r. Let A be the arc set, i.e., $(i, j) \in A$ iff there is a dependency from i to j. We use as alternative notation $(i \rightarrow j)$. This is the arc set the algorithm will create.

For ease of exposition, we will only address unlabeled parsing. However, for our experiments we do implement a labeled parsing variant using the standard convention of composing arc transitions with corresponding arc labels.

3.2 Generalized Parser State

Let U be an unordered set of buffered unprocessed tokens. This set is identical to the buffer from transition-based parsing. Following standard notation, we will use $i|U$ to indicate that i is the leftmost element of the set.

Let O be an ordered set of operative tokens. Specifically, O is the set of tokens that 1) have been moved out of U, and 2) are not themselves reduced. The set O is similar in nature to the traditional stack of transition-based parsing, but is not restricted to stack operations. As in transition-based parsing we will use the notation $O|i$ to indicate that i is the rightmost element of the set; $O[n]$ is the n-th rightmost element of the set. Figure 1 shows the set of tokens in O within shaded boxes and U within unshaded boxes.

Transitions

"Adds an arc from j to i, both $\in$ ACTIVE(O)."	
$\leftarrow_{i,j}$	$(O, U, A) \Rightarrow (O, U, A \cup (j \rightarrow i))$
"Adds an arc from i to j, both $\in$ ACTIVE(O)."	
$\rightarrow_{i,j}$	$(O, U, A) \Rightarrow (O, U, A \cup (i \rightarrow j))$
"Removes token $i \in$ ACTIVE(O) from O."	
$-_i$	$(O...\text{i}..., U, A) \Rightarrow (O, U, A)$
"Moves the top token from U to the top of O."	
$+$	$(O, i\|U, A) \Rightarrow (O\|i, U, A)$

Figure 2: Base generalized transitions over parser states.

3.3 Generalized Transitions

The above discussion describes what a generalized transition-based parser state looks like; it does not describe any transitions between these states, which is the core of transition-based parsing. In this section we present a set of basic transitions, which themselves can be composed to make more complex transitions (similar to Gómez-Rodríguez and Nivre (2013)).

Let $\mathcal{T} = \{\leftarrow_{i,j}, \rightarrow_{i,j}, -_i, +\}$ be the set of basic transitions (Figure 2), which have analogues to standard transition-based parsing. These transitions come with the standard preconditions, i.e., a root cannot be a modifier; each token can modify at most one word[1]; a token can only be reduced if it has a head; and a shift can only happen if the unoperative buffer is non-empty. We will often refer to these as LEFT-ARC ($\leftarrow_{i,j}$), RIGHT-ARC ($\rightarrow_{i,j}$), REDUCE ($-_i$), and SHIFT ($+$). We additionally refer to LEFT-ARC and RIGHT-ARC together as *arc-creation* actions.

3.4 Control Parameters

Instantiations of a transition system are defined via *control parameters*. We defined two sets of such parameters. The first, we call *global parameters*, dictates system wide behaviour. The second, we call *transition parameters*, dictates a specific behaviour for each transition.

[1]This precondition is not needed for DAG transition systems.

Global Control Parameters

We have two parameters for the broader behaviour of the system.

1. Active token capacity K. The active set of tokens ACTIVE(O) that can be operated on by the transitions is the set $\{O[min(|O|, K)], ..., O[1]\}$. K additionally determines the size of O at the start of parsing. E.g., if $K = 2$, then we populate O with the first two tokens. This is equivalent to making SHIFT deterministic while $|O| < K$.

2. Max arc distance D. I.e., arcs can only be created between two active tokens $O[i]$ and $O[j]$ if $|i - j| \leq D$.

Transition Control Paramters

Let $\mathcal{M}(\mathcal{T})$ be a *multiset* of transitions, such that if $t \in \mathcal{M}(\mathcal{T})$, then $t \in \mathcal{T}$. Note that $\mathcal{M}(\mathcal{T})$ is a multiset, and thus can have multiple transitions of the same type. For each $t \in \mathcal{M}(\mathcal{T})$, our generalization requires the following control parameters to be set (default in bold):

1. Bottom-up: $B \in \{[\mathbf{t}]\mathbf{rue}, [f]alse\}$. Whether creating an arc also reduces it. Specifically we will have two parameters, B_L and B_R, which specify whether LEFT/RIGHT-ARC actions are bottom up. We use the notation and say $B = true$ to mean $B_L = B_R = true$. For example $B_L = true$ indicates that $\leftarrow_{i,j}$ is immediately followed by a reduce $-_i$.

2. Arc-Shift: $S \in \{[t]rue, [\mathbf{f}]\mathbf{alse}\}$. Whether creating an arc also results in SHIFT. Specifically we will have two parameters, S_L and S_R, which specify whether LEFT/RIGHT-ARC actions are joined with a SHIFT. We use the notation and say $S = true$ to mean $S_L = S_R = true$. For example $S_L = true$ indicates that $\leftarrow_{i,j}$ is immediately followed by $+$.

3. Periphery: $P \in \{[l]eft, [r]ight, \mathbf{na}\}$. If a transition must operate on the left or right periphery of the active token set ACTIVE(O). For arc-creation transitions, this means that at least one of the head or modifier is on the specified periphery. If the value is na, that means that the action is not constrained to be on the periphery. Note, that when K ≤ 2, all arc-creation actions by default are on the periphery.

Each of these control parameters has a default value, which will be assumed if unspecified. Note that the relevance of these parameters is transition dependent. E.g., a SHIFT requires no such control parameters and a REDUCE needs neither B nor S.

These control parameters allow *limited* compositionality of the basic transitions in Figure 2. Unlike Gómez-Rodríguez and Nivre (2013), each transition includes at most one SHIFT, at most one REDUCE, and at most one LEFT-ARC or RIGHT-ARC. I.e., the most compositional transition is a LEFT/RIGHT-ARC with a REDUCE and/or a SHIFT. Even with this restriction, all of the transition systems covered by Gómez-Rodríguez and Nivre (2013) can still be expressed in our generalization.

3.5 Generalized Transition System

To summarize, a generalized transition is defined as follows:

1. A parser state: $\Gamma = \langle O, U, A \rangle$.

2. A set of basic transitions: $\mathcal{T} = \{\leftarrow_{i,j}, \rightarrow_{i,j}, -_i, +\}$.

And each transition system instantiation must further define:

1. Values for global control parameters: K and D.

2. A multiset of valid transitions $\mathcal{M}(\mathcal{T})$, where $\forall t \in \mathcal{M}(\mathcal{T})$, then $t \in \mathcal{T}$.

3. For each $t \in \mathcal{M}(\mathcal{T})$ values for B, S, and P.

4 Transition System Instantiations

The instantiation of the transition system consists of setting the capacity K and distance D as well as the transition control parameters. In order to make the comparison clearer, we will define typical transition systems using the notation of Kuhlmann et al. (2011). Here, a parser state is a triple (σ, β, A), where $\sigma|i$ is a stack with top element i, $j|\beta$ is a buffer whose next element is j, and A the set of created arcs. To make notation cleaner, we will drop indexes whenever the context makes it clear. E.g., if the active token capacity is 2 ($K = 2$), then necessarily for $\leftarrow_{i,j}$, $i = 2$ and $j = 1$ and we can write $\leftarrow$. When $K > 2$ or $D > 1$, a base arc-creation action can be instantiated into multiple transitions that only differ by the indexes. E.g., when $K = 3$ and $D = 2$, $\leftarrow$ with $P = na$ can have three instantiations: $\leftarrow_{3,2}$, $\leftarrow_{2,1}$ and

$\leftarrow_{3,1}$. To keep exposition compact, in such circumstances we use $\star$ to denote the set of index-pairs allowed by the given K, D and P values.

4.1 Arc-standard

Arc-standard parsing is a form of shift-reduce parsing where arc-creations actions happen between the top two elements on the stack:

LEFT-ARC: $(\sigma|i|j, \beta, A) \Rightarrow (\sigma|j, \beta, A \cup (j \rightarrow i))$

RIGHT-ARC: $(\sigma|i|j, \beta, A) \Rightarrow (\sigma|i, \beta, A \cup (i \rightarrow j))$

SHIFT: $(\sigma, i|\beta, A) \Rightarrow (\sigma|i, \beta, A)$

This is easily handled in our generalization by having only two active tokens. $\leftarrow$ and $\rightarrow$ actions with default parameter values are used to simulate LEFT-ARC and RIGHT-ARC respectively, and $+$ is used to shift tokens from U to O.

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow$	$\{$*default*$\}$
RIGHT-ARC	$\rightarrow$	$\{$*default*$\}$
SHIFT	$+$	
capacity K		2
arc distance D		1

Note that when $K = 2$, arc-creations are by definition always on the periphery.

4.2 Arc-eager

Arc-eager transition systems have been described in various ways. Kuhlmann et al. (2011) defines it as operations between tokens at the top of the stack and front of the buffer.

LEFT-ARC: $(\sigma|i, j|\beta, A) \Rightarrow (\sigma, j|\beta, A \cup (j \rightarrow i))$

RIGHT-ARC: $(\sigma|i, j|\beta, A) \Rightarrow (\sigma|i|j, \beta, A \cup (i \rightarrow j))$

REDUCE: $(\sigma|i, \beta, A) \Rightarrow (\sigma, \beta, A)$

SHIFT: $(\sigma, i|\beta, A) \Rightarrow (\sigma|i, \beta, A)$

In our generalization, we can simulate this by having two active tokens on the operative set, representing the top of the stack and front of the buffer. SHIFT and LEFT-ARC are handled in the same way as in the arc-standard system. The RIGHT-ARC action is simulated by $\rightarrow$ with modified parameters to account for the fact that the action keeps the modifier in O and also shifts a token from U to O. The REDUCE action is handled by $-$ with the periphery set to left so as to remove the second rightmost token from O.

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow$	$\{$*default*$\}$
RIGHT-ARC	$\rightarrow$	$\{B = f, S = t\}$
REDUCE	$-$	$\{P = l\}$
SHIFT	$+$	
K		2
D		1

Note that the artificial root must be on the right of the sentence to permit the reduce to operate at the left periphery of the active token set.

4.3 Easy-first

For easy-first parsing (Goldberg and Elhadad, 2010), the number of active tokens is infinite or, more precisely, equals to the number of tokens in the input sentence, and arc-creation actions can happen between any two adjacent tokens.

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow_\star$	$\{$*default*$\}$
RIGHT-ARC	$\rightarrow_\star$	$\{$*default*$\}$
K		∞
D		1

Note here $\star$ denotes the set of indexes $\{(i, j)|i \leq |O|, j \geq 1, i = j + 1\}$. Thus, at each time step there are $2 * |O|$ actions that need to be considered. Additionally, reduce is always composed with arc-creation and since $K = \infty$, then there is never a SHIFT operation as O is immediately populated with all tokens on the start of parsing.

4.4 Kuhlmann et al. (2011)

Kuhlmann et al. present a 'hybrid' transition system where the RIGHT-ARC action is arc-standard in nature, but LEFT-ARC actions is arc-eager in nature, which is equivalent to the system of Yamada and Matsumoto (2003). We can get the same effect as their system by allowing three active tokens, representing the top two tokens of the stack and the front of the buffer. Transitions can be handled similarly as the arc-standard system where only the periphery parameter need to be changed accordingly. This change also requires the root to be on the right of the sentence.

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow$	$\{P = r\}$
RIGHT-ARC	$\rightarrow$	$\{P = l\}$
SHIFT	$+$	
K		3
D		1

4.5 Sagae and Tsujii (2008)

Sagae and Tsuji present a model for projective DAG parsing by modifying the LEFT-ARC and RIGHT-ARC transitions of the arc-eager system. In their system, the LEFT-ARC transition does not remove the dependent, and RIGHT-ARC transition does not shift any token from the input to the stack.

LEFT-ARC: $(\sigma|i,j|\beta,A) \Rightarrow (\sigma|i,j|\beta,A \cup (j \rightarrow i))$

RIGHT-ARC: $(\sigma|i,j|\beta,A) \Rightarrow (\sigma|i,j|\beta,A \cup (i \rightarrow j))$

REDUCE: $(\sigma|i,\beta,A) \Rightarrow (\sigma,\beta,A)$

SHIFT: $(\sigma,i|\beta,A) \Rightarrow (\sigma|i,\beta,A)$

This can be easily simulated by modifying arc-eager system mentioned in Sec. 4.2 such that both the LEFT-ARC and RIGHT-ARC transition keep the stack/buffer untouched.

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow$	$\{B = f\}$
RIGHT-ARC	$\rightarrow$	$\{B = f\}$
REDUCE	$-$	$\{P = l\}$
SHIFT	$+$	
K		2
D		1

4.6 Attardi (2006)

Now we show how our framework can extend to non-projective systems. This is primarily controlled by the arc-distance parameter D.

The base of the Attardi non-projective transition system is the arc-standard system. Attardi modifies RIGHT/LEFT-ARC actions that operate over larger contexts in the stack. For simplicity of exposition below we model the variant of the Attardi system described in Cohen et al. (2011).

RIGHT-ARCN: $(\sigma|i_{1+N}|\ldots|i_2|i_1,\beta,A) \Rightarrow (\sigma|i_N|\ldots|i_2|i_1,\beta,A \cup (i_1 \rightarrow i_{1+N}))$

LEFT-ARCN: $(\sigma|i_{1+N}|\ldots|i_2|i_1,\beta,A) \Rightarrow (\sigma|i_{1+N}|\ldots|i_2,\beta,A \cup (i_{1+N} \rightarrow i_1))$

SHIFT: $(\sigma,i|\beta,A) \Rightarrow (\sigma|i,\beta,A)$

Conceptually the Attardi system can be generalized to any value of N[2], and Attardi specifically allows N=1,2,3. Actions that create arcs between non-adjacent tokens permit limited non-projectivity.

Thus, for Attardi, we set the number of active tokens to N+1, to simulate the top N+1 tokens of the stack, and set the max distance D to N to indicate that tokens up to N positions below the top of the stack can add arcs with the top of the stack.

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow_\star$	$\{P = r\}$
RIGHT-ARC	$\rightarrow_\star$	$\{P = r\}$
K		$N + 1$
D		N (3 for Attardi (2006))

The critical parameter for each arc action is that $P = r$. This means that the right peripheral active token always must participate in the action, as does the right-most token of the stack for the original Attardi. Here $\star$ denotes $\{(i,j)|j = 1, i - j \leq D\}$.

5 Novel Transition Systems

Any valid setting of the control parameters could theoretically define a new transition system, however not all such combinations will be empirically reasonable. We outline two potential novel transition systems suggested by our framework, which we will experiment with in Section 6. This is a key advantage of our framework (and implementation) – it provides an easy experimental solution to explore novel transition systems.

5.1 Bounded Capacity Easy-first

Easy-first and arc-standard are similar since they are both bottom-up and both create arcs between adjacent nodes. The main difference lies in the capacity K, which is 2 for arc-standard and ∞ for easy-first. In addition, the shift action is needed by the arc-standard system. Each system has some advantages: arc-standard is faster, and somewhat easier to train, while easy-first can be more accurate (under identical learning settings). Seen this way, it is natural to ask: what happens if the active token range K is set to k, with $2 < k < \infty$? We explore various values in the region between arc-standard and easy-first in Section 6.

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow_\star$	$\{default\}$
RIGHT-ARC	$\rightarrow_\star$	$\{default\}$
SHIFT	$+$	
K		k
D		1

5.2 Non-projective Easy-first

A simple observation is that by allowing $D > 1$ makes any transition system naturally non-

[2] Attardi (2006) also introduces Extract/Insert actions to a temporary buffer that he argues generalizes to all values of N. We don't account for that specific generalization here.

projective. One example would be a non-projective variant of easy-first parsing:

$\mathcal{M}(\mathcal{T})$	base	parameter values
LEFT-ARC	$\leftarrow_\star$	$\{default\}$
RIGHT-ARC	$\rightarrow_\star$	$\{default\}$
K		∞
D		any value of N

Here N denotes the maximum arc-creation distance.

We also note that one could potentially vary both K and D simultaneously, giving a non-projective limited capacity easy-first system.

6 Implementation and Experiments

Our implementation uses a linear model $\sum_{y_i \in y} \mathbf{w} \cdot \mathbf{f}(x, y_1, \ldots, y_i)$ to assign a score to a sequence $y = y_1, y_2, \ldots y_m$ of parser transitions, given sentence x. Model parameters are trained using the structured perceptron with "early update" (Collins and Roark, 2004) and features follow that of Zhang and Nivre (2011).

For the arc-standard and arc-eager transition systems, we use the static oracle to derive a single gold sequence for a given sentence and its gold tree. For systems where there is no such static oracle, for example the easy-first system, we use the method proposed by Ma et al. (2013) to select a gold sequence such that, for each update, the condition $\mathbf{w} \cdot \mathbf{f}(x, \hat{y}_k) < \mathbf{w} \cdot \mathbf{f}(x, \overline{y}_k)$ always holds, which is required for perceptron convergence. Here $\hat{y}_k$ denotes the length k prefix of a correct sequence and $\overline{y}_k$ denotes the highest scoring sequence in the beam.

We carry out the experiments on the Wall Street Journal using the standard splits for the training set (section 2-21), development set (section 22) and test set (section 23). We converted the constituency trees to Stanford dependencies version 3.3.0 (de Marneffe et al., 2006). We used a CRF-based Part-of-Speech tagger to generate 5-fold jack-knifed Part-of-Speech tag annotation of the training set and used predicted tags on the development and test set. The tagger reaches accuracy scores similar to the Stanford tagger (Toutanova et al., 2003) with 97.44% on the test set. The unlabeled and labeled accuracy scores exclude punctuation marks.

Obviously, there are many interesting instantiations for the generalized transition system. In particular, it would be interesting to investigate parsing performance of systems with different active token size and arc-distance. Before we investigate these system in the next subsections, we present the performance on standard systems.

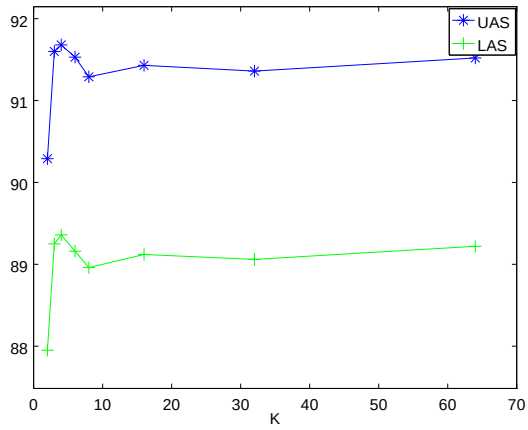


Figure 3: Labeled/unlabeled attachment scores with respect to the active capacity K.

6.1 Common Systems

The results for arc-standard, arc-eager and easy-first (bottom of Table 1) show how standard systems perform within our framework. Easy-first's labeled attachment score (LAS) is 0.46 higher than the LAS of arc-eager when using the same feature set. These results are competitive with the state-of-the-art linear parsers, but below recent work on neural network parsers. A future line of work is to adopt such training into our generalization.

System	UAS	LAS	b
Dyer et al. (2015)	93.10	90.90	–
Weiss et al. (2015)	93.99	92.05	–
Alberti et al. (2015)	94.23	92.23	8
Zhang and Nivre (2011)$\star$	92.92	90.88	32
Zhang and McDonald (2014)	93.22	91.02	–
Arc-standard (gen.)	92.81	90.68	32
Arc-eager (gen.)	92.88	90.73	32
Easy-first (gen.)	93.31	91.19	32

Table 1: State-of-the-art comparison. $\star$ denotes our own re-implementation. The systems in the first block on the top use neural networks.

6.2 Bounded Capacity Easy-first

Table 1 shows that easy-first is more accurate than arc-standard. However, it is also more computationally expensive. By varying the number of active tokens, we can investigate whether there is a sweet spot in the accuracy vs. speed trade-off.

Figure 3 shows labeled/unlabeled accuracy scores on the development set for active token

sizes ranging from 2 to 32, all with beam size 1. Different from the original easy-first system where all tokens are initialized as active tokens, in the bounded capacity system, a token can be active only after it has been shifted from U to O. We observe that the accuracy score increases remarkably by over a point when active token capacity gets increased from 2 to 3, and peaks at a active token capacity of 4. Generally, more active tokens allows the parser to delay "difficult" decisions to later steps and choose the "easy" ones at early steps. Such behavior has the effect of limiting the extent of error propagation. The result also suggests that a modification as simple as adding one more active token to the arc-standard system can yield significant improvement.

With a larger active token capacity, we see a slight drop of accuracy. This is likely related to the parser having to predict when to perform a shift transition. In comparison, the vanilla easy-first parser does not need to model this.

6.3 Non-projective Easy-first

For the experiments with non-projective easy-first, we use the Dutch and Danish CoNLL 2006 corpora. To assess the performance, we applied the evaluation rules of the 2006 shared task. In order to make the non-projective systems perform well, we added to all feature templates the arc-distance D. In these experiments, we included in the training an artificial root node on the right since Dutch as well a few sentences of the Danish corpus have more than one root node.

In the experiments, we use the easy-first setting with infinite set of active tokens K and increase stepwise the arc-distance D. For training, we filter out the sentences which contain non-projective arcs not parseable with the selected setting. Table 2 provides an overview of the performance with increasing arc-distance. At the bottom of the table, we added accuracy scores for the bounded capacity non-projective easy-first parser since we think these settings provide attractive trade-offs between accuracy and complexity.

The original easy-first performs $O(n)$ feature extractions and has a runtime of $O(n\log(n))$ assuming a heap is used to extract the argmax at each step and feature extraction is done over local contexts only (Goldberg and Elhadad, 2010). For the non-projective variant of easy-first with $D = d$, $O(dn)$ feature extractions are required. Thus, for the unrestricted variant where $K = D = \infty$, $O(n^2)$ feature extractions are required. Table 2 explored more practical settings: when $K = \infty$, $D \leq 7$, the number of feature extractions is back to $O(n)$ with a runtime of $O(n\log(n))$, matching the original easy-first complexity. When both K and D are small constants as in the lower portion of Table 2, the runtime is $O(n)$.

		Danish				Dutch			
D	K	UAS	LAS	CUR	NPS	UAS	LAS	CUR	NPS
1	∞	90.22	85.51	41.30	84.37	79.11	75.41	32.45	63.55
2	∞	90.28	85.85	59.78	96.91	84.73	81.01	70.59	92.44
3	∞	90.68	86.07	65.22	98.82	85.03	81.65	**77.99**	99.01
4	∞	90.58	85.53	69.57	99.69	85.99	82.73	76.85	99.89
5	∞	90.84	86.11	65.22	99.88	85.21	81.93	76.09	99.96
6	∞	90.78	**86.31**	68.48	99.94	84.57	81.13	75.90	**100.0**
7	∞	90.64	85.91	63.04	**100.0**	85.07	82.01	77.04	**100.0**
4	5	90.74	85.87	66.30	99.69	**86.51**	**82.91**	76.66	99.89
5	6	**91.00**	86.21	**72.83**	99.88	86.03	82.73	76.09	99.96

Table 2: Experiments with non-projective easy-first and bounded capacity easy-first with D the arc-distance, K the active token capacity (∞ = all tokens of a sentence), UAS and LAS are the unlabeled and labeled accuracy scores, CUR is the recall of crossing edges and NPS shows the percentage of sentences covered in the training set where 100% means all non-projective (and projective) sentences in the training can be parsed and are included in training.

7 Comparison with GR&N13

The work most similar to ours is that of Gómez-Rodríguez and Nivre (2013) (GR&N13). They define a *divisible transition system* with the principle to divide the transitions into elementary transitions and then to compose from these elementary transitions complex transitions. GR&N13 identified the following five elementary transitions:

SHIFT: $(\sigma, i|\beta, A) \Rightarrow (\sigma|i, \beta, A)$

UNSHIFT: $(\sigma|i, \beta, A) \Rightarrow (\sigma, i|\beta, A)$

REDUCE: $(\sigma|i, \beta, A) \Rightarrow (\sigma, \beta, A)$

LEFT-ARC: $(\sigma|i, j|\beta, A) \Rightarrow (\sigma|i, j|\beta, A \cup (j \rightarrow i))$

RIGHT-ARC: $(\sigma|i, j|\beta, A) \Rightarrow (\sigma|i, j|\beta, A \cup (i \rightarrow j))$

The notion of function composition is used to combine the elementary transitions to the complex ones. For instance, arc-standard would have three actions: SHIFT; RIGHT-ARC $\oplus$REDUCE; LEFT-ARC $\oplus$REDUCE.

The first difference we can note is the GR&N13 cannot instantiate transition systems that produce non-projective trees. This is surely a superficial

difference, as GR&N13 could easily add transitions with larger arc-distance or even SWAP actions (Nivre, 2009).

However, a less superficial difference is that our generalization uses control parameters to construct instantiations of transition systems, instead of solely via transition composition like GR&N13. Through this, our generalization results in a minimal departure from 'standard' representations of these systems. While this may seem like a notational difference, this is particularly a benefit with respect to implementation, as previous techniques for classification and feature extraction can largely be reused.

For example, in GR&N13, the definition of the easy-first transition system (Goldberg and Elhadad, 2010) is complex, e.g., a RIGHT-ARC at position i requires a compositional transition of i SHIFT actions, a RIGHT-ARC, a SHIFT, a REDUCE, then i UNSHIFT actions. Note, that this means in any implementation of this generalization, the output space for a classifier will be very large. Furthermore, the feature space would ultimately need to take the entire sentence into consideration, considering that all compositional actions are centered on the same state.

In our transition system, on the other hand, easy-first operates almost as it does in its native form, where n LEFT-ARC and n RIGHT-ARC actions are ranked relative to each other. There are only two actions, each instantiated for every location in the state. Thus the output space and feature extraction are quite natural.

This leads to straight-forward implementations allowing for easy experimentation and discovery. Unlike GR&N13, we present empirical results for both known transition systems as well as some novel systems (Section 6).

8 Conclusion

We presented a generalized transition system that is capable of representing and executing a wide range of transition systems within one single implementation. These transition systems include systems such as arc-standard, arc-eager, easy-first.

Transitions can be freely composed of elementary operations. The transition system shows perfect alignment between the elementary operations on one hand and their preconditions and the oracle on the other hand. We adjust the transition system to work on a stack in a uniform way starting at a node on the stack and ending with the top node of the stack. The results produced by this system are more comparable as they can be executed with the same classifier and feature extraction system.

Finally, we would like to highlight two insights that the experiments provide. First, a few more active tokens than two can boost the accuracy level of an arc-standard transition system towards the level of an easy-first transition system. These parsing systems maintain very nicely the linear complexity of the arc-standard transition system while they provide a higher accuracy similar to those of easy-first. Second, non-projective trees can be parsed by allowing a larger arc-distance which is a simple way to allow for non-projective edges.

We think that the transition systems with more active tokens or the combination with edges that span over more words provide very attractive transition systems for possible future parsers.

Acknowledgments

We would like to thank the anonymous reviewers, Slav Petrov and Michael Collins for valuable comments on earlier versions of this manuscript.

References

Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-Based Parsing*, pages 257–278. Kluwer.

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of EMNLP 2015*.

Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 166–170.

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

Bernd Bohnet and Jonas Kuhn. 2012. The best of both worlds – a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chpater of the Association for Computational Linguistics (EACL)*, pages 77–87.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical*

Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 1455–1465.

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

Jinho D Choi and Martha Palmer. 2011. Getting the most out of transition-based dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*.

Shay B Cohen, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Exact inference for generative probabilistic non-projective dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1234–1245.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 112–119.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL 2015*, pages 334–343. Association for Computational Linguistics.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 742–750.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1492–1501.

Carlos Gómez-Rodríguez and Joakim Nivre. 2013. Divisible transition systems and multiplanar dependency parsing. *Computational Linguistics*, 39(4):799–845.

Johan Hall, Jens Nilsson, Joakim Nivre, Gülsen Eryiğit, Beáta Megyesi, Mattias Nilsson, and Markus Saers. 2007. Single malt or blended? A study in multilingual parser optimization. In *Proceedings of the CoNLL Shared Task of EMNLP-CoNLL 2007*, pages 933–939.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2012. Incremental joint approach to word segmentation, pos tagging, and dependency parsing in chinese. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1045–1053.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool.

Taku Kudo and Yuji Matsumoto. 2000. Japanese dependency structure analysis based on support vector machines. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in NLP and Very Large Corpora*, pages 18–25.

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

Ji Ma, Jingbo Zhu, Tong Xiao, and Nan Yang. 2013. Easy-first pos tagging and dependency parsing with beam search. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 110–114, Sofia, Bulgaria, August. Association for Computational Linguistics.

Joakim Nivre, Johan Hall, and Jens Nilsson. 2006a. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 2216–2219.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülsen Eryiğit, and Svetoslav Marinov. 2006b. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL)*, pages 221–225.

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

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

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL-IJCNLP)*, pages 351–359.

Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL HLT)*, pages 662–671.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34:151–175.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency DAG parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 753–760.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL 2015*, pages 323–333.

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

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

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 656–661, Baltimore, Maryland, June. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*.

A Transition-Based System for Joint Lexical and Syntactic Analysis

Matthieu Constant
Université Paris-Est, LIGM (UMR 8049)
Alpage, INRIA, Université Paris Diderot
Paris, France
Matthieu.Constant@u-pem.fr

Joakim Nivre
Uppsala University
Dept. of Linguistics and Philology
Uppsala, Sweden
joakim.nivre@lingfil.uu.se

Abstract

We present a transition-based system that jointly predicts the syntactic structure and lexical units of a sentence by building two structures over the input words: a syntactic dependency tree and a forest of lexical units including multiword expressions (MWEs). This combined representation allows us to capture both the syntactic and semantic structure of MWEs, which in turn enables deeper downstream semantic analysis, especially for semi-compositional MWEs. The proposed system extends the arc-standard transition system for dependency parsing with transitions for building complex lexical units. Experiments on two different data sets show that the approach significantly improves MWE identification accuracy (and sometimes syntactic accuracy) compared to existing joint approaches.

1 Introduction

Multiword expressions (MWEs) are sequences of words that form non-compositional semantic units. Their identification is crucial for semantic analysis, which is traditionally based on the principle of compositionality. For instance, the meaning of *cut the mustard* cannot be compositionally derived from the meaning of its elements and the expression therefore has to be treated as a single unit. Since Sag et al. (2002), MWEs have attracted growing attention in the NLP community.

Identifying MWEs in running text is challenging for several reasons (Baldwin and Kim, 2010; Seretan, 2011; Ramisch, 2015). First, MWEs encompass very diverse linguistic phenomena, such as complex grammatical words (*in spite of*, *because of*), nominal compounds (*light house*), non-canonical prepositional phrases (*above board*), verbal idiomatic expressions (*burn the midnight oil*), light verb constructions (*have a bath*), multiword names (*New York*), and so on. They can also be discontiguous in the sense that the sequence can include intervening elements (*John **pulled** Mary's **leg***). They may also vary in their morphological forms (*hot dog, hot dogs*), in their lexical elements (*lose one's mind/head*), and in their syntactic structure (*he took a step*, *the step he took*).

The semantic processing of MWEs is further complicated by the fact that there exists a continuum between entirely non-compositional expressions (*piece of cake*) and almost free expressions (*traffic light*). Many MWEs are indeed semi-compositional. For example, the compound *white wine* denotes a type of wine, but the color of the wine is not white, so the expression is only partially transparent. In the light verb construction *take a nap*, *nap* keeps its usual meaning but the meaning of the verb *take* is bleached. In addition, the noun can be compositionally modified as in *take a long nap*. Such cases show that MWEs may be decomposable and partially analyzable, which implies the need for predicting their internal structure in order to compute their meaning.

From a syntactic point of view, MWEs often have a regular structure and do not need special syntactic annotation. Some MWEs have an irregular structure, such as *by and large* which on the surface is a coordination of a preposition and an adjective. They are syntactically as well as semantically non-compositional and cannot be represented with standard syntactic structures, as stated in Candito and Constant (2014). Many of these irregular MWEs are complex grammatical words like *because of*, *in spite of* and *in order to* – fixed (grammatical) MWEs in the sense of Sag et al. (2002). In some treebanks, these are annotated using special structures and labels because they can-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 161–171,
Berlin, Germany, August 7-12, 2016.

not be modified or decomposed. We hereafter use the term *fixed MWE* to refer to either fixed or irregular MWEs.

In this paper, we present a novel representation that allows both regular and irregular MWEs to be adequately represented without compromising the syntactic representation. We then show how this representation can be processed using a transition-based system that is a mild extension of a standard dependency parser. This system takes as input a sentence consisting of a sequence of tokens and predicts its syntactic dependency structure as well as its lexical units (including MWEs). The resulting structure combines two factorized substructures: (i) a standard tree representing the syntactic dependencies between the lexical elements of the sentence and (ii) a forest of lexical trees including MWEs identified in the sentence. Each MWE is represented by a constituency-like tree, which permits complex lexical units like MWE embeddings (for example, [[*Los Angeles*] *Lakers*], I will [*take a* [*rain check*]]). The syntactic and lexical structures are factorized in the sense that they share lexical elements: both tokens and fixed MWEs.

The proposed parsing model is an extension of a classical arc-standard parser, integrating specific transitions for MWE detection. In order to deal with the two linguistic dimensions separately, it uses two stacks (instead of one). It is synchronized by using a single buffer, in order to handle the factorization of the two structures. It also includes different hard constraints on the system in order to reduce ambiguities artificially created by the addition of new transitions. To the best of our knowledge, this system is the first transition-based parser that includes a specific mechanism for handling MWEs in two dimensions. Previous related research has usually proposed either pipeline approaches with MWE identification performed either before or after dependency parsing (Kong et al., 2014; Vincze et al., 2013a) or workaround joint solutions using off-the-shelf parsers trained on dependency treebanks where MWEs are annotated by specific subtrees (Nivre and Nilsson, 2004; Eryiğit et al., 2011; Vincze et al., 2013b; Candito and Constant, 2014; Nasr et al., 2015).

2 Syntactic and Lexical Representations

A standard dependency tree represents syntactic structure by establishing binary syntactic relations between words. This is an adequate representation of both syntactic and lexical structure on the assumption that words and lexical units are in a one-to-one correspondence. However, as argued in the introduction, this assumption is broken by the existence of MWEs, and we therefore need to distinguish lexical units as distinct from words.

In the new representation, each lexical unit – whether a single word or an MWE – is associated with a *lexical node*, which has linguistic attributes such as surface form, lemma, part-of-speech tag and morphological features. With an obvious reuse of terminology from context-free grammar, lexical nodes corresponding to MWEs are said to be *non-terminal*, because they have other lexical nodes as children, while lexical nodes corresponding to single words are *terminal* (and do not have any children).

Some lexical nodes are also *syntactic nodes*, that is, nodes of the syntactic dependency tree. These nodes are either non-terminal nodes corresponding to (complete) fixed MWEs or terminal nodes corresponding to words that do not belong to a fixed MWE. Syntactic nodes are connected into a tree structure by binary, asymmetric dependency relations pointing from a *head* node to a *dependent* node.

Figure 1 shows the representation of the sentence *the prime minister made a few good decisions*. It contains three non-terminal lexical nodes: one fixed MWE (*a few*), one contiguous non-fixed MWE (*prime minister*) and one discontiguous non-fixed MWE (*made decisions*). Of these, only the first is also a syntactic node. Note that, for reasons of clarity, we have suppressed the lexical children of the fixed MWE in Figure 1. (The non-terminal node corresponding to *a few* has the lexical children *a* and *few*.) For the same reason, we are not showing the linguistic attributes of lexical nodes. For example, the node *made-decisions* has the following set of features: surface-form='made decisions', lemma='make decision', POS='V'. Non-fixed MWEs have regular syntax and their components might have some autonomy. For example, in the light verb construction *made-decisions*, the noun *decisions* is modified by the adjective *good* that is not an element of the MWE.

The proposed representation of fixed MWEs is an alternative to using special dependency labels as has often been the case in the past (Nivre and Nilsson, 2004; Eryiğit et al., 2011). In addition

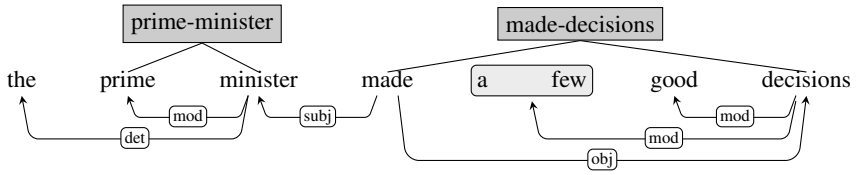


Figure 1: Representation of syntactic and lexical structure.

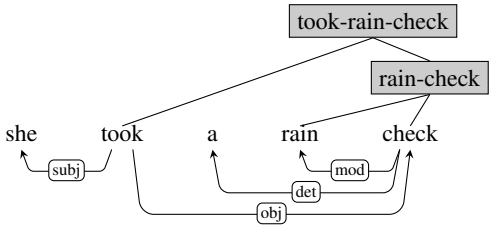


Figure 2: Lexical structure of embedded MWEs.

to special labels, MWEs are then represented as a flat subtree of the syntactic tree. The root of the subtree is the left-most or right-most element of the MWE, and all the other elements are attached to this root with dependencies having special labels. Despite the special labels, these subtrees look like ordinary dependency structures and may confuse a syntactic parser. In our representation, fixed MWEs are instead represented by nodes that are atomic with respect to syntactic structure (but complex with respect to lexical structure), which makes it easier to store linguistic attributes that belong to the fixed MWE and cannot be derived from its components. The new representation also allows us to represent the hierarchical structure of embedded MWEs. Figure 2 provides an analysis of *she took a rain check* that includes such an embedding. The lexical node *took-rain-check* corresponds to a light verb construction where the object is a compound noun that keeps its semantic interpretation whereas the verb has a neutral value. One of its children is the lexical node *rain-check* corresponding to a compound noun.

Let us now define the representation formally. Given a sentence $x = x_1, \ldots, x_n$ consisting of n tokens, the syntactic and lexical representation is a quadruple (V, F, N, A), where

1. V is the set of *terminal nodes*, corresponding one-to-one to the tokens $x_1, \ldots, x_n$,
2. F is a set of n-ary trees on V, with each tree corresponding to a fixed MWE and the root labeled with the part-of-speech tag for the MWE,
3. N is a set of n-ary trees on F, with each tree corresponding to a non-fixed MWE and the root labeled with the part-of-speech tag for the MWE,
4. A is a set of labeled dependency arcs defining a tree over F.

This is a generalization of the standard definition of a dependency tree (see, for example, Kübler et al. (2009)), where the dependency structure is defined over an intermediate layer of lexical nodes (F) instead of directly on the terminal nodes (V), with an additional layer of non-fixed MWEs added on top. To exemplify the definition, here are the formal structures corresponding to the representation visualized in Figure 1.

$$\begin{array}{lcl} V & = & \{1,2,3,4,5,6,7,8\} \\ F & = & \{1,2,3,4,\mathrm{A}(5,6),7,8\} \\ N & = & \{1,\mathrm{N}(2,3),\mathrm{V}(4,8),\mathrm{A}(5,6),7\} \\ A & = & \{(3,\mathrm{det},1),(3,\mathrm{mod},2), \\ & & (4,\mathrm{subj},3),(4,\mathrm{obj},8), \\ & & (8,\mathrm{mod},\mathrm{A}(5,6)),(8,\mathrm{mod},7)\} \end{array}$$

Terminal nodes are represented by integers corresponding to token positions, while trees are represented by n-ary terms $t(c_1, \ldots, c_n)$, where t is a part-of-speech tag and $c_1, \ldots, c_n$ are the subtrees immediately dominated by the root of the tree. The total set of lexical nodes is $L = V \cup F \cup N$, where V contains the terminal and $(F \cup N) - V$ the non-terminal lexical nodes. The set of syntactic nodes is simply F.

It is worth noting that the representation imposes some limitations on what MWEs can be represented. In particular, we can only represent overlapping MWEs if they are cases of embedding, that is, cases where one MWE is properly contained in the other. For example, in an example like *she took a walk then a bath*, it might be argued that *took* should be part of two lexical units: *took-walk* and *took-bath*. This cannot currently be represented. By contrast, we can accommodate cases where two lexical units are interleaved, as in the French example *il prend un cachet et demi*, with the two units *prend-cachet* and *un-et-demi*, which occur in the crossed pattern A1 B1 A2 B2. However, while these cases can be represented in

principle, the parsing model we propose will not be capable of processing them.

Finally, it is worth noting that, although our representation in general allows lexical nodes with arbitrary branching factor for flat MWEs, it is often convenient for parsing to assume that all trees are binary (Crabbé, 2014). For the rest of the paper, we therefore assume that non-binary trees are always transformed into equivalent binary trees using either right or left binarization. Such transformations add intermediate temporary nodes that are only used for internal processing.

3 Transition-Based Model

A transition-based parser is based on three components: a transition system for mapping sentences to their representation, a model for scoring different transition sequences (derivations), and a search algorithm for finding the highest scoring transition sequence for a given input sentence. Following Nivre (2008), we define a transition system as a quadruple $S = (C, T, c_s, C_t)$ where:

1. C is a set of configurations,
2. T is a set of transitions, each of which is a partial function $t : C \rightarrow C$,
3. c_s is an initialization function that maps each input sentence x to an initial configuration $c_s(x) \in C$,
4. $C_t \subseteq C$ is a set of terminal configurations.

A *transition sequence* for a sentence x is a sequence of configurations $C_{0,m} = c_0, \ldots, c_m$ such that $c_0 = c_s(x)$, $c_m \in C_t$, and for every c_i $(0 \leq i < m)$ there is some transition $t \in T$ such that $t(c_i) = c_{i+1}$. Every transition sequence defines a representation for the input sentence.

Training a transition-based parser means training the model for scoring transition sequences. This requires an *oracle* that determines what is an optimal transition sequence given an input sentence and the correct output representation (as given by treebank). Static oracles define a single unique transition sequence for each input-output pair. Dynamic oracles allow more than one optimal transition sequence and can also score non-optimal sequences (Goldberg and Nivre, 2013). Once a scoring model has been trained, parsing is usually performed as best-first search under this model, using greedy search or beam search.

3.1 Arc-Standard Dependency Parsing

Our starting point is the arc-standard transition system for dependency parsing first defined in Nivre (2004) and represented schematically in Figure 3. A configuration in this system consists of a triple $c = (\sigma, \beta, A)$, where σ is a stack containing partially processed nodes, β is a buffer containing remaining input nodes, and A is a set of dependency arcs. The initialization function maps $x = x_1, \ldots, x_n$ to $c_s(x) = ([\,], [1, \ldots, n], \{\,\})$, and the set C_t of terminal configurations contains any configuration of the form $c = ([i], [\,], A)$. The dependency tree defined by such a terminal configuration is $(\{1, \ldots, n\}, A)$. There are three possible transitions:

- **Shift** takes the first node in the buffer and pushes it onto the stack.
- **Right-Arc(**k**)** adds a dependency arc (i, k, j) to A, where j is the first and i the second element of the stack, and removes j from the stack.
- **Left-Arc(**k**)** adds a dependency arc (j, k, i) to A, where j is the first and i the second element of the stack, and removes i from the stack.

A transition sequence in the arc-standard system builds a projective dependency tree over the set of terminal nodes in V. The tree is built bottom-up by attaching dependents to their head and removing them from the stack until only the root of the tree remains on the stack.

3.2 Joint Syntactic and Lexical Analysis

To perform joint syntactic and lexical analysis we need to be able to build structure in two parallel dimensions: the syntactic dimension, represented by a dependency tree, and the lexical dimension, represented by a forest of (binary) trees. The two dimensions share the token-level representation, as well as the level of fixed MWEs, but the syntactic tree and the non-fixed MWEs are independent.

We extend the parser configuration to use two stacks, one for each dimension, but only one buffer. In addition, we need not only a set of dependency arcs, but also a set of lexical units. A configuration in the new system therefore consists of a quintuple $c = (\sigma_1, \sigma_2, \beta, A, L)$, where σ_1 and σ_2 are stacks containing partially processed nodes (which may now be complex MWEs), β is a buffer containing remaining input nodes (which

Initial:	$([\,], [0, \ldots, n], \{\,\})$		
Terminal:	$([i], [\,], A)$		
Shift:	$(\sigma, i\|\beta, A)$	$\Rightarrow$	$(\sigma\|i, \beta, A)$
Right-Arc(k):	$(\sigma\|i\|j, \beta, A)$	$\Rightarrow$	$(\sigma\|i, \beta, A \cup \{(i, k, j)\})$
Left-Arc(k):	$(\sigma\|i\|j, \beta, A)$	$\Rightarrow$	$(\sigma\|j, \beta, A \cup \{(j, k, i)\})$

Figure 3: Arc-standard transition system.

Initial:	$([\,], [\,], [0, \ldots, n], \{\,\}, \{\,\})$		
Terminal:	$([x], [\,], [\,], A, L)$		
Shift:	$(\sigma_1, \sigma_2, i\|\beta, A, L)$	$\Rightarrow$	$(\sigma_1\|i, \sigma_2\|i, \beta, A, L)$
Right-Arc(k):	$(\sigma_1\|x\|y, \sigma_2, \beta, A, L)$	$\Rightarrow$	$(\sigma_1\|x, \sigma_2, \beta, A \cup \{(x, k, y)\}, L)$
Left-Arc(k):	$(\sigma_1\|x\|y, \sigma_2, \beta, A, L)$	$\Rightarrow$	$(\sigma_1\|y, \sigma_2, \beta, A \cup \{(y, k, x)\}, L)$
Merge$_F$(t):	$(\sigma_1\|x\|y, \sigma_2\|x\|y, \beta, A, L)$	$\Rightarrow$	$(\sigma_1\|t(x, y), \sigma_2\|t(x, y), \beta, A, L)$
Merge$_N$(t):	$(\sigma_1, \sigma_2\|x\|y, \beta, A, L)$	$\Rightarrow$	$(\sigma_1, \sigma_2\|t(x, y), \beta, A, L)$
Complete:	$(\sigma_1, \sigma_2\|x, \beta, A, L)$	$\Rightarrow$	$(\sigma_1, \sigma_2, \beta, A, L \cup \{x\})$

Figure 4: Transition system for joint syntactic and lexical analysis.

are always tokens), A is a set of dependency arcs, and L is a set of lexical units (tokens or MWEs). The initialization function maps $x = x_1, \ldots, x_n$ to $c_s(x) = ([\,], [\,], [1, \ldots, n], \{\,\}, \{\,\})$, and the set C_t of terminal configurations contains any configuration of the form $c = ([x], [\,], [\,], A, L)$. The dependency tree defined by such a terminal configuration is (F, A), and the set of lexical units is $V \cup L$. Note that the set F of syntactic nodes is not explicitly represented in the configuration but is implicitly defined by A. Similarly, the set L only contains $F \cup N$.

The new transition system is shown in Figure 4. There are now six possible transitions:

- **Shift** takes the first node in the buffer and pushes it onto both stacks. This guarantees that the two dimensions are synchronized at the token level.
- **Right-Arc(k)** adds a dependency arc (x, k, y) to A, where y is the first and x the second element of the syntactic stack (σ_1), and removes y from this stack. It does not affect the lexical stack (σ_2).[1]
- **Left-Arc(k)** adds a dependency arc (y, k, x) to A, where y is the first and x the second element of the syntactic stack (σ_1), and removes x from this stack. Like **Right-Arc(k)**, it does not affect the lexical stack (σ_2).
- **Merge$_F$(t)** applies in a configuration where the two top elements x and y are identical on both stacks and combines these elements into a tree $t(x, y)$ representing a fixed MWE with part-of-speech tag t. Since it operates on both stacks, the new element will be a syntactic node as well as a lexical node.
- **Merge$_N$(t)** combines the two top elements x and y on the lexical stack (σ_2) into a tree $t(x, y)$ representing a non-fixed MWE with part-of-speech tag t. Since it only operates on the lexical stack, the new element will not be a syntactic node.
- **Complete** moves the top element x on the lexical stack (σ_2) to L, making it a final lexical unit in the output representation. Note that x can be a simple token, a fixed MWE (created on both stacks), or a non-fixed MWE (created only on the lexical stack).

A transition sequence in the new system derives the set of lexical nodes and simultaneously builds a projective dependency tree over the set of syntactic nodes. By way of example, Figure 5 shows the transition sequence for the example in Figure 1.

3.3 Implicit Completion

The system presented above has one potential drawback: it needs a separate Complete transition for every lexical unit, even in the default case

[1] We use the variables x and y, instead of i and j, because the stack elements can now be complex lexical units as well as simple tokens.

Transition		Configuration
		$([\,],[\,],[1,2,3,4,5,6,7,8],A_0=\{\,\},L_0=\{\,\})$
Shift	$\Rightarrow$	$([1],[1],[2,3,4,5,6,7,8],A_0,L_0)$
Complete	$\Rightarrow$	$([1],[\,],[2,3,4,5,6,7,8],A_0,L_1=L_0\cup\{1\})$
Shift	$\Rightarrow$	$([1,2],[2],[3,4,5,6,7,8],A_0,L_1)$
Shift	$\Rightarrow$	$([1,2,3],[2,3],[4,5,6,7,8],A_0,L_1)$
Merge$_N$(N)	$\Rightarrow$	$([1,2,3],[\mathrm{N}(2,3)],[4,5,6,7,8],A_1,L_1)$
Complete	$\Rightarrow$	$([1,2,3],[\,],[4,5,6,7,8],A_0,L_2=L_1\cup\{\mathrm{N}(2,3)\})$
Left-Arc(mod)	$\Rightarrow$	$([1,3],[\,],[4,5,6,7,8],A_1=A_0\cup\{(3,\mathrm{mod},2)\},L_2)$
Left-Arc(det)	$\Rightarrow$	$([3],[\,],[4,5,6,7,8],A_2=A_1\cup\{(3,\mathrm{det},1)\},L_2)$
Shift	$\Rightarrow$	$([3,4],[4],[5,6,7,8],A_2,L_2)$
Left-Arc(subj)	$\Rightarrow$	$([4],[4],[5,6,7,8],A_3=A_2\cup\{(4,\mathrm{subj},3)\},L_2)$
Shift	$\Rightarrow$	$([4,5],[4,5],[6,7,8],A_3,L_2)$
Shift	$\Rightarrow$	$([4,5,6],[4,5,6],[7,8],A_3,L_2)$
Merge$_F$(A)	$\Rightarrow$	$([4,\mathrm{A}(5,6)],[4,\mathrm{A}(5,6)],[7,8],A_3,L_2)$
Complete	$\Rightarrow$	$([4,\mathrm{A}(5,6)],[4],[7,8],A_3,L_3=L_2\cup\{\mathrm{A}(5,6)\})$
Shift	$\Rightarrow$	$([4,\mathrm{A}(5,6),7],[4,7],[8],A_3,L_3)$
Complete	$\Rightarrow$	$([4,\mathrm{A}(5,6),7],[4],[8],A_3,L_4=L_3\cup\{7\})$
Shift	$\Rightarrow$	$([4,\mathrm{A}(5,6),7,8],[4,8],[\,],A_3,L_4)$
Left-Arc(mod)	$\Rightarrow$	$([4,\mathrm{A}(5,6),8],[4,8],[\,],A_4=A_3\cup\{(8,\mathrm{mod},7)\},L_4)$
Left-Arc(mod)	$\Rightarrow$	$([4,8],[4,8],[\,],A_5=A_4\cup\{(8,\mathrm{mod},\mathrm{A}(5,6))\},L_4)$
Merge$_N$(V)	$\Rightarrow$	$([4,8],[\mathrm{V}(4,8)],[\,],A_5,L_4)$
Complete	$\Rightarrow$	$([4,8],[\,],[\,],A_5,L_5=L_4\cup\{\mathrm{V}(4,8)\})$
Right-Arc(obj)	$\Rightarrow$	$([4],[\,],[\,],A_6=A_5\cup\{(4,\mathrm{obj},8)\},L_5)$

Figure 5: Transition sequence for joint syntactic and lexical analysis.

when a lexical unit is just a token. This makes sequences much longer and increases the inherent ambiguity. One way to deal with this problem is to make the Complete transition implicit and deterministic, so that it is not scored by the model (or predicted by a classifier in the case of deterministic parsing) but is performed as a side effect of the Right-Arc and Left-Arc transitions. Every time we apply one of these transitions, we check whether the dependent x of the new arc is part of a unit y on the lexical stack satisfying one of the following conditions: (i) $x = y$; (ii) x is a lexical child of y and every lexical node z in y either has a syntactic head in A or is the root of the dependency tree. If (i) or (ii) is satisfied, we move y from the lexical stack to the set L of lexical units as a side effect of the arc transition.

4 Experiments

This section provides experimental results obtained with a simple implementation of our system using a greedy search parsing algorithm and a linear model trained with an averaged perceptron with shuffled examples and a static oracle. More precisely, the static oracle is defined using the following transition priorities: Merge$_F$ > Merge$_N$ > Complete > LeftArc > RightArc > Shift. At each state of the training phase, the static oracle selects the valid transition that has the higher priority.

We evaluated the two variants of the system, namely *Explicit* and *Implicit*, with explicit and implicit completion, respectively. They were compared against the joint approach proposed in Candito and Constant (2014) that we applied to an arc-standard parser, instead of a graph-based parser. The parser is trained on a treebank where MWE status and grammatical function are concatenated in arc labels. We consider it as the *Baseline*. We used classical transition-based parsing features consisting of patterns combining linguistic attributes of nodes on the stacks and the buffer, as well as processed subtrees and transition history. We can note that the joint systems do not contain features sharing elements of both stacks. Preliminary tuning experiments did not show gains when using such features.

We also compared these systems against weaker ones, obtained by disabling some transitions and using one stack only. Two systems, namely *Syntactic-baseline* and *Syntactic* only predict the syntactic nodes and the dependency structure by using respectively a baseline parser and our system where neither the lexical stack nor the Merge$_N$ and Complete transitions are used. The latter one is an implementation of the proposal in Nivre (2014). Two systems are devoted only to the lexical layer: *Lexical* only recognizes the lexical units (only the lexical stack and the Merge$_N$ and Complete transitions are activated); *Fixed* only identifies the fixed expressions.

Corpus	EWT		FTB		
	Train	Test	Train	Dev	Test
# sent.	3,312	500	14,759	1,235	2,541
# tokens	48,408	7,171	443,113	38,820	75,216
# MWEs	2,996	401	23,556	2,119	4,043
# fixed	-	-	10,987	925	1,992

Table 1: Dataset statistics.

We also implemented pipeline systems where: (i) fixed MWEs are identified by applying only the *Fixed* system; (ii) elements of predicted MWEs are merged into single tokens; (iii) the retokenized text is parsed using the *Baseline* or *Implicit* systems trained on a dataset where fixed MWEs consist of single tokens.

We carried out our experiments on two different datasets annotating both the syntactic structure and the MWEs: the French Treebank [FTB] (Abeillé et al., 2003) and the STREUSLE corpus (Schneider et al., 2014b) combined with the English Web Treebank [EWT] (Bies et al., 2012). They are commonly used for evaluating the most recent MWE-aware dependency parsers and supervised MWE identification systems. Concerning the FTB, we used the dependency version developed in Candito and Constant (2014) derived from the SPMRL shared task version (Seddah et al., 2013). Fixed and non-fixed MWEs are distinguished, but are limited to contiguous ones only. The STREUSLE corpus (Schneider et al., 2014b) corresponds to a subpart of the English Web Treebank (EWT). It consists of reviews and is comprehensively annotated in contiguous and discontiguous MWEs. Fixed and non-fixed expressions are not distinguished though the distinction between non-compositional and collocational MWEs is made. This implies that the Merge$_F$ transition is not used on this dataset. Practically, we used the LTH converter (Johansson and Nugues, 2007) to obtain the dependency version of the EWT constituent version. We also used the predicted linguistic attributes used in Constant and Le Roux (2015) and in Constant et al. (2016). Both datasets include predicted POS tags, lemmas and morphology, as well as features computed from compound dictionary lookup. None of them is entirely satisfying with respect to our model, but they allow us to evaluate the feasibility of the approach. Statistics on the two datasets are provided in Table 1.

Results are provided in Table 2 for French and in Table 3 for English. In order to evaluate the syntactic layer, we used classical UAS and LAS metrics. Before evaluation, merged units were automatically decomposed in the form of flat subtrees using specific arcs as in Seddah et al. (2013), so all systems can be evaluated and compared at the token level. MWE identification is evaluated with the F-score of the MWE segmentation, namely *MWE* for all MWEs and *FMWE* for fixed MWEs only. An MWE segment corresponds to the set of its component positions in the input token sequence.

First, results show that our joint system consistently and significantly outperforms the baseline in terms of MWE identification on both datasets. The merge transitions play a key role. In terms of syntax, the *Explicit* system does not have any positive impact (on par or degraded scores), whereas the *Implicit* system allows us to obtain slightly better results on French and a significant improvement on English. The very good performances on English might be explained by the fact that it contains a non-negligeable set of discontiguous MWEs which complicates the prediction of explicit Complete transitions.

When compared with weaker systems, we can see that the addition of the lexical layer helps improve the prediction of the syntactic layer, which confirms results on symbolic parsing (Wehrli, 2014). The syntactic layer does not seem to impact the lexical layer prediction: we observe comparable results. This might be due to the fact that syntax is helpful for long-distance discontiguity only, which does not appear in our datasets (the English dataset contains MWEs with small gaps). Another explanation could also be that syntactic parsing accuracy is rather low due to the use of a simple greedy algorithm. Developing more advanced transition-based parsing methods like beam-search may help improve both syntactic parsing accuracy and MWE identification. When comparing joint systems with pipeline ones, we can see that preidentifying fixed MWEs seems to help MWE identification whereas syntactic parsing accuracy tends to be slightly lower. One hypothesis could be that Merge$_F$ transitions may confuse the prediction of Merge$_N$ transitions.

When compared with existing state-of-the-art systems, we can see that the proposed systems achieve MWE identification scores that are comparable with the pipeline and joint approaches used in Candito and Constant (2014) with a graph-

	DEV				TEST			
System	UAS	LAS	MWE	FMWE	UAS	LAS	MWE	FMWE
Baseline	86.28	83.67	77.2	83.2	84.85	82.67	75.5	81.9
Explicit	86.36	83.77	79.7	86.0	84.98	82.79	**79.3**	**84.8**
Implicit	**86.61**	**84.10**	**80.0**	**86.2**	**85.04**	**82.93**	78.4	84.3
Syntactic only -Baseline	86.31	83.69	-	83.5	84.89	82.70	-	82.0
Syntactic only	86.39	83.77	-	85.0	85.02	82.84	-	83.8
Lexical only	-	-	80.0	-	-	-	79.5	-
Fixed only	-	-	-	85.7	-	-	-	85.7
Pipeline (Fixed only → Baseline)	85.33	83.29	80.6	85.7	84.86	82.86	80.4	85.7
Pipeline (Fixed only → Implicit)	85.49	83.50	**81.8**	85.7	84.84	82.89	**81.1**	**85.7**
graph-based (Candito and Constant, 2014)	89.7	87.5	77.6	85.4	89.21	86.92	77.0	85.1
CRF+graph-based (Candito and Constant, 2014)	89.8	87.4	79.0	85.0	86.97	89.24	78.6	86.3
CRF (SPMRL) (Le Roux et al., 2014)	-	-	82.4	-	-	-	80.5	-

Table 2: Results on the FTB. To reduce bias due to training with shuffled examples, scores are averages of 3 different training/parsing runs.

	TRAIN Cross-validation			TEST		
System	UAS	LAS	MWE	UAS	LAS	MWE
Baseline	86.16	81.76	49.6	86.31	82.02	46.8
Explicit	86.25	82.09	52.9	86.05	81.68	**53.4**
Implicit	**86.81**	**82.68**	**55.0**	**87.05**	**83.14**	51.6
Syntactic only	86.35	82.23	-	86.41	82.20	-
Lexical only	-	-	54.5	-	-	53.6
(Schneider et al., 2014a)	-	-	-	-	-	53.85

Table 3: Results on the reviews part of the English Web Treebank, via cross-validation on the training set with 8 splits, and simple validation on the test set.

based parser for French, and the base sequence tagger using a perceptron model with rich MWE-dedicated features of Schneider et al. (2014a) for English. It reaches lower scores than the best simple CRF-based MWE tagging system of Le Roux et al. (2014). These scores are obtained on the SPMRL shared task version, though they are not entirely comparable with our system as they do not distinguish fixed from non-fixed MWEs.

5 Related work

The present paper proposes a new representation for lexical and syntactic analysis in the framework of syntactic dependency parsing. Most existing MWE-aware dependency treebanks represent an MWE as a flat subtree of the syntactic tree with special labels, like in the UD treebanks (Nivre et al., 2016) or in the SPMRL shared task (Seddah et al., 2013), or in other individual treebanks (Nivre and Nilsson, 2004; Eryiğit et al., 2011). Such representation enables MWE discontinuity, but the internal syntactic structure is not annotated. Candito and Constant (2014) proposed a representation where the irregular and regular MWEs are distinguished: irregular MWEs are integrated in the syntactic tree as above; regular MWEs are annotated in their component attributes while their internal structure is annotated in the syntactic tree. The Prague Dependency Treebank (Bejček et al., 2013) has several interconnected annotation layers: morphological (*m*-layer), syntactic (*a*-layer) and semantic (*t*-layer). All these layers are trees that are interconnected. MWEs are annotated on the *t*-layer and are linked to an MWE lexicon (Bejček and Straňák, 2010). Constant and Le Roux (2015) proposed a dependency representation of lexical segmentation allowing annotations of deeper phenomena like MWE nesting. More details on MWE-aware treebanks (including constituent ones) can be found in Rosén et al. (2015).

Statistical MWE-aware dependency parsing has received a growing interest since Nivre and Nilsson (2004). The main challenge resides in finding the best orchestration strategy. Past research has explored either pipeline or joint approaches. Pipeline strategies consist in positioning the MWE recognition either before or after the parser itself, as in Nivre and Nilsson (2004), Eryiğit et al. (2011), Constant et al. (2013), and Kong et al. (2014) for pre-identification and as in Vincze et al. (2013a) for post-identification. Joint strategies have mainly consisted in using off-the-shelf

parsers and integrating MWE annotation in the syntactic structure, so that MWE identification is blind for the parser (Nivre and Nilsson, 2004; Eryiğit et al., 2011; Seddah et al., 2013; Vincze et al., 2013b; Candito and Constant, 2014; Nasr et al., 2015).

Our system includes a special treatment of MWEs using specific transitions in a classical transition-based system, in line with the proposal of Nivre (2014). Constant et al. (2016) also proposed a two-dimensional representation in the form of dependency trees anchored by the same words. The annotation of fixed MWEs is redundant on both dimensions, while they are shared in our representation. They propose, along with this representation, an adaptation of an easy-first parser able to predict both dimensions. Contrary to our system, there are no special mechanisms for treating MWEs.

The use of multiple stacks to capture partly independent dimensions is inspired by the multiplanar dependency parser of Gómez-Rodríguez and Nivre (2013). Our parsing strategy for (hierarchical) MWEs is very similar to the deterministic constituency parsing method of Crabbé (2014).

6 Conclusion

This paper proposes a transition-based system that extends a classical arc-standard parser to handle both lexical and syntactic analysis. It is based on a new representation having two linguistic layers sharing lexical nodes. Experimental results show that MWE identification is greatly improved with respect to the mainstream joint approach. This can be a useful starting point for several lines of research: implementing more advanced transition-based techniques (beam search, dynamic oracles, deep learning); extending other classical transition systems like arc-eager and hybrid as well as handling non-projectivity.

Acknowledgments

The authors would like to thank Marie Candito for her fruitful inputs. This work has been partly funded by the French Agence Nationale pour la Recherche, through the PARSEME-FR project (ANR-14-CERA-0001). This work has also been supported in part by the PARSEME European COST Action (IC1207).

References

Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

Timothy Baldwin and Su Nam Kim. 2010. Multiword Expressions. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*, pages 267–292. CRC Press, Taylor and Francis Group, Boca Raton, FL, USA, 2 edition.

Eduard Bejček and Pavel Straňák. 2010. Annotation of Multiword Expressions in the Prague Dependency Treebank. *Language Resources and Evaluation*, 44(1-2):7–21.

Eduard Bejček, Eva Hajičová, Jan Hajič, Pavlína Jínová, Václava Kettnerová, Veronika Kolářová, Marie Mikulová, Jiří Mírovský, Anna Nedoluzhko, Jarmila Panevová, Lucie Poláková, Magda Ševčíková, Jan Štěpánek, and Šárka Zikánová. 2013. Prague Dependency Treebank 3.0.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English web treebank.

Marie Candito and Matthieu Constant. 2014. Strategies for Contiguous Multiword Expression Analysis and Dependency Parsing. In *ACL 14 - The 52nd Annual Meeting of the Association for Computational Linguistics*, Baltimore, United States, June. ACL.

Matthieu Constant and Joseph Le Roux. 2015. Dependency Representations for Lexical Segmentation. In *6th Workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL 2015)*, Bilbao, Spain, July.

Matthieu Constant, Marie Candito, and Djamé Seddah. 2013. The LIGM-Alpage Architecture for the SPMRL 2013 Shared Task: Multiword Expression Analysis and Dependency Parsing. In *Fourth Workshop on Statistical Parsing of Morphologically Rich Languages*, pages 46–52, Seattle, United States, October.

Matthieu Constant, Joseph Le Roux, and Nadi Tomeh. 2016. Deep lexical segmentation and syntactic parsing in the easy-first dependency framework. In *Proceeedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2016)*.

Benoît Crabbé. 2014. An LR-inspired generalized lexicalized phrase structure parser. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 541–552.

Gülşen Eryiğit, Tugay İlbay, and Ozan Arkan Can. 2011. Multiword Expressions in Statistical Dependency Parsing. In *Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich*

Languages, SPMRL '11, pages 45–55, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2013. Training Deterministic Parsers with Non-Deterministic Oracles. *TACL*, 1:403–414.

Carlos Gómez-Rodríguez and Joakim Nivre. 2013. Divisible Transition Systems and Multiplanar Dependency Parsing. *Comput. Linguist.*, 39(4):799–845, December.

Richard Johansson and Pierre Nugues. 2007. Extended Constituent-to-dependency Conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. Dependency parsing for tweets. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Doha, Qatar, October. Association for Computational Linguistics.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan and Claypool.

Joseph Le Roux, Antoine Rozenknop, and Matthieu Constant. 2014. Syntactic Parsing and Compound Recognition via Dual Decomposition: Application to French. In *COLING*.

Alexis Nasr, Carlos Ramisch, José Deulofeu, and André Valli. 2015. Joint Dependency Parsing and Multiword Expression Tokenization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1116–1126.

Joakim Nivre and Jens Nilsson. 2004. Multiword units in syntactic parsing. *Proceedings of Methodologies and Evaluation of Multiword Units in Real-World Applications (MEMURA)*.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Dan Zeman. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*.

Joakim Nivre. 2004. Incrementality in Deterministic Dependency Parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together (ACL)*, pages 50–57.

Joakim Nivre. 2008. Algorithms for Deterministic Incremental Dependency Parsing. *Comput. Linguist.*, 34(4):513–553, December.

Joakim Nivre. 2014. Transition-Based Parsing with Multiword Expressions. In *2^{n}d PARSEME General Meeting, Athens, Greece*.

Carlos Ramisch. 2015. *Multiword Expressions Acquisition: A Generic and Open Framework*, volume XIV of *Theory and Applications of Natural Language Processing*. Springer.

Victoria Rosén, Gyri Smørdal Losnegaard, Koenraad De Smedt, Eduard Bejček, Agata Savary, Adam Przepiórkowski, Petya Osenova, and Verginica Mititelu. 2015. A survey of multiword expressions in treebanks. In *Proceedings of the 14th International Workshop on Treebanks and Linguistic Theories (TLT14)*.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword Expressions: A Pain in the Neck for NLP. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 2276 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014a. Discriminative lexical semantic segmentation with gaps: running the MWE gamut. *Transactions of the Association for Computational Linguistics (TACL)*.

Nathan Schneider, Spencer Onuffer, Nora Kazour, Emily Danchik, Michael T. Mordowanec, Henrietta Conrad, and Noah A. Smith. 2014b. Comprehensive Annotation of Multiword Expressions in a Social Web Corpus. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 455–461, Reykjavík, Iceland, May. ELRA.

Djamé Seddah, Reut Tsarfaty, Sandra K$'$ubler, Marie Candito, Jinho Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiorkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clérgerie. 2013. Overview of the SPMRL 2013 Shared Task: A Cross-Framework Evaluation of Parsing Morphologically Rich Languages. In *Proceedings of the 4th Workshop on Statistical Parsing of Morphologically Rich Languages*, Seattle, WA.

Violeta Seretan. 2011. *Syntax-Based Collocation Extraction*. Text, Speech and Language Technology. Springer.

Veronika Vincze, István Nagy T., and Richárd Farkas. 2013a. Identifying english and hungarian light verb constructions: A contrastive approach. In *Proceedings of the 51st Annual Meeting of the Association*

for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 2: Short Papers, pages 255–261.

Veronika Vincze, János Zsibrita, and Istvàn Nagy T. 2013b. Dependency parsing for identifying hungarian light verb constructions. In *Proceedings of International Joint Conference on Natural Language Processing (IJCNLP 2013)*, Nagoya, Japan.

Eric Wehrli. 2014. The Relevance of Collocations for Parsing. In *Proceedings of the 10th Workshop on Multiword Expressions (MWE)*, pages 26–32, Gothenburg, Sweden, April. Association for Computational Linguistics.

Neural Greedy Constituent Parsing with Dynamic Oracles

Maximin Coavoux[1,2] and **Benoît Crabbé**[1,2,3]
[1]Univ. Paris Diderot, Sorbonne Paris Cité
[2]Alpage, Inria
[3]Institut Universitaire de France
maximin.coavoux@inria.fr
benoit.crabbe@linguist.univ-paris-diderot.fr

Abstract

Dynamic oracle training has shown substantial improvements for dependency parsing in various settings, but has not been explored for constituent parsing. The present article introduces a dynamic oracle for transition-based constituent parsing. Experiments on the 9 languages of the SPMRL dataset show that a neural greedy parser with morphological features, trained with a dynamic oracle, leads to accuracies comparable with the best non-reranking and non-ensemble parsers.

1 Introduction

Constituent parsing often relies on search methods such as dynamic programming or beam search, because the search space of all possible predictions is prohibitively large. In this article, we present a greedy parsing model. Our main contribution is the design of a dynamic oracle for transition-based constituent parsing. In NLP, dynamic oracles were first proposed to improve greedy dependency parsing training without involving additional computational costs at test time (Goldberg and Nivre, 2012; Goldberg and Nivre, 2013).

The training of a transition-based parser involves an oracle, that is a function mapping a configuration to the best transition. Transition-based parsers usually rely on a static oracle, only well-defined for gold configurations, which transforms trees into sequences of gold actions. Training against a static oracle restricts the exploration of the search space to the gold sequence of actions. At test time, due to error propagation, the parser will be in a very different situation than at training time. It will have to infer good actions from noisy configurations. To alleviate error propagation, a solution is to train the parser to predict the best action given any configuration, by allowing it to explore a greater part of the search space at train time. Dynamic oracles are non-deterministic oracles well-defined for any configuration. They give the best possible transitions for any configuration. Although dynamic oracles are widely used in dependency parsing and available for most standard transition systems (Goldberg and Nivre, 2013; Goldberg et al., 2014; Gómez-Rodríguez et al., 2014; Straka et al., 2015), no dynamic oracle parsing model has yet been proposed for phrase structure grammars.

The model we present aims at parsing morphologically rich languages (MRL). Recent research has shown that morphological features are very important for MRL parsing (Björkelund et al., 2013; Crabbé, 2015). However, traditional linear models (such as the structured perceptron) need to define rather complex feature templates to capture interactions between features. Additional morphological features complicate this task (Crabbé, 2015). Instead, we propose to rely on a neural network weighting function which uses a non-linear hidden layer to automatically capture interactions between variables, and embeds morphological features in a vector space, as is usual for words and other symbols (Collobert and Weston, 2008; Chen and Manning, 2014).

The article is structured as follows. In Section 2, we present neural transition-based parsing. Section 3 motivates learning with a dynamic oracle and presents an algorithm to do so. Section 4 introduces the dynamic oracle. Finally, we present parsing experiments in Section 5 to evaluate our proposal.

2 Transition-Based Constituent Parsing

Transition-based parsers for phrase structure grammars generally derive from the work of Sagae

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 172–182,
Berlin, Germany, August 7-12, 2016.

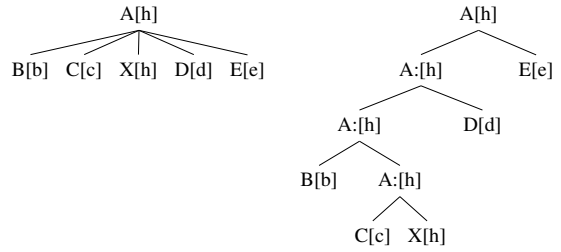


Figure 1: Order-0 head markovization.

and Lavie (2005). In the present paper, we extend Crabbé (2015)'s transition system.

Grammar form We extract the grammar from a head-annotated preprocessed constituent treebank (cf Section 5). The preprocessing involves two steps. First, unary chains are merged, except at the preterminal level, where at most one unary production is allowed. Second, an order-0 head-markovization is performed (Figure 1). This step introduces temporary symbols in the binarized grammar, which are suffixed by ":". The resulting productions have one the following form:

$$X[h] \rightarrow A[a]\ B[b] \quad X[h] \rightarrow A[a]\ b$$
$$X[h] \rightarrow h \quad X[h] \rightarrow a\ B[b]$$

where X, A, B are delexicalised non-terminals, a, b and $h \in \{a, b\}$ are tokens, and $X[h]$ is a lexicalized non-terminal. The purpose of lexicalization is to allow the extraction of features involving the heads of phrases together with their tags and morphological attributes.

Transition System In the transition-based framework, parsing relies on two data structures: a **buffer** containing the sequence of tokens to parse and a **stack** containing partial instantiated trees. A **configuration** $C = \langle j, S, b, \gamma \rangle$ is a tuple where j is the index of the next token in the buffer, S is the current stack, b is a boolean, and γ is the set of constituents constructed so far.[1]

Constituents are instantiated non-terminals, i.e. tuples (X, i, j) such that X is a non-terminal and (i, j) are two integers denoting its span. Although the content of γ could be retrieved from the stack, we make it explicit because it will be useful for the design of the oracle in Section 4.

From an initial configuration $C_0 = \langle 0, \epsilon, \bot, \emptyset \rangle$, the parser incrementally derives new configurations by performing actions until a final configuration is reached. S(HIFT) pops an element from the buffer and pushes it on the stack. R(EDUCE)(X) pops two elements from the stack, and pushes a new non-terminal X on the stack with the two elements as its children. There are two kinds of binary reductions, left (RL) or right (RR), depending on the position of the head. Finally, unary reductions (RU(X)) pops only one element from the stack and pushes a new non-terminal X. A **derivation** $C_{0 \Rightarrow \tau} = C_0 \overset{a_0}{\Rightarrow} \ldots \overset{a_{\tau-1}}{\Rightarrow} C_\tau$ is a sequence of configurations linked by actions and leading to a final configuration. Figure 2 presents the algorithm as a deductive system. G(HOST)R(EDUCE) actions and boolean b ($\top$ or $\bot$) are used to ensure that unary reductions (RU) can only take place once after a SHIFT action.[2]

Constraints on the transitions make sure that predicted trees can be unbinarized. Figure 3 shows two examples of trees that could not have been obtained by the binarization process. In the first tree, a temporary symbol rewrites as two tempo-

Stack: $S\|(C,l,i)\|(B,i,k)\|(A,k,j)$	
Action	Constraints
RL(X) or RR(X), X$\in N$	A$\notin N_{tmp}$ and B $\notin N_{tmp}$
RL(X:) or RR(X:), X:$\in N_{tmp}$	C$\notin N_{tmp}$ or $j < n$
RR(X)	B$\notin N_{tmp}$
RL(X)	A$\notin N_{tmp}$

Table 1: Constraints to ensure that binary trees can be unbinarized. n is the sentence length.

Input $\quad w_0 w_1 \ldots w_{n-1}$

Axiom $\quad \langle 0, \epsilon, \bot, \emptyset \rangle$

S $\quad \dfrac{\langle j, S, \bot, \gamma \rangle}{\langle j+1, S\|(t_j, j, j+1), \top, \gamma \rangle}$

RL(X) $\quad \dfrac{\langle j, S\|(A,i,k)\|(B,k,j), \bot, \gamma \rangle}{\langle j, S\|(X,i,j), \bot, \gamma \cup \{(X,i,j)\} \rangle}$

RU(X) $\quad \dfrac{\langle j, S\|(t_{j-1}, j-1, j), \top, \gamma \rangle}{\langle j, S\|(X, j-1, j), \bot, \gamma \cup \{(X,j-1,j)\} \rangle}$

GR $\quad \dfrac{\langle j, S, \top, \gamma \rangle}{\langle j, S, \bot, \gamma \rangle}$

Figure 2: Transition system, the transition RR(X) and the lexicalization of symbols are omitted.

[1] The introduction of γ is the main difference with Crabbé (2015)'s transition system.

[2] This transition system is similar to the extended system of Zhu et al. (2013). The main difference is the strategy used to deal with unary reductions. Our strategy ensures that derivations for a sentence all have the same number of steps, which can have an effect when using beam search. We use a GHOST-REDUCE action, whereas they use a padding strategy with an IDLE action.

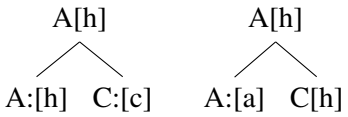


Figure 3: Examples of ill-formed binary trees

rary symbols. In the second one, the head of a temporary symbol is not the head of its direct parent. Table 1 shows a summary of the constraints used to ensure that any predicted tree is a well-formed binarized tree.[3] In this table, N is the set of non-terminals and $N_{tmp} \subset N$ is the set of temporary non-terminals.

Weighted Parsing The deductive system is inherently non-deterministic. Determinism is provided by a scoring function

$$s(C_{0 \Rightarrow \tau}) = \sum_{i=1}^{\tau} f_{\boldsymbol{\theta}}(C_{i-1}, a_i)$$

where $\boldsymbol{\theta}$ is a set of parameters. The score of a derivation decomposes as a sum of scores of actions. In practice, we used a feed-forward neural network very similar to the scoring model of Chen and Manning (2014). The input of the network is a sequence of typed symbols. We consider three main types (non-terminals, tags and terminals) plus a language-dependent set of morphological attribute types, for example, gender, number, or case (Crabbé, 2015). The first layer $\mathbf{h}^{(0)}$ is a lookup layer which concatenates the embeddings of each typed symbol extracted from a configuration. The second layer $\mathbf{h}^{(1)}$ is a non-linear layer with a rectifier activation (ReLU). Finally, the last layer $\mathbf{h}^{(2)}$ is a softmax layer giving a distribution over possible actions, given a configuration. The score of an action is its log probability.

Assuming $\mathbf{v_1}, \mathbf{v_2} \ldots, \mathbf{v}_\alpha$ are the embeddings of the sequence of symbols extracted from a configuration, the forward pass is summed up by the following equations:

$$\begin{aligned}
\mathbf{h}^{(0)} &= [\mathbf{v_1}; \mathbf{v_2}; \ldots; \mathbf{v}_\alpha] \\
\mathbf{h}^{(1)} &= \max\{0, \mathbf{W}^{(h)} \cdot \mathbf{h}^{(0)} + \mathbf{b}^{(h)}\} \\
\mathbf{h}^{(2)} &= \text{Softmax}(\mathbf{W}^{(o)} \cdot \mathbf{h}^{(1)} + \mathbf{b}^{(o)}) \\
f_{\boldsymbol{\theta}}(C, a) &= \log(\mathbf{h}^{(2)}_a)
\end{aligned}$$

[3]There are additional constraints which are not presented here. For example, SHIFT assumes that the buffer is not empty. A full description of constraints typically used in a slightly different transition system can be found in Zhang and Clark (2009)'s appendix section.

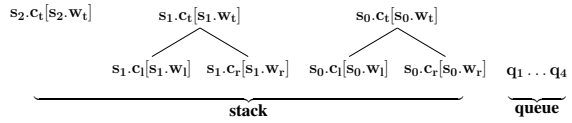


Figure 4: Schematic representation of local elements in a configuration.

Thus, $\boldsymbol{\theta}$ includes the weights and biases for each layer ($\mathbf{W}^{(h)}$, $\mathbf{W}^{(o)}$, $\mathbf{b}^{(h)}, \mathbf{b}^{(o)}$), and the embedding lookup table for each symbol type.

We perform greedy search to infer the best-scoring derivation. Note that this is not an exact inference. Most propositions in phrase structure parsing rely on dynamic programming (Durrett and Klein, 2015; Mi and Huang, 2015) or beam search (Crabbé, 2015; Watanabe and Sumita, 2015; Zhu et al., 2013). However we found that with a scoring function expressive enough and a rich feature set, greedy decoding can be surprisingly accurate (see Section 5).

Features Each terminal is a tuple containing the word form, its part-of-speech tag and an arbitrary number of language-specific morphological attributes, such as CASE, GENDER, NUMBER, ASPECT and others (Seddah et al., 2013; Crabbé, 2015). The representation of a configuration depends on symbols at the top of the two data structures, including the first tokens in the buffer, the first lexicalised non-terminals in the stack and possibly their immediate descendants (Figure 4). The full set of templates is specified in Table 6 of Annex A. The sequence of symbols that forms the input of the network is the instanciation of each position described in this table with a discrete symbol.

3 Training a Greedy Parser with an Oracle

An important component for the training of a parser is an oracle, that is a function mapping a gold tree and a configuration to an action. The oracle is used to generate local training examples from trees, and feed them to the local classifier.

A **static oracle** (Goldberg and Nivre, 2012) is an incomplete and deterministic oracle. It is only well-defined for gold configurations (the configurations derived by the gold action sequence) and returns the unique gold action. Usually, parsers use a static oracle to transform the set of binarized trees into a set $\mathcal{D} = \{C^{(i)}, a^{(i)}\}_{1 \leq i \leq T}$ of training examples. Training consists in minimiz-

ing the negative log likelihood of these examples. The limitation of this training method is that only gold configurations are seen during training. At test time, due to error propagation, the parser will have to predict good actions from noisy configurations, and will have much difficulty to recover after mistakes.

To alleviate this problem, a line of work (Daumé III et al., 2006; Ross et al., 2011) has cast the problem of structured prediction as a search problem and developed training algorithms aiming at exploring a greater part of the search space. These methods require an oracle well-defined for every search state, that is, for every parsing configuration.

A **dynamic oracle** is a complete and non-deterministic oracle (Goldberg and Nivre, 2012). It returns the non-empty set of the best transitions given a configuration and a gold tree. In dependency parsing, starting from Goldberg and Nivre (2012), dynamic oracle algorithms and training methods have been proposed for a variety of transition systems and led to substantial improvements in accuracy (Goldberg and Nivre, 2013; Goldberg et al., 2014; Gómez-Rodríguez et al., 2014; Straka et al., 2015; Gómez-Rodríguez and Fernández-González, 2015).

Online training An online trainer iterates several times over each sentence in the treebank, and updates its parameters until convergence. When a static oracle is used, the training examples can be pregenerated from the sentences. When we use a dynamic oracle instead, we generate training examples on the fly, by following the prediction of the parser (given the current parameters) instead of the gold action, with probability p, where p is a hyperparameter which controls the degree of exploration. The online training algorithm for a single sentence s, with an oracle function o is shown in Figure 5. It is a slightly modified version of Goldberg and Nivre (2013)'s algorithm 3, an approach they called *learning with exploration*.

In particular, as our neural network uses a cross-entropy loss, and not the perceptron loss used in Goldberg and Nivre (2013), updates are performed even when the prediction is correct. When $p = 0$, the algorithm acts identically to a static oracle trainer, as the parser always follows the gold transition. When the set of actions predicted by the oracle has more than one element, the best scoring element among them is chosen as the reference

function TRAINONESENTENCE($s, \boldsymbol{\theta}, p, o$)
  $C \leftarrow$ INITIAL(s)
  while C is not a final configuration **do**
    $A \leftarrow o(C, s)$ ▷ set of best actions
    $\hat{a} \leftarrow \text{argmax}_a\, f_{\boldsymbol{\theta}}(C)_a$
    if $\hat{a} \in A$ **then**
      $t \leftarrow \hat{a}$ ▷ t: target
    else
      $t \leftarrow \text{argmax}_{a \in A}\, f_{\boldsymbol{\theta}}(C)_a$
    $\boldsymbol{\theta} \leftarrow$ UPDATE($\boldsymbol{\theta}, C, t$) ▷ backprop
    if RANDOM() $< p$ **then**
      $C \leftarrow \hat{a}(C)$ ▷ Follow prediction
    else
      $C \leftarrow t(C)$ ▷ Follow best action
  return $\boldsymbol{\theta}$

Figure 5: Online training for a single annotated sentence s, using an oracle function o.

action to update the parameters of the neural network.

4 A Dynamic Oracle for Transition-Based Parsing

This section introduces a dynamic oracle algorithm for the parsing model presented in the previous 2 sections, that is the function o used in the algorithm in Figure 5.

The dynamic oracle must minimize a cost function $\mathcal{L}(c; t, T)$ computing the cost of applying transition t in configuration c, with respect to a gold parse T. As is shown by Goldberg and Nivre (2013), the oracle's correctness depends on the cost function. A correct dynamic oracle o will have the following general formulation:

$$o(c, T) = \{t | \mathcal{L}(c; t, T) = \min_{t'} \mathcal{L}(c; t', T)\} \quad (1)$$

The correctness of the oracle is not necessary to improve training. The oracle needs only to be good enough (Daumé et al., 2009), which is confirmed by empirical results (Straka et al., 2015). Goldberg and Nivre (2013) identified *arc-decomposability*, a powerful property of certain dependency parsing transition systems for which we can easily derive correct efficient oracles. When this property holds, we can infer whether a tree is reachable from the reachability of individual arcs. This simplifies the calculation of each transition cost. We rely on an analogue property we call **constituent decomposition**. A

set of constituents is **tree-consistent** if it is a subset of a set corresponding to a well-formed tree. A phrase structure transition system is constituent-decomposable iff *for any configuration C and any tree-consistent set of constituents γ, if every constituent in γ is reachable from C, then the whole set is reachable from C* (constituent reachability will be formally defined in Section 4.1).

The following subsections are structured as follows. First of all, we present a cost function (Section 4.1). Then, we derive a correct dynamic oracle algorithm for an ideal case where we assume that there is no temporary symbols in the grammar (Section 4.2). Finally, we present some heuristics to define a dynamic oracle for the general case (Section 4.3).

4.1 Cost Function

The cost function we use ignores the lexicalization of the symbols. For the sake of simplicity, we momentarily leave apart the headedness of the binary reductions (until the last paragraph of Section 4) and assume a unique binary REDUCE action.

For the purpose of defining a cost function for transitions, we adopt a representation of trees as sets of constituents. For example, `(S (NP (D the) (N cat)) (VP (V sleeps)))` corresponds to the set $\{(\text{S}, 0, 3), (\text{NP}, 0, 2), (\text{VP}, 2, 3)\}$. As is shown in Figure 2, every reduction action (unary or binary) adds a new constituent to the set γ of already predicted constituents, which was introduced in Section 2. We define the cost of a predicted set of constituents $\hat{\gamma}$ with respect to a gold set γ^* as the number of constituents in γ^* which are not in $\hat{\gamma}$ penalized by the number of predicted unary constituents which are not in the gold set:

$$\mathcal{L}_r(\hat{\gamma}, \gamma^*) = |\gamma^* - \hat{\gamma}| + |\{(X, i, i+1) \in \hat{\gamma} | (X, i, i+1) \notin \gamma^*\}| \quad (2)$$

The first term penalizes false negatives and the second one penalizes unary false positives. The number of binary constituents in γ^* and $\hat{\gamma}$ depends only on the sentence length n, thus binary false positives are implicitly taken into account by the fist term.

The cost of a transition and that of a configuration are based on **constituent reachability**. The relation $C \vdash C'$ holds iff C' can be deduced from C by performing a transition. Let $\vdash^*$ denote the reflexive transitive closure of $\vdash$. A set of constituents γ (possibly a singleton) is reachable from a configuration C iff there is a configuration $C' = \langle j, S, b, \gamma' \rangle$ such that $C \vdash^* C'$ and $\gamma \subseteq \gamma'$, which we write $C \rightsquigarrow \gamma$.

Then, the cost of an action t for a configuration C is the cost difference between the best tree reachable from $t(C)$ and the best tree reachable from C:

$$\mathcal{L}_r(t; C, \gamma^*) = \min_{\gamma : t(C) \rightsquigarrow \gamma} \mathcal{L}(\gamma, \gamma^*) - \min_{\gamma : C \rightsquigarrow \gamma} \mathcal{L}(\gamma, \gamma^*)$$

This cost function is easily decomposable (as a sum of costs of transitions) whereas F1 measure is not.

By definition, for each configuration, there is at least one transition with cost 0 with respect to the gold parse. Otherwise, it would entail that there is a tree reachable from C but unreachable from $t(C)$, for any t. Therefore, we reformulate equation 1:

$$o(C, \gamma^*) = \{t | \mathcal{L}_r(C; t, \gamma^*) = 0\} \quad (3)$$

In the transition system, the grammar is left implicit: any reduction is allowed (even if the corresponding grammar rule has never been seen in the training corpus). However, due to the introduction of temporary symbols during binarization, there are constraints to ensure that any derivation corresponds to a well-formed unbinarized tree. These constraints make it difficult to test the reachability of constituents. For this reason, we instantiate two transition systems. We call SR-TMP the transition system in Figure 2 which enforces the constraints in Table 1, and SR-BIN, the same transition system without any of such constraints. SR-BIN assumes an idealized case where the grammar contains no temporary symbols, whereas SR-TMP is the actual system we use in our experiments.

4.2 A Correct Oracle for SR-BIN Transition System

SR-BIN transition system provides no guarantees that predicted trees are unbinarisable. The only condition for a binary reduction to be allowed is that the stack contains at least two symbols. If so, any non-terminal in the grammar could be used. In such a case, we can define a simple necessary and sufficient condition for constituent reachability.

Constituent reachability Let γ^* be a tree-consistent constituent set, and $C = \langle j, S, b, \gamma \rangle$ a

parsing configuration, such that:

$$S = (X_1, i_0, i_1) \dots (X_p, i_{p-1}, i)|(A, i, k)|(B, k, j)$$

A binary constituent (X, m, n) is reachable iff it satisfies one of the three following properties :

1. $(X, m, n) \in \gamma$
2. $j < m < n$
3. $m \in \{i_0, \dots i_{p-1}, i, k\}, n \geq j$ and $(m, n) \neq (k, j)$

The first two cases are trivial and correspond respectively to a constituent already constructed and to a constituent spanning words which are still in the buffer.

In the third case, (X, m, n) can be constructed by performing $n - j$ times the transitions SHIFT and GHOST-REDUCE (or REDUCE-UNARY), and then a sequence of binary reductions ended by an X reduction. Note that as the index j in the configuration is non-decreasing during a derivation, the constituents whose span end is inferior to j are not reachable if they are not already constructed. For a unary constituent, the condition for reachability is straightforward: a constituent $(X, i-1, i)$ is reachable from configuration $C = \langle j, S, b, \gamma \rangle$ iff $(X, i-1, i) \in \gamma$ or $i > j$ or $i = j \wedge b = \top$.

Constituent decomposability SR-BIN is constituent decomposable. In this paragraph, we give some intuition about why this holds. Reasoning by contradiction, let's assume that every constituent of a tree-consistent set γ^* is reachable from $C = \langle j, S|(A, i, k)|(B, k, j), b, \gamma \rangle$ and that γ^* is not reachable (contraposition). This entails that at some point during a derivation, there is no possible transition which maintains reachability for all constituents of γ^*. Let's assume C is in such a case. If some constituent of γ^* is reachable from C, but not from SHIFT(C), its span must have the form (m, j), where $m \leq i$. If some constituent of γ^* is reachable from C, but not from REDUCE(X)(C), for any label X, its span must have the form (k, n), where $n > j$. If both conditions hold, γ^* contains incompatible constituents (crossing brackets), which contradicts the assumption that γ^* is tree-consistent.

Computing the cost of a transition The conditions on constituent reachability makes it easy to compute the cost of a transition t for a given configuration $C = \langle j, S|(A, i, k)|(B, k, j), b, \gamma \rangle$ and a gold set γ^*:

```
1: function O(⟨j, S|(A, i, k)|(B, k, j), b, γ⟩, γ*)
2:     if b = ⊤ then                    ▷ Last action was SHIFT
3:         if (X, j − 1, j) ∈ γ* then
4:             return {REDUCEUNARY(X)}
5:         else
6:             return {GHOSTREDUCE}
7:     if ∃n > j, (X, k, n) ∈ γ* then
8:         return {SHIFT}
9:     if (X, i, j) ∈ γ* then
10:        return {REDUCE(X)}
11:    if ∃m < i, (X, m, j) ∈ γ* then
12:        return {REDUCE(Y), ∀Y}
13:    return {a ∈ A|a is a possible action}
```

Figure 6: Oracle algorithm for SR-BIN.

- The cost of a SHIFT is the number of constituents not in γ, reachable from C and whose span ends in j.
- The cost of a binary reduction REDUCE(X) is a sum of two terms. The first one is the number of constituents of γ^* whose span has the form (k, n) with $n > j$. These are no longer compatible with (X, i, j) in a tree. The second one is one if $(Y, i, j) \in \gamma^*$ and $Y \neq X$ and zero otherwise. It is the cost of mislabelling a constituent with a gold span.
- The cost of a unary reduction or that of a ghost reduction can be computed straightforwardly by looking at the gold set of constituents.

We present in Figure 6 an oracle algorithm derived from these observations.

4.3 A Heuristic-based Dynamic Oracle for SR-TMP transition system

The conditions for constituent reachability for SR-BIN do not hold any longer for SR-TMP. In particular, constituent reachability depends crucially on the distinction between temporary and non-temporary symbols. The algorithm in Figure 6 is not correct for this transition system. In Figure 7, we give an illustration of a prototypical case in which the algorithm in Figure 6 will fail. The constituent (C:, i, j) is in the gold set of constituents and could be constructed with REDUCE(C:). The third symbol on the stack being temporary symbol D:, the reduction to a temporary symbol will jeopardize the reachability of (C, m, j) because reduc-

tions are not possible when the two symbols at the top of the stack are temporary symbols. The best course of action is then a reduction to any non-temporary symbol, so as to keep (C, m, j) reachable. Note that in this case, the cost of REDUCE(C:) cannot be smaller than that of a single mislabelled constituent.

In fact, this example shows that the constraints inherent to SR-TMP makes it non constituent-decomposable. In the example in Figure 7, both constituents in the set $\{(\mathrm{C}, m, j), (\mathrm{C}{:}, i, j)\}$, a tree-consistent constituent set, is reachable. However, the whole set is not reachable, as REDUCE(C:) would make (C, m, j) not reachable.

In dependency parsing, several exact dynamic oracles have been proposed for non arc-decomposable transition systems (Goldberg et al., 2014), including systems for non-projective parsing (Gómez-Rodríguez et al., 2014). These oracles rely on tabular methods to compute the cost of transitions and have (high-degree) polynomial worst case running time. Instead, to avoid resorting to more computationally expensive exact methods, we adapt the algorithm in Figure 6 to the constraints involving temporary symbols using the following heuristics:

- If the standard oracle predicts a reduction, make sure to choose its label so that every reachable constituent $(X, m, j) \in \gamma^*$ $(m < i)$ is still reachable after the transition. Practically, if such constituent exists and if the third symbol on the stack is a temporary symbol, then do not predict a temporary symbol.
- When reductions to both temporary symbols and non-temporary symbols have cost zero, only predict temporary symbols. This should not harm training and improve precision for the unbinarized tree, as any non temporary symbol in the binarized tree corresponds to a constituent in the n-ary tree.

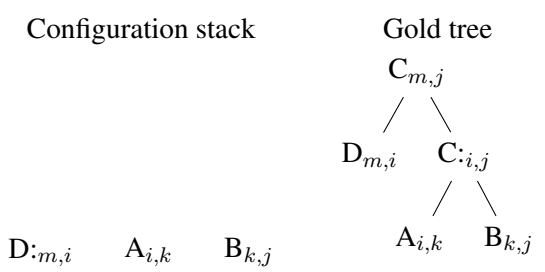


Figure 7: Problematic case. Due to the temporary symbol constraints enforced by SR-TMP, the algorithm in Figure 6 will fail on this example.

Dev F1 (EVALB)		Decoding	toks/sec
static (this work)	88.6	greedy	
dynamic (this work)	89.0	greedy	
Test F1 (EVALB)			
Hall et al. (2014)	89.2	CKY	12
Berkeley (Petrov et al., 2006)	90.1	CKY	169
Durrett and Klein (2015)†	91.1	CKY	-
Zhu et al. (2013)†	91.3	beam=16	1,290
Crabbé (2015)	90.0	beam=8	2,150
Sagae and Lavie (2006)	85.1	greedy	-
static (this work)	88.0	greedy	3,820◇
dynamic (this work)	88.6	greedy	3,950◇

Table 3: Results on the Penn Treebank (Marcus et al., 1993). † use clusters or word vectors learned on unannotated data. ◇ different architecture (2.3Ghz Intel), single processor.

Head choice In some cases, namely when reducing two non-temporary symbols to a new constituent (X, i, j), the oracle must determine the head position in the reduction (REDUCE-RIGHT or REDUCE-LEFT). We used the following heuristic: if (X, i, j) is in the gold set, choose the same head position, otherwise, predict both RR(X) and RL(X) to keep the non-determinism.

5 Experiments

We conducted parsing experiments to evaluate our proposal. We compare two experimental settings. In the 'static' setting, the parser is trained only on gold configurations; in the 'dynamic' setting, we use the dynamic oracle and the training method in Figure 5 to explore non-gold configurations. We used both the SPMRL dataset (Seddah et al., 2013) in the 'predicted tag' scenario, and the Penn Treebank (Marcus et al., 1993), to compare our proposal to existing systems. The tags and morphological attributes were predicted using Marmot (Mueller et al., 2013), by 10-fold jackknifing for the train and development sets. For the SPMRL dataset, the head annotation was carried out with the procedures described in Crabbé

Number of possible values	≤ 8	≤ 32	> 32
Dimensions for embedding	4	8	16

Table 4: Size of morphological attributes embeddings.

		Arabic	Basque	French	German	Hebrew	Hungarian	Korean	Polish	Swedish	Avg
	Decoding				Development F1 (EVALBSPMRL)						
Durrett and Klein (2015)†	CKY	80.68	84.37	80.65	**85.25**	89.37	**89.46**	82.35	92.10	**77.93**	84.68
Crabbé (2015)	beam=8	**81.25**	84.01	**80.87**	84.08	**90.69**	88.27	83.09	**92.78**	77.87	84.77
static (this work)	greedy	80.25	84.29	79.87	83.99	89.78	88.44	84.98	92.38	76.63	84.51
dynamic (this work)	greedy	80.94	**85.17**	80.31	84.61	90.20	88.70	**85.46**	92.57	77.87	**85.09**
					Test F1 (EVALBSPMRL)						
Björkelund et al. (2014)†		81.32*	**88.24**	**82.53**	**81.66**	**89.80**	**91.72**	83.81	90.50	**85.50**	**86.12**
Berkeley (Petrov et al., 2006)	CKY	79.19	70.50	80.38	78.30	86.96	81.62	71.42	79.23	79.18	78.53
Berkeley-Tags	CKY	78.66	74.74	79.76	78.28	85.42	85.22	78.56	86.75	80.64	80.89
Durrett and Klein (2015)†	CKY	80.24	85.41	**81.25**	**80.95**	88.61	**90.66**	82.23	**92.97**	**83.45**	**85.09**
Crabbé (2015)	beam=8	**81.31**	84.94	80.84	79.26	**89.65**	90.14	82.65	92.66	83.24	84.97
Fernández-González and Martins (2015)		-	85.90	78.75	78.66	88.97	88.16	79.28	91.20	82.80	(84.22)
static (this work)	greedy	79.77	85.91	79.62	79.20	88.64	90.54	84.53	92.69	81.45	84.71
dynamic (this work)	greedy	80.71	**86.24**	79.91	80.15	88.69	90.51	**85.10**	**92.96**	81.74	**85.11**
dynamic (this work)	beam=2	81.14	86.45	80.32	80.68	89.06	90.74	85.17	**93.15**	82.65	85.48
dynamic (this work)	beam=4	81.59	86.45	80.48	80.69	89.18	90.73	85.31	93.13	82.77	85.59
dynamic (this work)	beam=8	**81.80**	**86.48**	80.56	80.74	89.24	**90.76**	**85.33**	93.13	82.80	**85.64**

Table 2: Results on development and test corpora. Metrics are provided by `evalb_spmrl` with `spmrl.prm` parameters (`http://www.spmrl.org/spmrl2013-sharedtask.html`). † use clusters or word vectors learned on unannotated data. * Björkelund et al. (2013).

(2015), using the alignment between dependency treebanks and constituent treebanks. For English, we used Collins' head annotation rules (Collins, 2003). Our system is entirely supervised and uses no external data. Every embedding was initialised randomly (uniformly) in the interval $[-0.01, 0.01]$. Word embeddings have 32 dimensions, tags and non-terminal embeddings have 16 dimensions. The dimensions of the morphological attributes depend on the number of values they can have (Table 4). The hidden layer has 512 units.[4]

For the 'dynamic' setting, we trained every other k sentence with the dynamic oracle and the other sentences with the static oracle. This method, used by Straka et al. (2015), allows for high values of p, without slowing or preventing convergence. We used several hyperparameters combinations (see Table 5 of Annex A). For each language, we present the model with the combination which maximizes the developement set F-score. We used Averaged Stochastic Gradient Descent (Polyak and Juditsky, 1992) to minimize the negative log likelihood of the training examples. We shuffled the sentences in the training set before each iteration.

Results Results for English are shown in Table 3. The use of the dynamic oracle improves F-score by 0.4 on the development set and 0.6 on the test set. The resulting parser, despite using greedy decoding and no additional data, is quite accurate. For example, it compares well with Hall et al. (2014)'s span based model and is much faster.

For the SPMRL dataset, we report results on the development sets and test sets in Table 2. The metrics take punctuation and unparsed sentences into account (Seddah et al., 2013). We compare our results with the SPMRL shared task baselines (Seddah et al., 2013) and several other parsing models. The model of Björkelund et al. (2014) obtained the best results on this dataset. It is based on a product grammar and a discriminative reranker, together with morphological features and word clusters learned on unannotated data. Durrett and Klein (2015) use a neural CRF based on CKY decoding algorithm, with word embeddings pretrained on unannotated data. Fernández-González and Martins (2015) use a parsing-as-reduction approach, based on a dependency parser with a label set rich enough to reconstruct constituent trees from dependency trees. Finally, Crabbé (2015) uses a structured perceptron with rich features and beam-search decoding. Both Crabbé (2015) and Björkelund et al. (2014) use MARMOT-predicted morphological tags (Mueller et al., 2013), as is done in our experiments.

Our results show that, despite using a very simple greedy inference and being strictly supervised, our base model (static oracle training) is competitive with the best single parsers on this dataset.

[4]We did not tune these hyperparameters for each language. Instead, we chose a set of hyperparameters which achieved a tradeoff between training time and model accuracy. The effect of the morphological features and their dimensionality are left to future work.

We hypothesize that these surprising results come both from the neural scoring model and the morphological attribute embeddings (especially for Basque, Hebrew, Polish and Swedish). We did not test these hypotheses systematically and leave this investigation for future work.

Furthermore, we observe that the dynamic oracle improves training by up to 0.6 F-score (averaged over all languages). The improvement depends on the language. For example, Swedish, Arabic, Basque and German are the languages with the most important improvement. In terms of absolute score, the parser also achieves very good results on Korean and Basque, and even outperforms Björkelund et al. (2014)'s reranker on Korean.

Combined effect of beam and dynamic oracle Although initially, dynamic oracle training was designed to improve parsing without relying on more complex search methods (Goldberg and Nivre, 2012), we tested the combined effects of dynamic oracle training and beam search decoding. In Table 2, we provide results for beam decoding with the already trained local models in the 'dynamic' setting. The transition from greedy search to a beam of size two brings an improvement comparable to that of the dynamic oracle. Further increase in beam size does not seem to have any noticeable effect, except for Arabic. These results show that effects of the dynamic oracle and beam decoding are complementary and suggest that a good tradeoff between speed and accuracy is already achieved in a greedy setting or with a very small beam size

6 Conclusion

We have described a dynamic oracle for constituent parsing. Experiments show that training a parser against this oracle leads to an improvement in accuracy over a static oracle. Together with morphological features, we obtain a greedy parser as accurate as state-of-the-art (non reranking) parsers for morphologically-rich languages.

Acknowledgments

We thank the anonymous reviewers, along with Héctor Martínez Alonso and Olga Seminck for valuable suggestions to improve prior versions of this article.

References

Anders Björkelund, Özlem Çetinoğlu, Richárd Farkas, Thomas Mueller, and Wolfgang Seeker. 2013. (re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 135–145, Seattle, Washington, USA, October. Association for Computational Linguistics.

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

Léon Bottou. 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186. Physica-Verlag HD.

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

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Comput. Linguist.*, 29(4):589–637, December.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Benoit Crabbé. 2015. Multilingual discriminative lexicalized phrase structure parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1847–1856, Lisbon, Portugal, September. Association for Computational Linguistics.

Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Hal Daumé III, John Langford, and Daniel Marcu. 2006. Searn in practice.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the Association for Computational Linguistics*, Beijing, China, July. Association for Computational Linguistics.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the*

53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1523–1533, Beijing, China, July. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of COLING 2012*, pages 959–976, Mumbai, India, December. The COLING 2012 Organizing Committee.

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

Yoav Goldberg, Francesco Sartorio, and Giorgio Satta. 2014. A tabular method for dynamic oracles in transition-based parsing. *TACL*, 2:119–130.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2015. An efficient dynamic oracle for unrestricted non-projective parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 256–261, Beijing, China, July. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, Francesco Sartorio, and Giorgio Satta. 2014. A polynomial-time dynamic oracle for non-projective dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–927, Doha, Qatar, October. Association for Computational Linguistics.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Baltimore, Maryland, June. Association for Computational Linguistics.

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

Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1030–1035, Denver, Colorado, May–June. Association for Computational Linguistics.

Thomas Mueller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.

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

B. T. Polyak and A. B. Juditsky. 1992. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4):838–855, July.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132. Association for Computational Linguistics.

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

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA, October. Association for Computational Linguistics.

Milan Straka, Jan Hajič, Jana Straková, and Jan Hajič jr. 2015. Parsing universal dependency treebanks using neural networks and search-based oracle. In *Proceedings of Fourteenth International Workshop on Treebanks and Linguistic Theories (TLT 14)*, December.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*

(Volume 1: Long Papers), pages 1169–1179, Beijing, China, July. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies*, IWPT '09, pages 162–171, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

A Supplementary Material

'static' and 'dynamic' setting			'dynamic' setting	
learning rate	α	iterations	k	p
$\{0.01, 0.02\}$	$\{0, 10^{-6}\}$	$[1, 24]$	$\{8, 16\}$	$\{0.5, 0.9\}$

Table 5: Hyperparameters. α is the decrease constant used for the learning rate (Bottou, 2010).

$\mathbf{s_0.c_t}$	$\mathbf{s_0.w_t.tag}$	$\mathbf{s_0.w_t.form}$	$\mathbf{q_1.tag}$
$\mathbf{s_0.c_l}$	$\mathbf{s_0.w_l.tag}$	$\mathbf{s_0.w_l.form}$	$\mathbf{q_2.tag}$
$\mathbf{s_0.c_r}$	$\mathbf{s_0.w_r.tag}$	$\mathbf{s_0.w_r.form}$	$\mathbf{q_3.tag}$
$\mathbf{s_1.c_t}$	$\mathbf{s_1.w_t.tag}$	$\mathbf{s_1.w_t.form}$	$\mathbf{q_4.tag}$
$\mathbf{s_1.c_l}$	$\mathbf{s_1.w_l.tag}$	$\mathbf{s_1.w_l.form}$	$\mathbf{q_1.form}$
$\mathbf{s_1.c_r}$	$\mathbf{s_1.w_r.tag}$	$\mathbf{s_1.w_r.form}$	$\mathbf{q_2.form}$
$\mathbf{s_2.c_t}$	$\mathbf{s_2.w_t.tag}$	$\mathbf{s_2.w_t.form}$	$\mathbf{q_3.form}$
			$\mathbf{q_4.form}$
$\mathbf{s_0.w_t.m} \forall m \in M$		$\mathbf{q_0.m} \forall m \in M$	
$\mathbf{s_1.w_t.m} \forall m \in M$		$\mathbf{q_1.m} \forall m \in M$	

Table 6: These templates specify a list of addresses in a configuration. The input of the neural network is the instanciation of each address by a discrete typed symbol. Each $\mathbf{v_i}$ (Section 2) is the embedding of the i^{th} instantiated symbol of this list. M is the set of all available morphological attributes for a given language. We use the following notations (cf Figure 4): $\mathbf{s}_i$ is the i^{th} item in the stack, $\mathbf{c}$ denotes non-terminals, **t**op, **l**eft and **r**ight, indicate the position of an element in the subtree. Finally, $\mathbf{w}$ and $\mathbf{q}$ are respectively stack and buffer tokens.

Literal and Metaphorical Senses in Compositional Distributional Semantic Models

E. Darío Gutiérrez[1] **Ekaterina Shutova**[2] **Tyler Marghetis**[3] **Benjamin K. Bergen**[4]
[1] University of California San Diego
[2] University of Cambridge
[3] Indiana University Bloomington
edg@icsi.berkeley.edu tmarghet@cogsci.ucsd.edu
es407@cam.ac.uk bkbergen@ucsd.edu

Abstract

Metaphorical expressions are pervasive in natural language and pose a substantial challenge for computational semantics. The inherent compositionality of metaphor makes it an important test case for compositional distributional semantic models (CDSMs). This paper is the first to investigate whether metaphorical composition warrants a distinct treatment in the CDSM framework. We propose a method to learn metaphors as linear transformations in a vector space and find that, across a variety of semantic domains, explicitly modeling metaphor improves the resulting semantic representations. We then use these representations in a metaphor identification task, achieving a high performance of 0.82 in terms of F-score.

1 Introduction

An extensive body of behavioral and corpus-linguistic studies suggests that metaphors are pervasive in everyday language (Cameron, 2003; Steen et al., 2010) and play an important role in how humans define and understand the world. According to Conceptual Metaphor Theory (CMT) (Lakoff and Johnson, 1981), individual metaphorical expressions, or *linguistic metaphors* (LMs), are instantiations of broader generalizations referred to as *conceptual metaphors* (CMs). For example, the phrases *half-baked idea*, *food for thought*, and *spoon-fed information* are LMs that instantiate the CM IDEAS ARE FOOD. These phrases reflect a mapping from the *source domain* of FOOD to the *target domain* of IDEAS (Lakoff, 1989). Two central claims of the CMT are that this mapping is systematic, in the sense that it consists of a fixed set of ontological correspondences, such as *thinking is preparing*, *communication is feeding*, *understanding is digestion*; and that this mapping can be productively extended to produce novel LMs that obey these correspondences.

Recent years have seen the rise of statistical techniques for metaphor detection. Several of these techniques leverage distributional statistics and vector-space models of meaning to classify utterances as literal or metaphorical (Utsumi, 2006; Shutova et al., 2010; Hovy et al., 2013; Tsvetkov et al., 2014). An important insight of these studies is that metaphorical meaning is not merely a property of individual words, but rather arises through cross-domain composition. The meaning of *sweet*, for instance, is not intrinsically metaphorical. Yet this word may exhibit a range of metaphorical meanings—e.g., *sweet dreams, sweet person, sweet victory*–that are created through the interplay of source and target domains. If metaphor is compositional, how do we represent it, and how can we use it in a compositional framework for meaning?

Compositional distributional semantic models (CDSMs) provide a compact model of compositionality that produces vector representations of phrases while avoiding the sparsity and storage issues associated with storing vectors for each phrase in a language explicitly. One of the most popular CDSM frameworks (Baroni and Zamparelli, 2010; Guevara, 2010; Coecke et al., 2010) represents nouns as vectors, adjectives as matrices that act on the noun vectors, and transitive verbs as third-order tensors that act on noun or noun phrase vectors. The meaning of a phrase is then derived by composing these lexical representations. The vast majority of such models build a single representation for all senses of a word, collapsing distinct senses together. One exception is the work of Kartsaklis and Sadrzadeh (2013a), who investigated homonymy, in which lexical items

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 183–193,
Berlin, Germany, August 7-12, 2016.

have identical form but unrelated meanings (e.g., *bank*). They found that deriving verb tensors from all instances of a homonymous form (as compared to training a separate tensor for each distinct sense) loses information and degrades the resultant phrase vector representations. To the best of our knowledge, there has not yet been a study of regular polysemy (i.e. metaphorical or metonymic sense distinctions) in the context of compositional distributional semantics. Yet, due to systematicity in metaphorical cross-domain mappings, there are likely to be systematic contextual sense distinctions that can be captured by a CDSM, improving the resulting semantic representations.

In this paper, we investigate whether metaphor, as a case of regular polysemy, warrants distinct treatment under a compositional distributional semantic framework. We propose a new approach to CDSMs, in which metaphorical meanings are distinct but structurally related to literal meanings. We then extend the generalizability of our approach by proposing a method to automatically learn metaphorical mappings as linear transformations in a CDSM. We focus on modeling adjective senses and evaluate our methods on a new data set of 8592 adjective-noun pairs annotated for metaphoricity, which we will make publicly available. Finally, we apply our models to classify unseen adjective-noun (AN) phrases as literal or metaphorical and obtain state-of-the-art performance in the metaphor identification task.

2 Background & Related Work

Metaphors as Morphisms. The idea of metaphor as a systematic mapping has been formalized in the framework of category theory (Goguen, 1999; Kuhn and Frank, 1991). In category theory, morphisms are transformations from one object to another that preserve some essential structure of the original object. Category theory provides a general formalism for analyzing relationships as morphisms in a wide range of systems (see Spivak (2014)). Category theory has been used to formalize the CM hypothesis with applications to user interfaces, poetry, and information visualization (Kuhn and Frank, 1991; Goguen and Harrell, 2010; Goguen and Harrell, 2005). Although these formal treatments of metaphors as morphisms are rigorous and well-formalized, they have been applied at a relatively limited scale. This is because this work does not suggest a straightforward and data-driven way to quantify semantic domains or morphisms, but rather focuses on the transformations and relations between semantic domains and morphisms, assuming some appropriate quantification has already been established. In contrast, our methods can learn representations of source-target domain mappings from corpus data, and so are inherently more scalable.

Compositional DSMs. Similar issues arose in modeling compositional semantics. Formal semantics has dealt with compositional meaning for decades, by using mathematical structures from abstract algebra, logic, and category theory (Montague, 1970; Partee, 1994; Lambek, 1999). However, formal semantics requires manual crafting of features. The central insight of CDSMs is to model the composition of words as algebraic operations on their vector representations, as provided by a conventional DSM (Mitchell and Lapata, 2008). Guevara (2010) and Baroni and Zamparelli (2010) were the first to treat adjectives and verbs differently from nouns. In their models, adjectives are represented by matrices that act on noun vectors. Adjective matrices can be learned using regression techniques. Other CDSMs have also been proposed and successfully applied to tasks such as sentiment analysis and paraphrase (Socher et al., 2011; Socher et al., 2012; Tsubaki et al., 2013; Turney, 2013).

Handling Polysemy in CDSMs. Several researchers argue that terms with ambiguous senses can be handled by DSMs without any recourse to additional disambiguation steps, as long as contextual information is available (Boleda et al., 2012; Erk and Padó, 2010; Pantel and Lin, 2002; Schütze, 1998; Tsubaki et al., 2013). Baroni et al. (2014) conjecture that CDSMs might largely avoid problems handling adjectives with multiple senses because the matrices for adjectives implicitly incorporate contextual information. However, they do draw a distinction between two ways in which the meaning of a term can vary. Continuous *polysemy*—the subtle and continuous variations in meaning resulting from the different contexts in which a word appears—is relatively tractable, in their opinion. This contrasts with discrete *homonymy*—the association of a single term with completely independent meanings (e.g., *light house* vs. *light work*). Baroni et al. concede that homonymy is more difficult to handle in

CDSMs. Unfortunately, they do not propose a definite way to determine whether any given variation in meaning is polysemy or homonymy, and offer no account of regular polysemy (i.e., metaphor and metonymy) or whether it would pose similar problems as homonymy for CDSMs.

To handle the problematic case of homonymy, Kartsaklis and Sadrzadeh (2013b) adapt a clustering technique to disambiguate the senses of verbs, and then train separate tensors for each sense, using the previously mentioned CDSM framework of Coecke et al. (2010). They found that prior disambiguation resulted in semantic similarity measures that correlated more closely with human judgments.

In principle, metaphor, as a type of regular polysemy, is different from the sort of semantic ambiguity described above. General ambiguity or vagueness in meaning (e.g. *bright light* vs *bright color*) is generally context-dependent in an unsystematic manner. In contrast, in regular polysemy meaning transfer happens in a systematic way (e.g. *bright light* vs. *bright idea*), which can be explicitly modeled within a CDSM. The above CDSMs provide no account of such systematic polysemy, which is the gap this paper aims to fill.

Computational Work on Metaphor. There is now an extensive literature on statistical approaches to metaphor detection. The investigated methods include clustering (Birke and Sarkar, 2006; Shutova et al., 2010; Li and Sporleder, 2010); topic modeling (Bethard et al., 2009; Li et al., 2010; Heintz et al., 2013); topical structure and imageability analysis (Strzalkowski et al., 2013); semantic similarity graphs (Sporleder and Li, 2009), and feature-based classifiers (Gedigian et al., 2006; Li and Sporleder, 2009; Turney et al., 2011; Dunn, 2013a; Dunn, 2013b; Hovy et al., 2013; Mohler et al., 2013; Neuman et al., 2013; Tsvetkov et al., 2013; Tsvetkov et al., 2014). We refer readers to the survey by Shutova (2015) for a more thorough review.

Most relevant to the present work are approaches that attempt to identify whether adjective-noun phrases are metaphorical or literal. Krishnakumaran and Zhu (2007) use AN co-occurrence counts and WordNet hyponym/hypernym relations for this task. If the noun and its hyponyms/hypernyms do not occur frequently with the given adjective, then the AN phrase is labeled as metaphorical. Krishnakumaran and Zhu's system achieves a precision of 0.67. Turney et al. (2011) classify verb and adjective phrases based on their level of concreteness or abstractness in relation to the noun they appear with. They learn concreteness rankings for words automatically (starting from a set of examples) and then search for expressions where a concrete adjective or verb is used with an abstract noun (e.g., *dark humor* is tagged as a metaphor; *dark hair* is not). They measure performance on a set of 100 phrases involving one of five adjectives, attaining an average accuracy of 0.79. Tsvetkov et al. (2014) train a random-forest classifier using several features, including abstractness and imageability rankings, WordNet supersenses, and DSM vectors. They report an accuracy of 0.81 on the Turney et al. (2011) AN phrase set. They also introduce a new set of 200 AN phrases, on which they measure an F-score of 0.85.

3 Experimental Data

Corpus. We trained our DSMs from a corpus of 4.58 billion tokens. Our corpus construction procedure is modeled on that of Baroni and Zamparelli (2010). The corpus consisted of a 2011 dump of English Wikipedia, the UKWaC (Baroni et al., 2009), the BNC (BNC Consortium, 2007), and the English Gigaword corpus (Graff et al., 2003). The corpus was tokenized, lemmatized, and POS-tagged using the NLTK toolkit (Bird and Loper, 2004) for Python.

Metaphor Annotations. We created an annotated dataset of 8592 AN phrases (3991 literal, 4601 metaphorical). Our choice of adjectives was inspired by the test set of Tsvetkov et al. (2014), though our annotated dataset is considerably larger. We focused on 23 adjectives that can have both metaphorical and literal senses, and which function as source-domain words in relatively productive CMs: TEMPERATURE (*cold*, *heated*, *icy*, *warm*), LIGHT (*bright*, *brilliant*, *dim*), TEXTURE (*rough*, *smooth*, *soft*); SUBSTANCE (*dense*, *heavy*, *solid*), CLARITY (*clean*, *clear*, *murky*), TASTE (*bitter*, *sour*, *sweet*), STRENGTH (*strong*, *weak*), and DEPTH (*deep*, *shallow*). We extracted all AN phrases involving these adjectives that occur in our corpus at least 10 times. We filtered out all phrases that require wider context to establish their meaning or metaphoricity—e.g., *bright side*, *weak point*.

The remaining phrases were annotated using a

procedure based on Shutova et al. (2010). Annotators were encouraged to rely on their own intuition of metaphor, but were provided with the following guidance:

- For each phrase, establish the meaning of the adjective in the context of the phrase.
- Try to imagine a more basic meaning of this adjective in other contexts. Basic meanings tend to be: more concrete; related to embodied actions/perceptions/sensations; more precise; historically older/more "original".
- If you can establish a basic meaning distinct from the meaning of the adjective in this context, it is likely to be used metaphorically.

If requested, a randomly sampled sentence from the corpus that contained the phrase in question was also provided. The annotation was performed by one of the authors. The author's annotations were compared against those of a university graduate native English-speaking volunteer who was not involved in the research, on a sample of 500 phrases. Interannotator reliability (Cohen, 1960; Fleiss et al., 1969) was $\kappa = 0.80$ ($SE = .02$). Our annotated data set is publicly available at `http://bit.ly/1TQ5czN`

4 Representing Metaphorical Senses in a Compositional DSM

In this section we test whether separate treatment of literal and metaphorical senses is justified in a CDSM framework. In that case, training adjective matrix representations on literal and metaphorical subsets separately may result in systematically improved phrase vector representations, despite each matrix making use of fewer training examples.

4.1 Method

Our goal is to learn accurate vector representations for unseen adjective-noun (AN) phrases, where adjectives can take on metaphorical or literal senses. Our models build off the CDSM framework of Baroni and Zamparelli (2010), as extended by Li et al. (2014). Each adjective a is treated as a linear map from nouns to AN phrases:

$$\mathbf{p} = \mathbf{A}_a \mathbf{n},$$

where $\mathbf{p}$ is a vector for the phrase, $\mathbf{n}$ is a vector for the noun, and $\mathbf{A}_a$ is a matrix for the adjective.

Contextual Variation Model. The traditional representations do not account for the differences in meaning of an adjective in literal vs metaphorical phrases. Their assumption is that the contextual variations in meaning that are encoded by literal and metaphorical senses may be subtle enough that they can be handled by a single catch-all matrix per adjective, $\mathbf{A}_{\text{BOTH}(a)}$. In this model, every phrase i can be represented by

$$\mathbf{p}_i = \mathbf{A}_{\text{BOTH}(a)} \mathbf{n}_i \quad (1)$$

regardless of whether a is used metaphorically or literally in i. This model has the advantage of simplicity and requires no information about whether an adjective is being used literally or metaphorically. In fact, to our knowledge, all previous literature has handled metaphor in this way.

Discrete Polysemy Model Alternatively, the metaphorical and literal senses of an adjective may be distinct enough that averaging the two senses together in a single adjective matrix produces representations that are not well-suited for either metaphorical or literal phrases. Thus, the literal-metaphorical distinction could be problematic for CDSMs in the way that Baroni et al. (2014) suggested that homonyms are. Just as Kartsaklis and Sadrzadeh (2013a) solve this problem by representing each sense of a homonym by a different adjective matrix, we represent literal and metaphorical senses by different adjective matrices. Each literal phrase i is represented by

$$\mathbf{p}_i = \mathbf{A}_{\text{LIT}(a)} \mathbf{n}_i, \quad (2)$$

where $\mathbf{A}_{\text{LIT}(a)}$ is the literal matrix for adjective a. Likewise, a metaphorical phrase is represented by

$$\mathbf{p}_i = \mathbf{A}_{\text{MET}(a)} \mathbf{n}_i, \quad (3)$$

where $\mathbf{A}_{\text{MET}(a)}$ is the metaphorical matrix for a.

Learning. Given a data set of noun and phrase vectors $\mathcal{D}(a) = \{(\mathbf{n}_i, \mathbf{p}_i)\}_{i=1}^{N}$ for AN phrases involving adjective a extracted using a conventional DSM, our goal is to learn $\mathbf{A}_{\mathcal{D}(a)}$. This can be treated as an optimization problem, of learning an estimate $\hat{\mathbf{A}}_{\mathcal{D}(a)}$ that minimizes a specified loss function. In the case of the squared error loss, $L(\mathbf{A}_{\mathcal{D}(a)}) = \sum_{i \in \mathcal{D}(a)} \|\mathbf{p}_i - \mathbf{A}_{\mathcal{D}(a)} \mathbf{n}_i\|_2^2$, the optimal solution can be found precisely using ordinary least-squares regression. However, this may result in overfitting because of the large number of parameters relative to the number of samples (i.e., phrases). Regularization parameters $\lambda = (\lambda_1, \lambda_2)$ can be introduced to keep $\hat{\mathbf{A}}_{\mathcal{D}(a)}$ small:

$$\sum_{i \in \mathcal{D}(a)} \|\mathbf{p}_i - \hat{\mathbf{A}}_{\mathcal{D}(a)}\mathbf{n}_i\|_2^2 + R(\lambda; \hat{\mathbf{A}}_{\mathcal{D}(a)}),$$

where $R(\lambda; \hat{\mathbf{A}}_{\mathcal{D}}) = \lambda_1\|\hat{\mathbf{A}}_{\mathcal{D}}\|_1 + \lambda_2\|\hat{\mathbf{A}}_{\mathcal{D}}\|_2$. This approach, known as elastic-net regression (Zou and Hastie, 2005), produces better adjective matrices than unregularized regression (Li et al., 2014). Note that the same procedure can be used to learn the adjective representations in both the Contextual Variation model and the Discrete Polysemy model by varying what phrases are included in the training set $\mathcal{D}(a)$. In the Contextual Variation model $\mathcal{D}(a)$ includes both metaphorical and literal phrases, while in the Discrete Polysemy model it includes only metaphorical phrases when learning $\hat{\mathbf{A}}_{\text{MET}(a)}$ and testing on metaphorical phrases (and only literal phrases when learning $\hat{\mathbf{A}}_{\text{LIT}(a)}$ and testing on literal phrases).

4.2 Experimental Setup

Extracting Noun & Phrase Vectors. Our approach for constructing term vector representations is similar to that of Dinu et al. (2013). We first selected the 10K most frequent nouns, adjectives, and verbs to serve as context terms. We then constructed a co-occurrence matrix that recorded term-context co-occurrence within a symmetric 5-word context window of the 50K most frequent POS-tagged terms in the corpus. We then used these co-occurrences to compute the positive pointwise mutual information (PPMI) between every pair of terms, and collected these into a term-term matrix. Next, we reduced the dimensionality of this matrix to 100 dimensions using singular-value decomposition. Additionally, we computed "ground truth" distributional vectors for all the annotated AN phrases in our data set by treating the phrases as single terms and computing their PPMI with the 50K single-word terms, and then projecting them onto the same 100-dimensional basis.

Training Adjective Matrices. For each adjective a that we are testing, we split the phrases involving that adjective into two subsets, the literal (LIT) subset and the metaphorical (MET) subset. We then split the subsets into 10 folds, so that we do not train and test any matrices on the same phrases. For each fold k, we train three adjective matrices: $\hat{\mathbf{A}}_{\text{MET}(a)}$ using all phrases from the MET set not in fold k; $\hat{\mathbf{A}}_{\text{LIT}(a)}$ using all phrases from the LIT set not in fold k; and $\hat{\mathbf{A}}_{\text{BOTH}(a)}$ using all the phrases from either subset not in fold k. Within each fold, we use nested cross-validation as outlined in Li et al. (2014) to determine the regularization parameters for each regression problem.

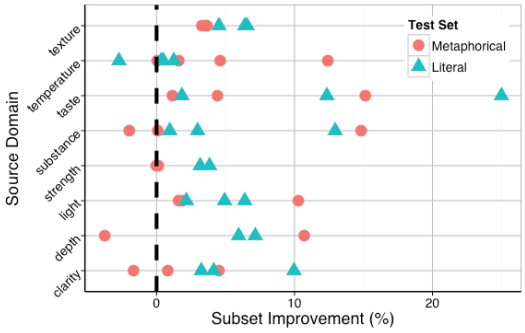


Figure 1: Reduction in error from training on targeted subset (MET/LIT) rather than on all phrases.

4.3 Evaluating Vector Representations

Evaluation. Our goal is to produce a vector prediction of each phrase that will be close to its ground truth distributional vector. Phrase vectors directly extracted from the corpus by treating the phrase as a single term are the gold standard for predicting human judgment and producing paraphrases (Dinu et al., 2013), so we use these as our ground truth. The quality of the vector prediction for phrase i is measured using the cosine distance between the phrase's ground truth vector $\mathbf{p}_i$ and the vector prediction $\hat{\mathbf{p}}_i$:

$$err(\hat{\mathbf{p}}_i) = 1 - \cos(\hat{\mathbf{p}}_i, \mathbf{p}_i).$$

We then analyze the benefit of training on a reduced subset by calculating a "subset improvement" (SI) score for the MET and LIT subsets of each adjective a. We define the SI for each subset $\mathcal{D}(a) \in \{\text{LIT}(a), \text{MET}(a)\}$ as:

$$SI(\mathcal{D}(a)) = 1 - \frac{\sum_{i \in \mathcal{D}(a)} err(\hat{\mathbf{A}}_{\mathcal{D}(a)}\mathbf{n}_i)}{\sum_{i \in \mathcal{D}(a)} err(\hat{\mathbf{A}}_{\text{BOTH}(a)}\mathbf{n}_i)}$$

Positive values of SI thus indicate improved performance when trained on a reduced subset compared to the full set of phrases. For example $SI_{\text{LIT}(a)} = 5\%$ tells us that predicting the phrase vectors for LIT phrases of adjective a using the LIT matrix resulted in a 5% reduction in mean cosine error compared to predicting the phrase vectors using the BOTH matrix.

Results. The results are summarized in Fig. 1. Each point indicates the SI for a single adjective and for a single subset. Adjectives are grouped by source domain along the y-axis. Overall, almost every item shows a subset improvement; and, for every source domain, the majority of adjectives show a subset improvement.

We analyzed per-adjective SI by fitting a linear mixed-effects model, with a fixed intercept, a fixed effect of test subset (MET vs. LIT), a random effect of source domain, and the maximal converging random effects structure (uncorrelated random intercepts and slopes) (Barr et al., 2013). Training on a targeted subset improved performance by $4.4\% \pm 0.009(SE)$ ($p = .002$). There was no evidence that this differed by test subset (i.e., metaphorical vs. literal senses, $p = .35$). The positive SI from training on a targeted subset suggests that metaphorical and literal uses of the same adjective are semantically distinct.

4.4 Metaphor Classification

Method. The results of the previous section suggest a straightforward classification rule: classify unseen phrase i involving adjective a as metaphorical if $\cos(\mathbf{p}_i, \hat{\mathbf{A}}_{\text{MET}(a)}\mathbf{n}_i) < \cos(\hat{\mathbf{A}}_{\text{LIT}(a)}\mathbf{n}_i)$. Otherwise, we classify it as literal.

Evaluation. We test this method on our data set of 8593 annotated AN phrases using 10-fold cross validation. It is possible that our method's classification performance is not due to the compositional aspect of the model, but rather to some semantic coherence property among the nouns in the AN phrases that we are testing. To control for this possibility, we compare the performance of our method against four baselines. The first baseline, NOUN-NN, measures the cosine distance between the vector for the noun of the AN phrase being tested and the noun vectors of the nouns participating in an AN phrase in the training folds. The test phrase is then assigned the label of the AN phrase whose noun vector is nearest. PHRASE-NN proceeds similarly, but using the ground-truth phrase vectors for the test phrase and the training phrases. The test phrase is then assigned the label of the AN phrase whose vector is nearest. The baseline NOUN-CENT first computes the centroid of the noun vectors of the training phrases that are literal, and the centroid of the noun vectors of the training phrases that are metaphorical. It then assigns the test phrase the label of the centroid whose cosine distance from the test phrase's noun vector is smallest. PHRASE-CENT, proceeds similarly, but using phrase vectors. We measure performance against the manual annotations.

Results. Our classification method achieved a held-out F-score of 0.817, recall of 0.793, precision of 0.842, and accuracy of 0.809. These results were superior to those of the baselines (Table 1). These results are competitive with the state of the art and demonstrate the importance of compositionality in metaphor identification.

Method	F-score	Precision	Recall	Accuracy
MET-LIT	0.817	0.842	0.793	0.809
NOUN-NN	0.709	0.748	0.675	0.703
PHRASE-NN	0.590	0.640	0.547	0.592
NOUN-CENT	0.717	0.741	0.695	0.706
PHRASE-CENT	0.629	0.574	0.695	0.559

Table 1: Performance of the method of §4.4 (MET-LIT) against various baselines.

5 Metaphors as Linear Transformations

One of the principal claims of the CM hypothesis is that CMs are productive: A CM (i.e., mapping) can generate endless new LMs (i.e., linguistic expressions). Cases where the LMs involve an adjective that has already been used metaphorically and for which we have annotated metaphorical and literal examples can be handled by the methods of §4, but when the novel LM involves an adjective that has only been observed in literal usage, we need a more elaborate model. According to the CM hypothesis, an adjective's metaphorical meaning is a result of the action of a source-to-target CM mapping on the adjective's literal sense. If so, then given an appropriate representation of this mapping it should be possible to infer the metaphorical sense of an adjective without ever seeing metaphorical exemplars—that is, using only the adjective's literal sense. Our next experiments seek to determine whether it is possible to represent and learn CM mappings as linear maps in distributional vector space.

5.1 Model

We model each CM mapping $\mathcal{M}$ from source to target domain as a linear transformation $\mathbf{C}_{\mathcal{M}}$:

$$\mathbf{A}_{\text{MET}(a)}\mathbf{n}_i \approx \mathbf{C}_{\mathcal{M}}\mathbf{A}_{\text{LIT}(a)}\mathbf{n}_i \tag{4}$$

We can apply a two-step regression to learn $\mathbf{C}_{\mathcal{M}}$. First we apply elastic-net regression to learn the literal adjective matrix $\hat{\mathbf{A}}_{\text{LIT}(a)}$ as in §4.2. Then we can substitute this estimate into Eq. (4), and apply elastic-net regression to learn the $\hat{\mathbf{C}}_{\mathcal{M}}$ that minimizes the regularized squared error loss:

$$\sum_{a \in \mathcal{M}} \sum_{i \in \mathcal{D}(a)} \|\mathbf{p}_i - \hat{\mathbf{C}}_{\mathcal{M}}\hat{\mathbf{A}}_{\text{LIT}(a_i)}\mathbf{n}_i\|_2^2 + R(\lambda; \hat{\mathbf{C}}_{\mathcal{M}}).$$

To learn $C_{\mathcal{M}}$ in this regression problem, we can pool together and train on phrases from many different adjectives that participate in $\mathcal{M}$.

5.2 Experimental Setup

We used a cross-validation scheme where we treated each adjective in a source domain as a fold in training the domain's metaphor transformation matrix. The nested cross-validation procedure we use to set regularization parameters λ and evaluate performance requires at least 3 adjectives in a source domain, so we evaluate on the 6 source domain classes containing at least 3 adjectives. The total number of phrases for these 19 adjectives is 6987 (3659 metaphorical, 3328 literal).

5.3 Evaluating Vector Representations

Evaluation. We wish to test whether CM mappings learned from one set of adjectives are transferable to new adjectives for which metaphorical phrases are unseen. As in §4, models were evaluated using cosine error compared to the ground truth phrase vector representation. Since our goal is to improve the vector representation of metaphorical phrases given no metaphorical annotations, we measure performance on the MET phrase subset for each adjective. We compare the performance of the transformed LIT matrix $\mathbf{C}_{\mathcal{M}}\mathbf{A}_{\text{LIT}(a)}$ against the performance of the original LIT matrix $\mathbf{A}_{\text{LIT}(a)}$ by defining the metaphor transformation improvement (MTI) as:

$$MTI(a) = 1 - \frac{\sum_{i \in \text{MET}} err(\mathbf{C}_{\mathcal{M}}\hat{\mathbf{A}}_{\text{LIT}(a)})}{\sum_{i \in \text{MET}} err(\hat{\mathbf{A}}_{\text{LIT}(a)})}.$$

Results. Per-adjective MTI was analyzed with a linear mixed-effects model, with a fixed intercept, a random effect of source domain, and random intercepts. Transforming the LIT matrix using the CM mapping matrix improved performance by $11.5\% \pm 0.023(SE)$ ($p < .001$). On average, performance improved for 18 of 19 adjectives and for every source domain ($p = .03$, binomial test; Fig. 2). Thus, mapping structure is indeed shared across adjectives participating in the same CM.

5.4 Metaphor Classification

Method. Once again our results suggest a procedure for metaphor classification. This procedure can classify phrases involving adjectives without seeing any metaphorical annotations. For any unseen phrase i involving an adjective a_i, we classify the phrase as metaphorical if $\cos(\mathbf{p}_i, \hat{\mathbf{C}}_{\mathcal{M}}\hat{\mathbf{A}}_{\text{LIT}(a_i)}\mathbf{n}_i) < \cos(\mathbf{p}_i, \hat{\mathbf{A}}_{\text{LIT}(a_i)}\mathbf{n}_i)$. Otherwise, we classify it as literal. We used the same procedure as in §4.2 to learn $\hat{\mathbf{A}}_{\text{LIT}(a_i)}$.

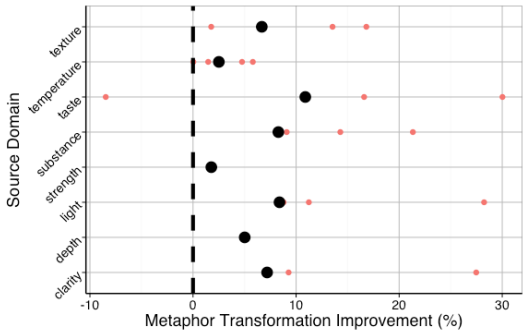


Figure 2: Reduction in error from transforming LIT matrix using metaphorical mapping. Mean change was positive for every domain (large black), and for all but one adjective (small red).

Method	F-score	Precision	Recall	Accuracy
TRANS-LIT	0.793	0.716	0.819	0.804
MET-LIT	0.838	0.856	0820	0.833
NOUN-NN	0.692	0.732	0.655	0.693
PHRASE-NN	0.575	0.625	0.532	0.587
NOUN-CENT	0.703	0.722	0.685	0.696
PHRASE-CENT	0.610	0.552	0.681	0.542

Table 2: Performance of method of §5.4 (TRANS-LIT) against method of §4.4 (MET-LIT) and various baselines.

Results. Our method achieved an F-score of 0.793 on the classification of phrases involving unseen adjectives. On this same set of phrases, the method of §4.4 achieved an F-score of 0.838. Once again, the performance of our method was superior to the performance of the baselines (Table 2; the MET-LIT figures in Table 2 differ slightly from those in Table 1 because only 19 of 23 adjectives are tested). For comparison, we also include the classification performance using the MET-LIT method of §4.4. While MET-LIT slightly outperforms TRANS-LIT, the latter has the benefit of not needing annotations for metaphorical phrases for the test adjective. Hence, our approach is generalizable to cases where such annotations are unavailable with only slight performance reduction.

6 Discussion

Overall, our results show that taking metaphor into account has the potential to improve CDSMs and expand their domain of applicability. The findings of §4 suggest that collapsing across metaphorical and literal uses may hurt accuracy of vector rep-

resentations in CDSMs. While the method in §4 depends on explicit annotations of metaphorical and literal senses, the method in §5 provides a way to generalize these representations to adjectives for which metaphorical training data is unavailable, by showing that metaphorical mappings are transferable across adjectives from the same source domain. Note that an accurate matrix representation of the literal sense of each adjective is still required in the experimental setup of §5. This particular choice of setup allowed a proof of concept of the hypothesis that metaphors function as cross-domain transformations, but in principle it would be desirable to learn transformations from a general BOTH matrix representation for any adjective in a source domain to its MET matrix representation. This would enable improved vector representations of metaphorical AN phrases without annotation for unseen adjectives.

The success of our models on the metaphor classification tasks demonstrates that there is information about metaphoricity of a phrase inherent in the composition of the meanings of its components. Notably, our results show that this metaphorical compositionality can be captured from corpus-derived distributional statistics. We also noticed some trends at the level of individual phrases. In particular, classification performance and vector accuracy tended to be lower for metaphorical phrases whose nouns are distributionally similar to nouns that tend to participate in literal phrases (e.g., *reception* is similar to *foyer* and *refreshment* in our corpus; *warm reception* is metaphorical while *warm foyer* is literal). Another area where classification accuracy is low is in phrases with low corpus occurrence frequency. The ground truth vectors for these phrases exhibit high sample variance and sparsity. Many such phrases sound paradoxical (e.g., *bitter sweetness*).

Our results could also inform debates within cognitive science. First, cognitive scientists debate whether words that are used both literally and figuratively (e.g., *long road*, *long meeting*) are best understood as having a single, abstract meaning that varies with context or two distinct but related meanings. For instance, some argue that domains like space, time, and number operate over a shared, generalized magnitude system, yet others maintain that our mental representation of time and number is distinct from our mental representation of space, yet inherited metaphorically from it (Winter et al., 2015). Our results suggest that figurative and literal senses involve quite different patterns of use. This is statistical evidence that adjectives that are used metaphorically have distinct related senses, not a single abstract sense.

Second, the Conceptual Metaphor Theory account hypothesizes that LMs are an outgrowth of metaphorical thought, which is in turn an outgrowth of embodied experiences that conflate source and target domains—experience structures thought, and thought structures language (Lakoff, 1993). However, recent critics have argued for the opposite causal direction: Linguistic regularities may drive the mental mapping between source and target domains (Hutchinson and Louwerse, 2013; Casasanto, 2014; Hutchinson and Louwerse, 2014). Our results show that, at least for AN pairs, the semantic structure of a source domain and its mapping to a metaphorical target domain are available in the distributional statistics of language itself. There may be no need, therefore, to invoke embodied experience to explain the prevalence of metaphorical thought in adult language users. A lifetime of experience with literal and metaphorical language may suffice.

7 Conclusion

We have shown that modeling metaphor explicitly within a CDSM can improve the resulting vector representations. According to our results, the systematicity of metaphor can be exploited to learn linear transformations that represent the action of metaphorical mappings across many different adjectives in the same semantic domain. Our classification results suggest that the compositional distributional semantics of a phrase can inform classification of the phrase for metaphoricity.

Beyond improvements to the applications we presented, the principles underlying our methods also show potential for other tasks. For instance, the LIT and MET adjective matrices and the CM mapping matrix learned with our methods could be applied to improve automated paraphrasing of AN phrases. Our work is also directly extendable to other syntactic constructions. In the CDSM framework we apply, verbs would be represented as third-order tensors. Tractable and efficient methods for estimating these verb tensors are now available (Fried et al., 2015). It may also be possible to extend the coverage of our system by using automated word-sense disambiguation to bootstrap annotations and therefore construct LIT

and MET matrices in a minimally supervised fashion (Kartsaklis et al., 2013b). Finally, it would be interesting to investigate modeling metaphorical mappings as nonlinear mappings within the deep learning framework.

Acknowledgments

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575. Ekaterina Shutova's research is supported by the Leverhulme Trust Early Career Fellowship.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

M. Baroni, S. Bernardini, A. Ferraresi, and E. Zanchetta. 2009. The WaCky wide web: A collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Marco Baroni, Raffaela Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program of compositional distributional semantics. *Linguistic Issues in Language Technology*, 9.

Dale J. Barr, Roger Levy, Christoph Scheepers, and Harry J. Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3):255–278.

Steven Bethard, Vicky Tzuyin Lai, and James H. Martin. 2009. Topic model analysis of metaphor frequency for psycholinguistic stimuli. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, pages 9–16. Association for Computational Linguistics.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 1–4.

Julia Birke and Anoop Sarkar. 2006. A clustering approach for nearly unsupervised recognition of nonliteral language. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 329–336.

BNC Consortium. 2007. British National Corpus, Version 3 BNC XML edition.

Gemma Boleda, Eva Maria Vecchi, Miquel Cornudella, and Louise McNally. 2012. First-order vs. higher-order modification in distributional semantics. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1223–1233. Association for Computational Linguistics.

Lynne Cameron. 2003. *Metaphor in Educational Discourse*. A&C Black, London.

Daniel Casasanto. 2014. Development of metaphorical thinking: The role of language. In Mike Borkent, Barbara Dancygier, and Jennifer Hinnell, editors, *Language and the Creative Mind*, pages 3–18. CSLI Publications, Stanford.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. In *Linguistic Analysis (Lambek Festschrift)*, pages 345–384.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. educational and psychosocial measurement.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. General estimation and evaluation of compositional distributional semantic models. In *Proceedings of the ACL 2013 Workshop on Continuous Vector Space Models and their Compositionality (CVSC 2013)*, pages 50–58, East Stroudsburg, Pennsylvania. ACL.

Jonathan Dunn. 2013a. Evaluating the premises and results of four metaphor identification systems. In *Computational Linguistics and Intelligent Text Processing*, pages 471–486. Springer.

Jonathan Dunn. 2013b. What metaphor identification systems can tell us about metaphor-in-language. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 1–10.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 92–97. Association for Computational Linguistics.

Joseph L. Fleiss, Jacob Cohen, and B.S. Everitt. 1969. Large sample standard errors of kappa and weighted kappa. *Psychological Bulletin*, 72(5):323.

Daniel Fried, Tamara Polajnar, and Stephen Clark. 2015. Low-rank tensors for verbs in compositional distributional semantics. In *Proceedings of the 53nd Annual Meeting of the Association for Computational Linguistics*, Beijing.

Matt Gedigian, John Bryant, Srini Narayanan, and Branimir Ciric. 2006. Catching metaphors. In *Proceedings of the Third Workshop on Scalable Natural Language Understanding*, pages 41–48, New York. Association for Computational Linguistics.

Joseph A. Goguen and D. Fox Harrell. 2005. 7 information visualisation and semiotic morphisms. *Studies in Multidisciplinarity*, 2:83–97.

Joseph A. Goguen and D. Fox Harrell. 2010. Style: A computational and conceptual blending-based approach. In *The Structure of Style*, pages 291–316. Springer, New York.

Joseph Goguen. 1999. An introduction to algebraic semiotics, with application to user interface design. In *Computation for metaphors, analogy, and agents*, pages 242–291. Springer.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English Gigaword. *Linguistic Data Consortium, Philadelphia*.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 33–37. Association for Computational Linguistics.

Ilana Heintz, Ryan Gabbard, Mahesh Srinivasan, David Barner, Donald S Black, Marjorie Freedman, and Ralph Weischedel. 2013. Automatic extraction of linguistic metaphor with lda topic modeling. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 58–66.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 52–57.

Sterling Hutchinson and Max Louwerse. 2013. Language statistics and individual differences in processing primary metaphors. *Cognitive Linguistics*, 24(4):667–687.

Sterling Hutchinson and Max M. Louwerse. 2014. Language statistics explain the spatial–numerical association of response codes. *Psychonomic Bulletin & Review*, 21(2):470–478.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, et al. 2013a. Prior disambiguation of word tensors for constructing sentence vectors. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1590–1601.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2013b. Separating disambiguation from composition in distributional semantics. In *Proceedings of the 2013 Conference on Computational Natural Language Learning*, pages 114–123.

Saisuresh Krishnakumaran and Xiaojin Zhu. 2007. Hunting elusive metaphors using lexical resources. In *Proceedings of the Workshop on Computational approaches to Figurative Language*, pages 13–20. Association for Computational Linguistics.

Werner Kuhn and Andrew U Frank. 1991. A formalization of metaphors and image-schemas in user interfaces. In *Cognitive and linguistic aspects of geographic space*, pages 419–434. Springer.

George Lakoff and Mark Johnson. 1981. *Metaphors we live by*. University of Chicago Press, Chicago.

George Lakoff. 1989. Some empirical results about the nature of concepts. *Mind & Language*, 4(1-2):103–129.

George Lakoff. 1993. The contemporary theory of metaphor. In Andrew Ortony, editor, *Metaphor and Thought*. Cambridge University Press, Cambridge.

Joachim Lambek. 1999. Type grammar revisited. In *Logical aspects of computational linguistics*, pages 1–27. Springer, Berlin.

Linlin Li and Caroline Sporleder. 2009. Classifier combination for contextual idiom detection without labelled data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 315–323. Association for Computational Linguistics.

Linlin Li and Caroline Sporleder. 2010. Using Gaussian mixture models to detect figurative language in context. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 297–300. Association for Computational Linguistics.

Linlin Li, Benjamin Roth, and Caroline Sporleder. 2010. Topic models for word sense disambiguation and token-based idiom detection. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1138–1147. Association for Computational Linguistics.

Jiming Li, Marco Baroni, and Georgiana Dinu. 2014. Improving the lexical function composition model with pathwise optimized elastic-net regression. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 434–442.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL-08: HLT*, pages 236–244.

Michael Mohler, David Bracewell, David Hinote, and Marc Tomlinson. 2013. Semantic signatures for example-based linguistic metaphor detection. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 27–35.

Richard Montague. 1970. English as a formal language. In B Visentini and et al, editors, *Linguaggi nella Società e nella Tecnica*. Edizioni di Comunitá, Milan.

Yair Neuman, Dan Assaf, Yohai Cohen, Mark Last, Shlomo Argamon, Newton Howard, and Ophir Frieder. 2013. Metaphor identification in large texts corpora. *PLoS ONE*, 8:e62343.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 613–619. ACM.

Barbara H. Partee. 1994. Lexical semantics and compositionality. In Lila Gleitman and Mark Liberman, editors, *Invitation to Cognitive Science 2nd Edition, Part I: Language*. MIT Press, Cambridge, Mass., USA.

Hinrich Schütze. 1998. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1002–1010. Association for Computational Linguistics.

Ekatrina Shutova. 2015. Design and evaluation of metaphor processing systems. volume Forthcoming.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

David I. Spivak. 2014. *Category Theory for the Sciences*. MIT Press, Cambridge, Mass., USA.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 754–762. Association for Computational Linguistics.

Gerard J. Steen, Aletta G. Dorst, J. Berenike Herrmann, Anna Kaal, Tina Krennmayr, and Trijntje Pasma. 2010. *A method for linguistic metaphor identification: From MIP to MIPVU*, volume 14. John Benjamins Publishing, Amsterdam/Philadelphia.

Tomek Strzalkowski, George A. Broadwell, Sarah Taylor, Laurie Feldman, Boris Yamrom, Samira Shaikh, Ting Liu, Kit Cho, Umit Boz, Ignacio Cases, and Kyle Elliot. 2013. Robust extraction of metaphors from novel data. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 67–76, Atlanta, Georgia. Association for Computational Linguistics.

Masashi Tsubaki, Kevin Duh, Masashi Shimbo, and Yuji Matsumoto. 2013. Modeling and learning semantic co-compositionality through prototype projections and neural networks. In *The 2013 Conference on Empirical Methods in Natural Language Processing*, pages 130–140.

Yulia Tsvetkov, Elena Mukomel, and Anatole Gershman. 2013. Cross-lingual metaphor detection using common semantic features.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Peter D. Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on the Empirical Methods in Natural Language Processing*, EMNLP '11, pages 680–690, Stroudsburg, PA, USA. Association for Computational Linguistics.

Peter D. Turney. 2013. Distributional semantics beyond words: supervised learning of analogy and paraphrase. *Transactions of the Association for Computational Linguistics (TACL)*, 1:353–366.

Akira Utsumi. 2006. Computational exploration of metaphor comprehension processes. In *Proceedings of the 28th Annual Meeting of the Cognitive Science Society (CogSci2006)*, pages 2281–2286.

Bodo Winter, Tyler Marghetis, and Teenie Matlock. 2015. Of magnitudes and metaphors: Explaining cognitive interactions between space, time, and number. *Cortex*, 64:209–224.

Hui Zou and Trevor Hastie. 2005. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Idiom Token Classification using Sentential Distributed Semantics

Giancarlo D. Salton and **Robert J. Ross** and **John D. Kelleher**
Applied Intelligence Research Centre
School of Computing
Dublin Institute of Technology
Ireland
giancarlo.salton@mydit.ie {robert.ross,john.d.kelleher}@dit.ie

Abstract

Idiom token classification is the task of deciding for a set of potentially idiomatic phrases whether each occurrence of a phrase is a literal or idiomatic usage of the phrase. In this work we explore the use of Skip-Thought Vectors to create distributed representations that encode features that are predictive with respect to idiom token classification. We show that classifiers using these representations have competitive performance compared with the state of the art in idiom token classification. Importantly, however, our models use only the sentence containing the target phrase as input and are thus less dependent on a potentially inaccurate or incomplete model of discourse context. We further demonstrate the feasibility of using these representations to train a competitive general idiom token classifier.

1 Introduction

Idioms are a class of multiword expressions (MWEs) whose meaning cannot be derived from their individual constituents (Sporleder et al., 2010). Idioms often present idiosyncratic behaviour such as violating selection restrictions or changing the default semantic roles of syntactic categories (Sporleder and Li, 2009). Consequently, they present many challenges for Natural Language Processing (NLP) systems. For example, in Statistical Machine Translation (SMT) it has been shown that translations of sentences containing idioms receive lower scores than translations of sentences that do not contain idioms (Salton et al., 2014).

Idioms are pervasive across almost all languages and text genres and as a result broad coverage NLP systems must explicitly handle idioms (Villavicencio et al., 2005). A complicating factor, however, is that many idiomatic expressions can be used both literally or figuratively. In general, idiomatic usages are more frequent, but for some expressions the literal meaning may be more common (Li and Sporleder, 2010a). As a result, there are two fundamental tasks in NLP idiom processing: *idiom type* classification is the task of identifying expressions that have possible idiomatic interpretations and *idiom token* classification is the task of distinguishing between idiomatic and literal usages of potentially idiomatic phrases (Fazly et al., 2009). In this paper we focus on this second task, *idiom token* classification.

Previous work on idiom token classification, such as (Sporleder and Li, 2009) and (Peng et al., 2014), often frame the problem in terms of modelling the global lexical context. For example, these models try to capture the fact that the idiomatic expression *break the ice* is likely to have a literal meaning in a context containing words such as *cold*, *frozen* or *water* and an idiomatic meaning in a context containing words such as *meet* or *discuss* (Li and Sporleder, 2010a). Frequently these global lexical models create a different idiom token classifier for each phrase. However, a number of papers on idiom type and token classification have pointed to a range of other features that could be useful for idiom token classification; including local syntactic and lexical patterns (Fazly et al., 2009) and cue words (Li and Sporleder, 2010a). However, in most cases these non-global features are specific to a particular phrase. So a key challenge is to identify from a range of features which features are the correct features to use for idiom token classification for a specific expression.

Meanwhile, in recent years there has been an explosion in the use of neural networks for learning distributed representations for language (e.g.,

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 194–204,
Berlin, Germany, August 7-12, 2016.

Socher et al. (2013), Kalchbrenner et al. (2014) and Kim (2014)). These representations are automatically trained from data and can simultaneously encode multiple linguistics features. For example, word embeddings can encode gender distinctions and plural-singular distinctions (Mikolov et al., 2013b) and the representations generated in sequence to sequence mappings have been shown to be sensitive to word order (Sutskever et al., 2014). The recent development of Skip-Thought Vectors (or Sent2Vec) (Kiros et al., 2015) has provided an approach to learn distributed representations of sentences in an unsupervised manner.

In this paper we explore whether the representations generated by Sent2Vec encodes features that are useful for idiom token classification. This question is particularly interesting because the Sent2Vec based models only use the sentence containing the phrase as input whereas the baselines systems use full the paragraph surrounding the sentence. We further investigate the construction of a "general" classifier that can predict if a sentence contains literal or idiomatic language (independent of the expression) using just the distributed representation of the sentence. This approach contrasts with previous work that has primarily adopted a "per expression" classifier approach and has been based on more elaborate context features, such as discourse and lexical cohesion between and sentence and the larger context. We show that our method needs less contextual information than the state-of-the-art method and achieves competitive results, making it an important contribution to a range of applications that do not have access to a full discourse context. We proceed by introducing that previous work in more detail.

2 Previous Work

One of the earliest works on idiom token classification was on Japanese idioms (Hashimoto and Kawahara, 2008). This work used a set of features, commonly used in Word Sense Disambiguation (WSD) research, that were defined over the text surrounding a phrase, as well as a number of idiom specific features, which were in turn used to train an SVM classifier based on a corpus of sentences tagged as either containing an idiomatic usage or a literal usage of a phrase. Their results indicated that the WSD features worked well on idiom token classification but that their idioms specific features did not help on the task.

Focusing on idiom token classification in English, Fazly et al. (2009) developed the concept of a canonical form (defined in terms of local syntactic and lexical patterns) and argued that for each idiom there is a distinct canonical form (or small set of forms) that mark idiomatic usages of a phrase. Meanwhile Sporleder and Li (2009) proposed a model based on how strongly an expression is linked to the overall cohesive structure of the discourse. Strong links result in a literal classification, otherwise an idiomatic classification is returned. In related work, Li and Sporleder (2010a) experimented with a range of features for idiom token classification models, including: global lexical context, discourse cohesion, syntactic structures based on dependency parsing, and local lexical features such as cue words, occurring just before or after a phrase. An example of a local lexical feature is when the word *between* occurs directly after *break the ice*; here this could mark an idiomatic usage of the phrase: *it helped to break the ice between Joe and Olivia*. The results of this work indicated that features based on global lexical context and discourse cohesion were the best features to use for idiom token classification. The inclusion of syntactic structures in the feature set provided a boost to the performance of the model trained on global lexical context and discourse cohesion. Interestingly, unlike the majority of previous work on idiom token classification Li and Sporleder (2010a) also investigated building general models that could work across multiple expressions. Again they found that global lexical context and discourse cohesion were the best features in their experiments.

Continuing work on this topic, Li and Sporleder (2010b) present research based on the assumption that literal and figurative language are generated by two different Gaussians. The model representation is based on semantic relatedness features similar to those used earlier in (Sporleder and Li, 2009). A Gaussian Mixture Model was trained using an Expectation Maximization method with the classification of instances performed by choosing the category which maximises the probability of fitting either of the Gaussian components. Li and Sporleder (2010b)'s results confirmed the findings from previous work that figurative language exhibits less cohesion with the surrounding context then literal language.

More recently, Feldman and Peng (2013) describes an approach to idiom token identification that frames the problem as one of outlier detection. The intuition behind this work is that because idiomatic usages of phrases have weak cohesion with the surrounding context they are semantically distant from local topics. As a result, phrases that are semantic outliers with respect to the context are likely to be idioms. Feldman and Peng (2013) explore two different approaches to outlier detection based on principle component analysis (PCA) and linear discriminant analysis (LDA) respectively. Building on this work, Peng et al. (2014) assume that phrases within a given text segment (e.g., a paragraph) that are semantically similar to the main topic of discussion in the segment are likely to be literal usages. They use Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to extract a topic representation, defined as a topic term document matrix, of each text segment within a corpus. They then trained a number of models that classify a phrase in a given text segment as a literal or idiomatic usage by using the topic term document matrix to project the phrase into a topic space representation and label outliers within the topic space as idiomatic. To the best of our knowledge, Peng et al. (2014) is currently the best performing approach to idiom token classification and we use their models as our baseline[1].

3 Skip-Thought Vectors

While idiom token classification based on long range contexts, such as is explored in a number of the models outlined in the previous section, generally achieve good performance, an NLP system may not always have access to the surrounding context, or may indeed find it challenging to construct a reliable interpretation of that context. Moreover, the construction of classifiers for each individual idiom case is resource intensive, and we argue fails to easily scale to under-resourced languages. In light of this, in our work we are exploring the potential of distributed compositional semantic models to produce reliable estimates of idiom token classification.

Skip-Thought Vectors (Sent2Vec) (Kiros et al., 2015) are a recent prominent example of such distributed models. Skip-Thought Vectors are an application of the Encoder/Decoder framework (Sutskever et al., 2014), a popular architecture for NMT (Bahdanau et al., 2015) based on recurrent neural networks (RNN). The encoder takes an input sentence and maps it into a distributed representation (a vector of real numbers). The decoder is a language model that is conditioned on the distributed representation and, in Sent2Vec, is used to "predict" the sentences surrounding the input sentence. Consequently, the Sent2Vec encoder learns (among other things) to encode information about the context of an input sentence without the need of explicit access to it. Figure 1 presents the architecture of Sent2Vec.

More formally, assume a given tuple (s_{i-1}, s_i, s_{i+1}) where s_i is the input sentence, s_{i-1} is the previous sentence to s_i and s_{i+1} is the next sentence to s_i. Let w_i^t denote the t-th word for s_i and $\mathbf{x}_i^t$ denote its word embedding. We follow Kiros et al. (2015) and describe the model in three parts: encoder, decoder and objective function.

Encoder. Given the sentence s_i of length N, let $w_i^1, \ldots, w_i^N$ denote the words in s_i. At each timestep t, the encoder (in this case an RNN with Gated Recurrent Units - GRUs (Cho et al., 2014)) produces a hidden state $\mathbf{h}_i^t$ that represents the sequence $w_i^1, \ldots, w_i^t$. Therefore, $\mathbf{h}_i^N$ represents the full sentence. Each $\mathbf{h}_i^N$ is produced by iterating the following equations (without the subscript i):

$$\mathbf{r}^t = \sigma(\mathbf{W}_r^e \mathbf{x}^t + \mathbf{U}_r^e \mathbf{h}^{t-1}) \quad (1)$$

$$\mathbf{z}^t = \sigma(\mathbf{W}_z^e \mathbf{x}^t + \mathbf{U}_z^e \mathbf{h}^{t-1}) \quad (2)$$

$$\tilde{\mathbf{h}}^t = tanh(\mathbf{W}^e \mathbf{x}^t + \mathbf{U}^{\mathbf{e}}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \quad (3)$$

$$\mathbf{h}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \tilde{\mathbf{h}}^t \quad (4)$$

where $\mathbf{r}^t$ is the reset gate, $\mathbf{z}^t$ is the update gate, $\tilde{\mathbf{h}}^t$ is the proposed update state at time t and $\odot$ denotes a component-wise product.

Decoder. The decoder is essentially a neural language model conditioned on the input sentence representation $\mathbf{h}_i^N$. However, two RNNs are used (one for the sentence s_{i-1} and the other for the sentence s_{i+1}) with different parameters except the embedding matrix ($\mathbf{E}$), and a new set of matrices ($\mathbf{C}_r$, $\mathbf{C}_z$ and $\mathbf{C}$) are introduced to condition the GRU on $\mathbf{h}_i^N$. Let $\mathbf{h}_{i+1}^t$ denote the hidden state of the decoder of the sentence s_{i+1} at time t. De-

[1]However, it is not possible for us to reproduce their results directly as they "apply the (modified) Google stop list before extracting the topics" (Peng et al., 2014, p. 2023) and, to date, we do not have access to the modified list. So in our experiments we compare our results with the results they report on the same data.

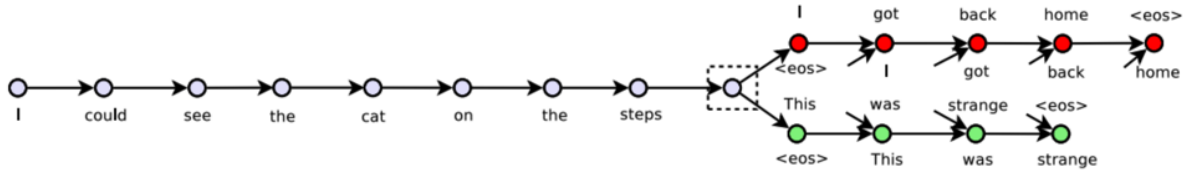


Figure 1: Picture representing the Encoder/Decoder architecture used in the Sent2Vec as shown in Kiros et al. (2015). The gray circles represent the Encoder unfolded in time, the red and the green circles represent the Decoder for the previous and the next sentences respectively also unfolded in time. In this example, the input sentence presented to the Encoder is *I could see the cat on the steps*. The previous sentence is *I got back home* and the next sentence is *This was strange*. Unattached arrows are connected to the encoder output (which is the last gray circle).

coding s_{i+1} requires iterating the following equations:

$$\mathbf{r}^t = \sigma(\mathbf{W}_r^d \mathbf{x}^t + \mathbf{U}_r^d \mathbf{h}^{t-1} + \mathbf{C}_r \mathbf{h}_i^N) \quad (5)$$

$$\mathbf{z}^t = \sigma(\mathbf{W}_z^d \mathbf{x}^t + \mathbf{U}_z^d \mathbf{h}^{t-1} + \mathbf{C}_z \mathbf{h}_i^N) \quad (6)$$

$$\tilde{\mathbf{h}}^t = tanh(\mathbf{W}^d \mathbf{x}^t + \mathbf{U}^{\mathbf{d}}(\mathbf{r}^t \odot \mathbf{h}^{t-1}) + \mathbf{C}\mathbf{h}_i^N) \quad (7)$$

$$\mathbf{h}_{i+1}^t = (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \tilde{\mathbf{h}}^t \quad (8)$$

where $\mathbf{r}^t$ is the reset gate, $\mathbf{z}^t$ is the update gate, $\tilde{\mathbf{h}}^t$ is the proposed update state at time t and $\odot$ denotes a component-wise product. An analogous computation is required to decode s_{i-1}.

Given $\mathbf{h}_{i+1}^t$, the probability of the word w_{i+1}^t conditioned on the previous $w_{i+1}^{<t}$ words and the encoded representation produced by the encoder (h_i^N) is:

$$P(w_{i+1}^t | w_{i+1}^{<t}, h_i^N) \propto exp(\mathbf{E}_{w_{i+1}^t} \mathbf{h}_{i+1}^t) \quad (9)$$

where $\mathbf{E}_{w_{i+1}^t}$ denotes the embedding for the word w_{i+1}^t. An analogous computation is performed to find the probability of s_{i-1}.

Objective. Given the tuple (s_{i-1}, s_i, s_{i+1}), the objective is to optimize the sum of the log-probabilities of the next (s_{i+1}) and previous (s_{i-1}) sentences given the distributed representation ($\mathbf{h}_i^N$) of s_i:

$$\sum \log P(w_{i+1}^t | w_{i+1}^{<t}, h_i^N) + P(w_{i-1}^t | w_{i-1}^{<t}, h_i^N) \quad (10)$$

where the total objective is summed over all training tuples (s_{i-1}, s_i, s_{i+1}).

The utility of Sent2Vec is that it is possible to infer properties of the surrounding context only from the input sentence. Therefore, we can assume that the Sent2Vec distributed representation is also carrying information regarding its context (without the need to explicitly access it). Following that intuition, we can train a supervised classifier only using the labelled sentences containing examples of idiomatic or literal language use without modelling long windows of context or using methods to extract topic representations.

4 Experiments

In the following we describe a study that evaluates the predictiveness of the distributed representations generated by Sent2Vec for idiom token classifier. We first evaluate these representations using a "per expression" study design (i.e., one classifier per expression) and compare our results to those of Peng et al. (2014) who applied multi-paragraphs contexts to generate best results. We also experiment with a "general" classifier trained and tested on a set of mixed expressions.

4.1 Dataset

In order to make our results comparable with (Peng et al., 2014) we used the same VNC-Tokens dataset (Cook et al., 2008) that they used in their experiments. The dataset used is a collection of sentences containing 53 different Verb Noun Constructions[2] (VNCs) extracted from the British National Corpus (BNC) (Burnard, 2007). In total, the VNC-Token dataset has 2984 sentences where each sample sentence is labelled with one of three labels: *I* (idiomatic); *L* (literal); or *Q* (unknown).

[2]This verb-noun constructions can be used either idiomatically or literally.

Of the 56 VNCs in the dataset 28 of these expressions have a reasonably balanced representation (with similar numbers of idiomatic and literal occurrences in the corpus) and the other 28 expressions have a skewed representation (with one class much more common then the other). Following the approach taken by (Peng et al., 2014), in this study we use the "balanced" part of the dataset and considered only those sentences labelled as "I" and "L" (1205 sentences - 749 labelled as "I" and 456 labelled as "L").

Peng et al. (2014) reported the precision, recall and f1-score of their models on 4 of the expressions from the balanced section of dataset: **BlowWhistle**; **MakeScene**; **LoseHead**; and **TakeHeart**. So, our first experiment is designed to compare our models with these baseline systems on a "per-expression" basis. For this experiment we built a training and test set for each of these expressions by randomly sampling expressions following the same distributions presented in Peng et al. (2014). In Table 1 we present those distribution and the split into training and test sets. The numbers in parentheses denote the number of samples labelled as "I".

Expression	Samples	Train Size	Test Size
BlowWhistle	78 (27)	40 (20)	38 (7)
LoseHead	40 (21)	30 (15)	10 (6)
MakeScene	50 (30)	30 (15)	20 (15)
TakeHeart	81 (61)	30 (15)	51 (46)

Table 1: The sizes of the samples for each expression and the split into training and test set. The numbers in parentheses indicates the number of idiomatic labels within the set. We follow the same split as described in Peng et al. (2014).

While we wish to base our comparison on the work of Peng et al. (2014) as it is the current state of the art, this is not without its own challenges. In particular we see the choice of these 4 expression as a somewhat random decision as other expressions could also be selected for the evaluation with similar ratios to those described in Table 1. Moreover, the choosen expressions are all semi-compositional and do not consider fully non-compositional expressions (although we believe the task of classifying non-compositional expressions would be easier for any method aimed at idiom token classification as these expressions are high-fixed) .A better evaluation would consider all the 28 expressions of the balanced part of the VNC-tokens dataset. In addition, we also see this choice of training and test splits as somewhat arbitrary. For two of the expressions the test set contain samples in a way that one of the classes outnumber the other by a great amount: for *BlowWhistle*, the literal class contains roughly 4 times more samples than the idiomatic class; and for *TakeHeart* the idiomatic class contains roughly 9 times more samples than the literal class. Our concerns with these very skewed test set ratios is that it is very easy when applying a per expression approach (i.e., a separate model for each expression) for a model to achieve good performance (in terms of precision, recall, ad f1) if the positive class is the majority class in the test set. However, despite these concerns, in our first experiment in order to facilitate comparison with the prior art we follow the expression selections and training/test splits described in Peng et al. (2014).

Studies on the characteristics of distributed semantic representations of words have shown that similar words tend to be represented by points that are close to each other in the semantic feature space (e.g. Mikolov et al. (2013a)). Inspired by these results we designed a second experiment to test whether the Sent2Vec representations would cluster idiomatic sentences in one part of the feature space and literal sentences in another part of the space. For this experiment we used the entire "balanced" part of the VNC-tokens dataset to train and test our "general" (multi-expression) models. In this experiment we wanted the data to reflect, as much as possible, the real distribution of the idiomatic and literal usages of each expression. So, in constructing our training and test set we tried to maintain for each expression the same ratio of idiomatic and literal examples across the training and test set. To create the training and test sets, we split the dataset into roughly 75% for training (917 samples) and 25% for testing (288 samples). We randomly sample the expressions ensuring that the ratio of idiomatic to literal expressions of each expression were maintained across both sets. In Table 2 we show the expressions used and their split into training and testing. The numbers in parentheses are the number of samples labelled as "I".

4.2 Sent2Vec Models

To encode the sentences into their distributed representations we used the code and models made available[3] by Kiros et al. (2015). Using their

[3]https://github.com/ryankiros/skip-thoughts

Expression	Samples	Train Size	Test Size
BlowTop	28 (23)	21 (18)	7 (5)
BlowTrumpet	29 (19)	21 (14)	8 (5)
BlowWhistle	78 (27)	59 (20)	19 (7)
CutFigure	43 (36)	33 (28)	10 (8)
FindFoot	53 (48)	39 (36)	14 (12)
GetNod	26 (23)	19 (17)	7 (6)
GetSack	50 (43)	40 (34)	10 (9)
GetWind	28 (13)	20 (9)	8 (4)
HaveWord	91 (80)	69 (61)	22 (19)
HitRoad	32 (25)	24 (19)	8 (6)
HitRoof	18 (11)	14 (9)	4 (2)
HitWall	63 (7)	50 (6)	13 (1)
HoldFire	23 (7)	19 (5)	4 (2)
KickHeel	39 (31)	30 (23)	9 (8)
LoseHead	40 (21)	29 (15)	11 (6)
LoseThread	20 (18)	16 (15)	4 (3)
MakeFace	41 (27)	31 (21)	10 (6)
MakeHay	17 (9)	12 (6)	5 (3)
MakeHit	14 (5)	9 (3)	5 (2)
MakeMark	85 (72)	66 (56)	19 (16)
MakePile	25 (8)	18 (6)	7 (2)
MakeScene	50 (30)	37 (22)	13 (8)
PullLeg	51 (11)	40 (8)	11 (3)
PullPlug	64 (44)	49 (33)	15 (11)
PullPunch	22 (18)	18 (15)	4 (3)
PullWeight	33 (27)	24 (20)	9 (7)
SeeStar	61 (5)	49 (3)	12 (2)
TakeHeart	81 (61)	61 (45)	20 (16)

Table 2: The sizes of the samples for each expression and the split into training and test set. The numbers in parentheses indicates the number of idiomatic labels within the set.

models it is possible to encode the sentences into three different formats: *uni-skip* (which uses a regular RNN to encode the sentence into a 2400-dimensional vector); *bi-skip* (that uses a bidirectional RNN to encode the sentence also into a 2400-dimensional vector); and the *comb-skip* (a concatenation of *uni-skip* and *bi-skip* which has 4800 dimensions). Their models were trained using the BookCorpus dataset (Zhu et al., 2015) and has been tested in several different NLP tasks as semantic relatedness, paraphrase detection and image-sentence ranking. Although we experimented with all the three models, in this paper we only report the results of classifiers trained and tested using the *comb-skip* features.

4.3 Classifiers

4.3.1 "Per-expression" models

The idea behind Sent2Vec is similar to those of word embeddings experiments: sentences containing similar meanings should be represented by points close to each other in the feature space. Following this intuition we experiment first with a similarity based classifier, the K-Nearest Neighbours (k-NN). For the k-NNs we experimented with $k = \{2, 3, 5, 10\}$.

We also experimented with a more advanced algorithm, namely the Support Vector Machine (SVM) (Vapnik, 1995). We trained the SVM under three different configurations:

Linear-SVM-PE[4]. This model used a "linear" kernel with $C = 1.0$ on all the classification setups.

Grid-SVM-PE. For this model we performed a grid search for the best parameters for each expression. The parameters are: *BlowWhiste* $=$ { kernel: 'rbf', $C = 100$}; *LoseHead* $=$ { kernel: 'rbf', $C = 1$ }; *MakeSene* $=$ { kernel: 'rbf', $C = 100$ }; *TakeHeart* $=$ { kernel: 'rbf', $C = 1000$ }.

SGD-SVM-PE. This model is a SVM with linear kernel but trained using stochastic gradient descent (Bottou, 2010). We set the SGD's learning rates (α) using a grid search: *BlowWhiste* $= \{\alpha = 0.001\}$; *LoseHead* $= \{\alpha = 0.01\}$; *MakeSene* $= \{\alpha = 0.0001\}$; *TakeHeart* $= \{\alpha = 0.0001\}$; *FullDataset* $= \{\alpha = 0.0001\}$. We trained these classifiers for 15 epochs.

4.3.2 "General" models

We consider the task of creating a "general" classifier that takes an example of any potential idiom and classifying it into idiomatic or literal usage more difficult than the "per-expression" classification task. Hence we executed this part of the study with the SVM models only. We trained the same three types of SVM models used in the "per-expression" approach but with the following parameters:

Linear-SVM-GE[5]. This model used a linear kernel with $C = 1.0$ for all the classification sets.

Grid-SVM-GE. For this model we also performed a grid search and set the kernel to "polynomial kernel" of $degree = 2$ with $C = 1000$.

SGD-SVM-GE. We also experimented with a SVM with linear kernel trained using stochastic gradient descent. We set the SGD's learning rate $\alpha = 0.0001$ after performing a grid search. We trained this classifier for 15 epochs.

5 Results and Discussion

We first present the results for the per expression comparison with Peng et al. (2014) and then in

[4]PE stands for "per-expression"

[5]GE stands for "general".

Models	BlowWhistle			LoseHead			MakeScene			TakeHeart		
	P.	R.	F1	P.	R.	F1	P.	R.	F1	P.	R.	F1
Peng et. al (2014)												
FDA-Topics	0.62	0.60	0.61	0.76	0.97	**0.85**	0.79	0.95	0.86	0.93	0.99	**0.96**
FDA-Topics+A	0.47	0.44	0.45	0.74	0.93	0.82	0.82	0.69	0.75	0.92	0.98	0.95
FDA-Text	0.65	0.43	0.52	0.72	0.73	0.72	0.79	0.95	0.86	0.46	0.40	0.43
FDA-Text+A	0.45	0.49	0.47	0.67	0.88	0.76	0.80	0.99	**0.88**	0.47	0.29	0.36
SVMs-Topics	0.07	0.40	0.12	0.60	0.83	0.70	0.46	0.57	0.51	0.90	1.00	0.95
SVMs-Topics+A	0.21	0.54	0.30	0.66	0.77	0.71	0.42	0.29	0.34	0.91	1.00	0.95
SVMs-Text	0.17	0.90	0.29	0.30	0.50	0.38	0.10	0.01	0.02	0.65	0.21	0.32
SVMs-Text+A	0.24	0.87	0.38	0.66	0.85	0.74	0.07	0.01	0.02	0.74	0.13	0.22
Distributed Representations												
KNN-2	0.61	0.41	0.49	0.30	0.64	0.41	0.55	0.89	0.68	0.46	0.96	0.62
KNN-3	0.84	0.32	0.46	0.58	0.65	0.61	0.88	0.88	**0.88**	0.72	0.94	0.81
KNN-5	0.79	0.28	0.41	0.57	0.65	0.61	0.87	0.83	0.85	0.73	0.94	0.82
KNN-10	0.83	0.30	0.44	0.28	0.68	0.40	0.85	0.83	0.84	0.78	0.94	0.85
Linear SVM	0.77	0.50	0.60	0.72	0.84	0.77	0.81	0.91	0.86	0.73	0.96	0.83
Grid SVM	0.80	0.51	**0.62**	0.83	0.89	**0.85**	0.80	0.91	0.85	0.72	0.96	0.82
SGD SVM	0.70	0.40	0.51	0.73	0.79	0.76	0.85	0.91	**0.88**	0.61	0.95	0.74

Table 3: Results in terms of precision (P.), recall (R.) and f1-score (F1) on the four chosen expressions. The results of (Peng et al., 2014) are those of the multi-paragraphs method. The bold values indicates the best results for that expression in terms of f1-score.

Section 5.2 we present the results for the "general' classifier approach.

5.1 Per-Expression Classification

The averaged results over 10 runs in terms of precision, recall and f1-score are presented in Table 3. When calculating these metrics, we considered the positive class to be the "I" (idiomatic) label. We used McNemar's test (McNemar, 1947) to check the statistical significance of our models' results and found all our results to be significant at $p < 0.05$.

We can see in Table 3 that some of our models outperform the baselines on 1 expression (*BlowWhistle*) and achieved the same f1-scores on 2 expressions (*LoseHead* and *MakeScene*). For theses 3 expressions, our best models generally had higher precision than the baselines, finding more idioms on the test sets. In addition, for *MakeScene*, 2 of our models achieved the same f1-scores (*KNN-3* and *SGD-SVM-PE*), although they have different precision and recall.

The only expression on which a baseline model outperformed all our models was *TakeHeart* where it achieved higher precision, recall and f1-scores. Nevertheless, this expression had the most imbalanced test set, with roughly 9 times more idioms than literal samples. Therefore, if the baseline label all the test set samples as idiomatic (including the literal examples), it would still have the best results. It is thus worth emphasizing that the choices of distributions for training and test sets in Peng et al's work seems arbitrary and does not reflect the real distribution of the data in a balanced corpus. Also, Peng et al. (2014) did not provide the confusion matrices for their models so we cannot analyse their model behaviour across the classes.

That aside, while our best models share the same f1-score with the baseline on 2 of the expressions, we believe that our method is more powerful if we take into account that we do not explicitly access the context surrounding our input sentences. We can also consider that our method is cheaper than the baseline in the sense that we do not need to process words other than the words in the input sentence.

In addition, we note that the SVMs generally outperform the KNNs, although no single model perform best across all expressions. Regardless of the fact that the KNN-3 achieved the same f1-score as SGD-SVM on *MakeScene*, the SVM consistently scored higher than the KNNs on all expressions. This is an interesting finding if we consider that our feature vector is 4800-dimensional and the SVMs are projecting these features into a space that has much more than 4800 dimensions and not incurring into the "curse of dimensionality". Furthermore, other work using Sent2vec have shown the capabilities of the Sent2Vec representations to capture features that are suited to various NLP tasks where semantics is involved (e.g., paraphrase detection and semantic relatedness (Kiros et al., 2015)). These results together with our findings suggests that the factors in-

Expressions	Linear-SVM-GE			Grid-SVM-GE			SGD-SVM-GE		
	P.	R.	F1	P.	R.	F1	P.	R.	F1
BlowTop	0.91	0.96	0.94	0.91	0.93	0.94	0.80	0.98	0.88
BlowTrumpet	0.98	0.88	0.93	0.98	0.88	0.93	0.89	0.93	0.90
BlowWhistle*	0.84	0.67	0.75	0.84	0.68	0.75	0.67	0.59	0.63
CutFigure	0.91	0.85	0.88	0.89	0.85	0.87	0.86	0.85	0.86
FindFoot	0.96	0.93	0.94	0.97	0.93	0.95	0.85	0.90	0.87
GetNod	0.98	0.91	0.95	0.98	0.91	0.95	0.91	0.91	0.91
GetSack	0.87	0.89	0.88	0.86	0.88	0.87	0.81	0.89	0.84
GetWind	0.86	0.82	0.84	0.92	0.85	0.88	0.69	0.81	0.75
HaveWord	0.99	0.89	0.94	0.99	0.89	0.94	0.95	0.91	0.93
HitRoad	0.86	0.98	0.92	0.89	0.98	0.93	0.83	0.98	0.90
HitRoof	0.88	0.88	0.88	0.92	0.88	0.90	0.80	0.83	0.82
HitWall	0.74	0.58	0.65	0.74	0.58	0.65	0.74	0.45	0.56
HoldFire	1.00	0.63	0.77	1.00	0.63	0.77	0.82	0.67	0.74
KickHeel	0.92	0.96	0.94	0.92	0.99	0.95	0.89	0.92	0.91
LoseHead*	0.78	0.66	0.72	0.75	0.64	0.69	0.75	0.67	0.71
LoseThread	1.00	0.88	0.93	1.00	0.86	0.92	0.81	0.85	0.83
MakeFace	0.70	0.83	0.76	0.69	0.76	0.72	0.62	0.81	0.70
MakeHay	0.81	0.78	0.79	0.81	0.84	0.82	0.73	0.76	0.75
MakeHit	0.10	0.54	0.70	0.10	0.54	0.70	0.85	0.55	0.67
MakeMark	0.99	0.92	0.95	0.98	0.91	0.94	0.93	0.93	0.93
MakePile	0.84	0.67	0.74	0.84	0.70	0.76	0.74	0.70	0.72
MakeScene*	0.92	0.84	0.88	0.92	0.81	0.86	0.78	0.81	0.79
PullLeg	0.79	0.71	0.75	0.82	0.72	0.77	0.75	0.70	0.72
PullPlug	0.91	0.91	0.91	0.91	0.91	0.91	0.90	0.92	0.91
PullPunch	0.85	0.87	0.86	0.87	0.87	0.87	0.70	0.85	0.77
PullWeight	1.00	0.96	0.98	1.00	0.96	0.98	0.89	0.93	0.93
SeeStar	0.17	0.13	0.15	0.17	0.13	0.15	0.17	0.17	0.17
TakeHeart*	0.94	0.79	0.86	0.94	0.80	0.86	0.86	0.80	0.83
Total	0.84	0.80	0.83	0.84	0.80	0.83	0.79	0.79	0.78

Table 4: Precision (P.), recall (R.) and f1-scores (F1) calculated on the expressions of the balanced part of the VNC-Tokens dataset. The expressions marked with * indicate the expressions also evaluated with the "per-expression" classifiers.

volved in distinguishing between the semantics of idiomatic and literal language are deeply entrenched in language generation and only a high-dimensional representation can enable a classifier to make that distinction. This observation also implies that the contribution of each feature (generated by the distributed representation) is very small, given the fact that we need that many dimensions and the space needed to unpack the components of literal and idiomatic language has many more dimensions than the input space. Therefore, the current manually engineered features (i.e., the features used in previous idiom token classification) are only capturing a small portion of these dimensions and assigning more weight to these dimensions while other dimensions (not captured) are not considered (i.e., as they are not considered, the features represented by these dimensions have their weight equal to 0)

Another point for consideration is the fact that the combination of our model with the work of Peng et al. (2014) may result in a stronger model on this "per-expression" setting. Nevertheless, as previously highlighted, it was not possible for us to directly re-implement their work.

5.2 General Classification

Moving on to the general classification case, we present the average results (in terms of precision, recall and f1-score) over 10 runs to our "general" classifiers on the balanced part of the VNC-Tokens dataset. Once again, the positive class is assumed to be the "I" (idiomatic) label and we split the outcomes per expression. It should be noted that the "per-expression" evaluation was performed using a balanced set to train the classifiers while in this experiment we maintained the ratio of idiomatic to literal usages for each expression across the training and test sets. Our motivation for maintaining this ratio was to simulate the real distribution of the classes in the corpus.

We present results for the four individual MWEs used in the per-sentence based evaluation as well as a set of averages made over all 28 expression in the "balanced" portion of the dataset. Referring to the results we first of all note the overall performance of the "general" classifiers is

fairly high with 2 classifiers (Linear-SVM-GE and Grid-SVM-GE) sharing the same precision, recall and f1-scores. While averages here are the same across the two classifiers, it is worth noting that deviations occured across individual MWE types, though these deviations balanced out across the data set. Although not displayed in this table due to space limitations, it should be noted that all the 3 classifier had a extremely low performance on *SeeStar* (f1 = 0.15, 0.15 and 0.17 respectively).

If we compare the performance of the 4 expressions analysed in the "per-expression" experiment we can observe that all the "general" classifiers had a better performance over *BlowWhistle* and the Linear-SVM-GE also performed better on *MakeScene*. Nevertheless we should emphasize that the "general" classifier's evaluation is closer to what we would expect in a real data distribution than the evaluation presented on the "per-expression" section. This does not invalidate the evaluation of the latter but when we have access to a real data distribution it should also be taken into account when performing a ML evaluation.

In general, the results look promising. It is interesting to see how the classifiers trained on a set of mixed expressions ("general" classifiers) had a performance close to the "per-expression" classifiers, even though the latter were trained and tested on "artificial" training and test sets that do not reflect the real data distributions. We believe that these results indicate that the distributed representations generated by Sent2Vec are indeed clustering together sentences within the same class (idiomatic or literal) in feature space.

6 Conclusions and Future Work

In this paper we have investigated the use of distributed compositional semantics in literal and idiomatic language classification, more specifically using Skip-Thought Vectors (Sent2Vec). We followed the intuition that the distributed representations generated by Sent2Vec also include information regarding the context where the potential idiomatic expression is inserted and therefore is sufficient for distinguishing between idiomatic and literal language use.

We tested this approach with different Machine Learning (ML) algorithms (K-Nearest Neighbours and Support Vector Machines) and compared our work against a topic model representation that include the full paragraph or the surrounding paragraphs where the potential idiom is inserted. We have shown that using the Sent2Vec representations our classifiers achieve better results in 3 out of 4 expressions tested. We have also shown that our models generally present better precision and/or recall than the baselines.

We also investigated the capability of Sent2Vec clustering representations of sentences within the same class in feature space. We followed the intuition presented by previous experiments with distributed representations that words with similar meaning are clustered together in feature space and experimented with a "general" classifier that is trained on a dataset of mixed expressions. We have shown that the "general" classifier is feasible but the traditional "per-expression" does achieve better results in some cases.

In future work we plan to investigate the use of Sent2Vec to encode larger samples of text - not only the sentence containing idioms. We also plan to further analyse the errors made by our "general" model and investigate the "general" approach on the skewed part of the VNC-tokens dataset. We also plan to investigate an end-to-end approach based on deep learning-based representations to classify literal and idiomatic language use.

In addition, we also plan to compare our work to the method of Sporleder et al. (2010) as well apply our work on the IDX Corpus (Sporleder et al., 2010) and to other languages. The focus of these future experiments will be to test how our approach which is relatively less dependent on NLP resources compares with these other methods for idiom token classification.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and feedback. Giancarlo D. Salton would like to thank CAPES ("Coordenação de Aperfeiçoamento de Pessoal de Nível Superior") for his Science Without Borders scholarship, proc n. 9050-13-2.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of the 19th International Conference on Computational Statistics (COMPSTAT'2010)*, pages 177–187.

Lou Burnard. 2007. Reference guide for the british national corpus (xml edition). Technical report, http://www.natcorp.ox.ac.uk/.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Paul Cook, Afsaneh Fazly, and Suzanne Stevenson. 2008. The VNC-Tokens Dataset. In *Proceedings of the LREC Workshop: Towards a Shared Task for Multiword Expressions (MWE 2008)*, Marrakech, Morocco.

Afsanesh Fazly, Paul Cook, and Suzanne Stevenson. 2009. Unsupervised type and token identification of idiomatic expressions. In *Computational Linguistics*, volume 35, pages 61–103.

Anna Feldman and Jing Peng. 2013. Automatic detection of idiomatic clauses. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'13, pages 435–446.

Chikara Hashimoto and Daisuke Kawahara. 2008. Construction of an idiom corpus and its application to idiom identification based on wsd incorporating idiom-specific features. In *Proceedings of the conference on empirical methods in natural language processing*, pages 992–1001. Association for Computational Linguistics.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. June.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28*, pages 3276–3284.

Linlin Li and Caroline Sporleder. 2010a. Linguistic cues for distinguishing literal and non-literal usages. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 683–691.

Linlin Li and Caroline Sporleder. 2010b. Using gaussian mixture models to detect figurative language in context. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 297–300, Stroudsburg, PA, USA. Association for Computational Linguistics.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *The 2013 Conference of the North Americal Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 746–751.

Jing Peng, Anna Feldman, and Ekaterina Vylomova. 2014. Classifying idiomatic and literal expressions using topic models and intensity of emotions. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2019–2027, October.

Giancarlo D. Salton, Robert J. Ross, and John D. Kelleher. 2014. An Empirical Study of the Impact of Idioms on Phrase Based Statistical Machine Translation of English to Brazilian-Portuguese. In *Third Workshop on Hybrid Approaches to Translation (HyTra) at 14th Conference of the European Chapter of the Association for Computational Linguistics*.

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

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 754–762.

Caroline Sporleder, Linlin Li, Philip Gorinski, and Xaver Koch. 2010. Idioms in context: The idix corpus. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC-2010)*, pages 639–646.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA.

Aline Villavicencio, Francis Bond, Anna Korhonen, and Diana McCarthy. 2005. Editorial: Introduction to the special issue on multiword expressions: Having a crack at a hard nut. *Comput. Speech Lang.*, 19(4):365–377.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724*.

Adaptive Joint Learning of Compositional and Non-Compositional Phrase Embeddings

Kazuma Hashimoto and Yoshimasa Tsuruoka
The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
{hassy,tsuruoka}@logos.t.u-tokyo.ac.jp

Abstract

We present a novel method for jointly learning compositional and non-compositional phrase embeddings by adaptively weighting both types of embeddings using a compositionality scoring function. The scoring function is used to quantify the level of compositionality of each phrase, and the parameters of the function are jointly optimized with the objective for learning phrase embeddings. In experiments, we apply the adaptive joint learning method to the task of learning embeddings of transitive verb phrases, and show that the compositionality scores have strong correlation with human ratings for verb-object compositionality, substantially outperforming the previous state of the art. Moreover, our embeddings improve upon the previous best model on a transitive verb disambiguation task. We also show that a simple ensemble technique further improves the results for both tasks.

1 Introduction

Representing words and phrases in a vector space has proven effective in a variety of language processing tasks (Pham et al., 2015; Sutskever et al., 2014). In most of the previous work, phrase embeddings are computed from word embeddings by using various kinds of composition functions. Such composed embeddings are called *compositional embeddings*. An alternative way of computing phrase embeddings is to treat phrases as single units and assigning a unique embedding to each candidate phrase (Mikolov et al., 2013; Yazdani et al., 2015). Such embeddings are called *non-compositional embeddings*.

Relying solely on non-compositional embeddings has the obvious problem of data sparsity (i.e. rare or unknown phrase problems). At the same time, however, using compositional embeddings is not always the best option since some phrases are inherently non-compositional. For example, the phrase "bear fruits" means "to yield results"[1] but it is hard to infer its meaning by composing the meanings of "bear" and "fruit". Treating all phrases as compositional also has a negative effect in learning the composition function because the words in those idiomatic phrases are not just uninformative but can serve as noisy samples in the training. These problems have motivated us to adaptively combine both types of embeddings.

Most of the existing methods for learning phrase embeddings can be divided into two approaches. One approach is to learn compositional embeddings by regarding all phrases as compositional (Pham et al., 2015; Socher et al., 2012). The other approach is to learn both types of embeddings separately and use the better ones (Kartsaklis et al., 2014; Muraoka et al., 2014). Kartsaklis et al. (2014) show that non-compositional embeddings are better suited for a phrase similarity task, whereas Muraoka et al. (2014) report the opposite results on other tasks. These results suggest that we should not stick to either of the two types of embeddings unconditionally and could learn better phrase embeddings by considering the compositionality levels of the individual phrases in a more flexible fashion.

In this paper, we propose a method that jointly learns compositional and non-compositional embeddings by adaptively weighting both types of phrase embeddings using a compositionality scoring function. The scoring function is used to quantify the level of compositionality of each phrase

[1]The definition is found at http://idioms.thefreedictionary.com/bear+fruit.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 205–215,
Berlin, Germany, August 7-12, 2016.

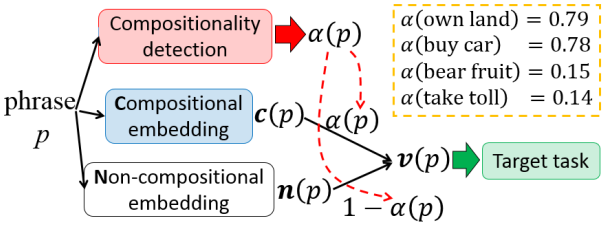


Figure 1: The overview of our method and examples of the compositionality scores. Given a phrase p, our method first computes the compositionality score $\alpha(p)$ (Eq. (3)), and then computes the phrase embedding $\boldsymbol{v}(p)$ using the compositional and non-compositional embeddings, $\boldsymbol{c}(p)$ and $\boldsymbol{n}(p)$, respectively (Eq. (2)).

and learned in conjunction with the target task for learning phrase embeddings. In experiments, we apply our method to the task of learning transitive verb phrase embeddings and demonstrate that it allows us to achieve state-of-the-art performance on standard datasets for compositionality detection and verb disambiguation.

2 Method

In this section, we describe our approach in the most general form, without specifying the function to compute the compositional embeddings or the target task for optimizing the embeddings.

Figure 1 shows the overview of our proposed method. At each iteration of the training (i.e. gradient calculation) of a certain target task (e.g. language modeling or sentiment analysis), our method first computes a compositionality score for each phrase. Then the score is used to weight the compositional and non-compositional embeddings of the phrase in order to compute the expected embedding of the phrase which is to be used in the target task. Some examples of the compositionality scores are also shown in the figure.

2.1 Compositional Phrase Embeddings

The compositional embedding $\boldsymbol{c}(p) \in \mathbb{R}^{d \times 1}$ of a phrase $p = (w_1, \cdots, w_L)$ is formulated as

$$\boldsymbol{c}(p) = f(\boldsymbol{v}(w_1), \cdots, \boldsymbol{v}(w_L)), \qquad (1)$$

where d is the dimensionality, L is the phrase length, $\boldsymbol{v}(\cdot) \in \mathbb{R}^{d \times 1}$ is a word embedding, and $f(\cdot)$ is a composition function. The function can be simple ones such as element-wise addition or multiplication (Mitchell and Lapata, 2008). More complex ones such as recurrent neural networks (Sutskever et al., 2014) are also commonly used. The word embeddings and the composition function are jointly learned on a certain target task. Since compositional embeddings are built on word-level (i.e. unigram) information, they are less prone to the data sparseness problem.

2.2 Non-Compositional Phrase Embeddings

In contrast to the compositional embedding, the non-compositional embedding of a phrase $\boldsymbol{n}(p) \in \mathbb{R}^{d \times 1}$ is independently parameterized, i.e., the phrase p is treated just like a single word. Mikolov et al. (2013) show that non-compositional embeddings are preferable when dealing with idiomatic phrases. Some recent studies (Kartsaklis et al., 2014; Muraoka et al., 2014) have discussed the (dis)advantages of using compositional or non-compositional embeddings. However, in most cases, a phrase is neither completely compositional nor completely non-compositional. To the best of our knowledge, there is no method that allows us to jointly learn both types of phrase embeddings by incorporating the levels of compositionality of the phrases as real-valued scores.

2.3 Adaptive Joint Learning

To simultaneously consider both compositional and non-compositional aspects of each phrase, we compute a phrase embedding $\boldsymbol{v}(p)$ by adaptively weighting $\boldsymbol{c}(p)$ and $\boldsymbol{n}(p)$ as follows:

$$\boldsymbol{v}(p) = \alpha(p)\boldsymbol{c}(p) + (1 - \alpha(p))\boldsymbol{n}(p), \qquad (2)$$

where $\alpha(\cdot)$ is a scoring function that quantifies the compositionality levels, and outputs a real value ranging from 0 to 1. What we expect from the scoring function is that large scores indicate high levels of compositionality. In other words, when $\alpha(p)$ is close to 1, the compositional embedding is mainly considered, and vice versa. For example, we expect $\alpha(\text{buy car})$ to be large and $\alpha(\text{bear fruit})$ to be small as shown in Figure 1.

We parameterize the scoring function $\alpha(p)$ as logistic regression:

$$\alpha(p) = \sigma(\boldsymbol{W} \cdot \boldsymbol{\phi}(p)), \qquad (3)$$

where $\boldsymbol{\phi}(p) \in \mathbb{R}^{N \times 1}$ is a feature vector of the phrase p, $\boldsymbol{W} \in \mathbb{R}^{N \times 1}$ is a weight vector, N is the number of features, and $\sigma(\cdot)$ is the logistic function. The weight vector $\boldsymbol{W}$ is jointly optimized in conjunction with the objective J for the target task of learning phrase embeddings $\boldsymbol{v}(p)$.

Updating the model parameters Given the partial derivative $\boldsymbol{\delta}_p = \frac{\partial J}{\partial \boldsymbol{v}(p)} \in \mathbb{R}^{d \times 1}$ for the target task, we can compute the partial derivative for updating $\boldsymbol{W}$ as follows:

$$\delta_\alpha = \alpha(p)(1-\alpha(p))\{\boldsymbol{\delta}_p \cdot (\boldsymbol{c}(p) - \boldsymbol{n}(p))\} \quad (4)$$

$$\frac{\partial J}{\partial \boldsymbol{W}} = \delta_\alpha \boldsymbol{\phi}(p). \quad (5)$$

If $\boldsymbol{\phi}(p)$ is not constructed by static features but is computed by a feature learning model such as neural networks, we can propagate the error term δ_α into the feature learning model by the following equation:

$$\frac{\partial J}{\partial \boldsymbol{\phi}(p)} = \delta_\alpha \boldsymbol{W}. \quad (6)$$

When we use only static features, as in this work, we can simply compute the partial derivatives of J with respect to $\boldsymbol{c}(p)$ and $\boldsymbol{n}(p)$ as follows:

$$\frac{\partial J}{\partial \boldsymbol{c}(p)} = \alpha(p)\boldsymbol{\delta}_p \quad (7)$$

$$\frac{\partial J}{\partial \boldsymbol{n}(p)} = (1-\alpha(p))\boldsymbol{\delta}_p. \quad (8)$$

As mentioned above, Eq. (7) and (8) show that the non-compositional embeddings are mainly updated when $\alpha(p)$ is close to 0, and vice versa. The partial derivative $\frac{\partial J}{\partial \boldsymbol{c}(p)}$ is used to update the model parameters in the composition function via the backpropagation algorithm. Any differentiable composition functions can be used in our method.

Expected behavior of our method The training of our method depends on the target task; that is, the model parameters are updated so as to minimize the cost function as described above. More concretely, $\alpha(p)$ for each phrase p is adaptively adjusted so that the corresponding parameter updates contribute to minimizing the cost function. As a result, different phrases will have different $\alpha(p)$ values depending on their compositionality. If the size of the training data were almost infinitely large, $\alpha(p)$ for all phrases would become nearly zero, and the non-compositional embeddings $\boldsymbol{n}(p)$ are dominantly used (since that would allow the model to better fit the data). In reality, however, the amount of the training data is limited, and thus the compositional embeddings $\boldsymbol{c}(p)$ are effectively used to overcome the data sparseness problem.

3 Learning Verb Phrase Embeddings

This section describes a particular instantiation of our approach presented in the previous section, focusing on the task of learning the embeddings of transitive verb phrases.

3.1 Word and Phrase Prediction in Predicate-Argument Relations

Acquisition of selectional preference using embeddings has been widely studied, where word and/or phrase embeddings are learned based on syntactic links (Bansal et al., 2014; Hashimoto and Tsuruoka, 2015; Levy and Goldberg, 2014; Van de Cruys, 2014). As with language modeling, these methods perform word (or phrase) prediction using (syntactic) contexts.

In this work, we focus on verb-object relationships and employ a phrase embedding learning method presented in Hashimoto and Tsuruoka (2015). The task is a plausibility judgment task for predicate-argument tuples. They extracted Subject-Verb-Object (*SVO*) and SVO-Preposition-Noun (*SVOPN*) tuples using a probabilistic HPSG parser, *Enju* (Miyao and Tsujii, 2008), from the training corpora. Transitive verbs and prepositions are extracted as predicates with two arguments. For example, the extracted tuples include (S, V, O) = ("importer", "make", "payment") and (SVO, P, N) = ("importer make payment", "in", "currency"). The task is to discriminate between observed and unobserved tuples, such as the (S, V, O) tuple mentioned above and (S, V', O) = ("importer", "*eat*", "payment"), which is generated by replacing "make" with "eat". The (S, V', O) tuple is unlikely to be observed.

For each tuple (p, a_1, a_2) observed in the training data, a cost function is defined as follows:

$$\begin{aligned} -\log\sigma(s(p,a_1,a_2)) &- \log\sigma(-s(p',a_1,a_2)) \\ &- \log\sigma(-s(p,a_1',a_2)) \\ &- \log\sigma(-s(p,a_1,a_2')), \end{aligned} \quad (9)$$

where $s(\cdot)$ is a plausibility scoring function, and p, a_1 and a_2 are a predicate and its arguments, respectively. Each of the three unobserved tuples (p', a_1, a_2), (p, a_1', a_2), and (p, a_1, a_2') is generated by replacing one of the entries with a random sample.

In their method, each predicate p is represented with a matrix $\boldsymbol{M}(p) \in \mathbb{R}^{d \times d}$ and each argument a with an embedding $\boldsymbol{v}(a) \in \mathbb{R}^{d \times 1}$. The matrices and embeddings are learned by minimizing the cost function using *AdaGrad* (Duchi et al., 2011). The scoring function is parameterized as

$$s(p, a_1, a_2) = \boldsymbol{v}(a_1) \cdot (\boldsymbol{M}(p)\boldsymbol{v}(a_2)), \quad (10)$$

and the VO and SVO embeddings are computed as

$$\boldsymbol{v}(VO) = \boldsymbol{M}(V)\boldsymbol{v}(O) \quad (11)$$

$$\boldsymbol{v}(SVO) = \boldsymbol{v}(S) \odot \boldsymbol{v}(VO), \quad (12)$$

as proposed by Kartsaklis et al. (2012). The operator $\odot$ denotes element-wise multiplication. In summary, the scores are computed as

$$s(V, S, O) = \boldsymbol{v}(S) \cdot \boldsymbol{v}(VO) \quad (13)$$

$$s(P, SVO, N) = \boldsymbol{v}(SVO) \cdot (\boldsymbol{M}(P)\boldsymbol{v}(N)). \quad (14)$$

With this method, the word and composed phrase embeddings are jointly learned based on co-occurrence statistics of predicate-argument structures. Using the learned embeddings, they achieved state-of-the-art accuracy on a transitive verb disambiguation task (Grefenstette and Sadrzadeh, 2011).

3.2 Applying the Adaptive Joint Learning

In this section, we apply our adaptive joint learning method to the task described in Section 3.1. We here redefine the computation of $\boldsymbol{v}(VO)$ by first replacing $\boldsymbol{v}(VO)$ in Eq. (11) with $\boldsymbol{c}(VO)$ as,

$$\boldsymbol{c}(VO) = \boldsymbol{M}(V)\boldsymbol{v}(O), \quad (15)$$

and then assigning VO to p in Eq. (2) and (3):

$$\boldsymbol{v}(VO) = \alpha(VO)\boldsymbol{c}(VO) + (1 - \alpha(VO))\boldsymbol{n}(VO), \quad (16)$$

$$\alpha(VO) = \sigma(\boldsymbol{W} \cdot \boldsymbol{\phi}(VO)). \quad (17)$$

The $\boldsymbol{v}(VO)$ in Eq. (16) is used in Eq. (12) and (13). We assume that the candidates of the phrases are given in advance. For the phrases not included in the candidates, we set $\boldsymbol{v}(VO) = \boldsymbol{c}(VO)$. This is analogous to the way a human guesses the meaning of an idiomatic phrase she does not know. We should note that $\boldsymbol{\phi}(VO)$ can be computed for phrases not included in the candidates, using partial features among the features described below. If any features do not fire, $\boldsymbol{\phi}(VO)$ becomes 0.5 according to the logistic function.

For the feature vector $\boldsymbol{\phi}(VO)$, we use the following simple binary and real-valued features:

- indices of V, O, and VO
- frequency and Pointwise Mutual Information (PMI) values of VO.

More concretely, the first set of the features (indices of V, O, and VO) is the concatenation of traditional one-hot vectors. The second set of features, frequency and PMI (Church and Hanks, 1990) features, have proven effective in detecting the compositionality of transitive verbs in McCarthy et al. (2007) and Venkatapathy and Joshi (2005). Given the training corpus, the frequency feature for a VO pair is computed as

$$freq(VO) = \log(count(VO)), \quad (18)$$

where $count(VO)$ counts how many times the VO pair appears in the training corpus, and the PMI feature is computed as

$$\mathrm{PMI}(VO) = \log \frac{count(VO)count(*)}{count(V)count(O)}, \quad (19)$$

where $count(V)$, $count(O)$, and $count(*)$ are the counts of the verb V, the object O, and all VO pairs in the training corpus, respectively. We normalize the frequency and PMI features so that their maximum absolute value becomes 1.

4 Experimental Settings

4.1 Training Data

As the training data, we used two datasets, one small and one large: the British National Corpus (BNC) (Leech, 1992) and the English Wikipedia. More concretely, we used the publicly available data[2] preprocessed by Hashimoto and Tsuruoka (2015). The BNC data consists of 1.38 million SVO tuples and 0.93 million SVOPN tuples. The Wikipedia data consists of 23.6 million SVO tuples and 17.3 million SVOPN tuples. Following the provided code[3], we used exactly the same train/development/test split (0.8/0.1/0.1) for training the overall model. As the third training data, we also used the concatenation of the two data, which is hereafter referred to as *BNC-Wikipedia*.

We applied our adaptive joint learning method to verb-object phrases observed more than K times in each corpus. K was set to 10 for the BNC data and 100 for the Wikipedia and BNC-Wikipedia data. Consequently, the non-compositional embeddings were assigned to 17,817, 28,933, and 30,682 verb-object phrase types in the BNC, Wikipedia, and BNC-Wikipedia data, respectively.

[2] http://www.logos.t.u-tokyo.ac.jp/~hassy/publications/cvsc2015/

[3] https://github.com/hassyGo/SVOembedding

4.2 Training Details

The model parameters consist of d-dimensional word embeddings for nouns, non-compositional phrase embeddings, $d \times d$-dimensional matrices for verbs and prepositions, and a weight vector $\boldsymbol{W}$ for $\alpha(VO)$. All the model parameters are jointly optimized. We initialized the embeddings and matrices with zero-mean gaussian random values with a variance of $\frac{1}{d}$ and $\frac{1}{d^2}$, respectively, and $\boldsymbol{W}$ with zeros. Initializing $\boldsymbol{W}$ with zeros forces the initial value of each $\alpha(VO)$ to be 0.5 since we use the logistic function to compute $\alpha(VO)$.

The optimization was performed via mini-batch AdaGrad (Duchi et al., 2011). We fixed d to 25 and the mini-batch size to 100. We set candidate values for the learning rate ε to $\{0.01, 0.02, 0.03, 0.04, 0.05\}$. For the weight vector $\boldsymbol{W}$, we employed L2-norm regularization and set the coefficient λ to $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 0\}$. For selecting the hyperparameters, each training process was stopped when the evaluation score on the development split decreased. Then the best performing hyperparameters were selected for each training dataset. Consequently, ε was set to 0.05 for all training datasets, and λ was set to 10^{-6}, 10^{-3}, and 10^{-5} for the BNC, Wikipedia, and BNC-Wikipedia data, respectively. Once the training is finished, we can use the learned embeddings and the scoring function in downstream target tasks.

5 Evaluation on the Compositionality Detection Function

5.1 Evaluation Settings

Datasets First, we evaluated the learned compositionality detection function on two datasets, **VJ'05**[4] and **MC'07**[5], provided by Venkatapathy and Joshi (2005) and McCarthy et al. (2007), respectively. VJ'05 consists of 765 verb-object pairs with human ratings for the compositionality. MC'07 is a subset of VJ'05 and consists of 638 verb-object pairs. For example, the rating of "buy car" is 6, which is the highest score, indicating the phrase is highly compositional. The rating of "bear fruit " is 1, which is the lowest score, indicating the phrase is highly non-compositional.

[4] http://www.dianamccarthy.co.uk/downloads/SVAJ2005compositionality_rating.txt

[5] http://www.dianamccarthy.co.uk/downloads/emnlp2007data.txt

Method	MC'07	VJ'05
Proposed method (Wikipedia)	0.508	0.514
Proposed method (BNC)	0.507	0.507
Proposed method (BNC-Wikipedia)	0.518	0.527
Proposed method (Ensemble)	0.550	0.552
Kiela and Clark (2013) w/ WordNet	n/a	0.461
Kiela and Clark (2013)	n/a	0.420
DSPROTO (McCarthy et al., 2007)	0.398	n/a
PMI (McCarthy et al., 2007)	0.274	n/a
Frequency (McCarthy et al., 2007)	0.141	n/a
DSPROTO+ (McCarthy et al., 2007)	0.454	n/a
Human agreement	0.702	0.716

Table 1: Compositionality detection task.

Evaluation metric The evaluation was performed by calculating Spearman's rank correlation scores[6] between the averaged human ratings and the learned compositionality scores $\alpha(VO)$.

Ensemble technique We also produced the result by employing an *ensemble* technique. More concretely, we used the averaged compositionality scores from the results of the BNC and Wikipedia data for the ensemble result.

5.2 Results and Discussion

5.2.1 Result Overview

Table 1 shows our results and the state of the art. Our method outperforms the previous state of the art in all settings. The result denoted as *Ensemble* is the one that employs the ensemble technique, and achieves the strongest correlation with the human-annotated datasets. Even without the ensemble technique, our method performs better than all of the previous methods.

Kiela and Clark (2013) used window-based co-occurrence vectors and improved their score using WordNet hypernyms. By contrast, our method does not rely on such external resources, and only needs parsed corpora. We should note that Kiela and Clark (2013) reported that their score did not improve when using parsed corpora. Our method also outperforms DSPROTO+, which used a small amount of the labeled data, while our method is fully unsupervised.

We calculated confidence intervals ($P < 0.05$) using bootstrap resampling (Noreen, 1989). For example, for the results using the BNC-Wikipedia data, the intervals on MC'07 and VJ'05 are (0.455, 0.574) and (0.475, 0.579), respectively. These results show that our method significantly outperforms the previous state-of-the-art results.

[6] We used the Scipy 0.12.0 implementation in Python.

Phrase		Gold standard	(a) BNC	(b) Wikipedia	BNC-Wikipedia	Ensemble ((a)+(b))×0.5
(A)	buy car	6	0.78	0.71	0.80	0.74
	own land	6	0.79	0.73	0.76	0.76
	take toll	1.5	0.14	0.11	0.06	0.13
	shed light	1	0.21	0.07	0.07	0.14
	bear fruit	1	0.15	0.19	0.17	0.17
(B)	make noise	6	0.37	0.33	0.30	0.35
	have reason	5	0.26	0.39	0.33	0.33
(C)	smoke cigarette	6	0.56	0.90	0.78	0.73
	catch eye	1	0.48	0.14	0.17	0.31

Table 2: Examples of the compositionality scores.

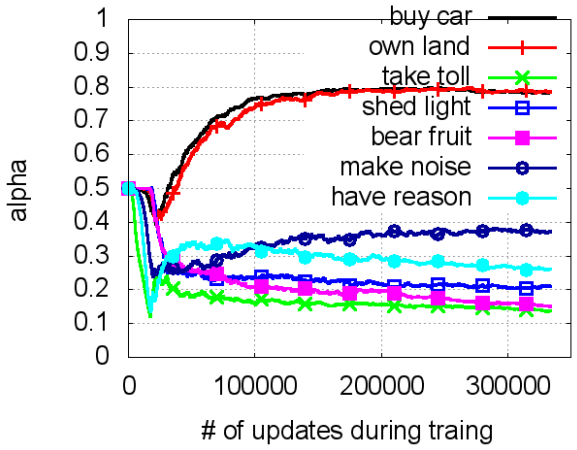


Figure 2: Trends of $\alpha(VO)$ during the training on the BNC data.

Highest average scores		Lowest average scores	
approve	0.83	bear	0.37
reject	0.72	play	0.38
discuss	0.71	have	0.38
visit	0.70	make	0.39
want	0.70	break	0.40
describe	0.70	take	0.40
involve	0.69	raise	0.41
own	0.68	reach	0.41
attend	0.68	gain	0.42
reflect	0.67	draw	0.42

Table 3: The 10 highest and lowest average compositionality scores with the corresponding verbs on the BNC data.

5.2.2 Analysis of Compositionality Scores

Figure 2 shows how $\alpha(VO)$ changes for the seven phrases during the training on the BNC data. As shown in the figure, starting from 0.5, $\alpha(VO)$ for each phrase converges to its corresponding value. The differences in the trends indicate that our method can adaptively learn compositionality levels for the phrases. Table 2 shows the learned compositionality scores for the three groups of the examples along with the gold-standard scores given by the annotators. The group (A) is considered to be consistent with the gold-standard scores, the group (B) is not, and the group (C) shows examples for which the difference between the compositionality scores of our results is large.

Characteristics of light verbs The verbs "take", "make", and "have" are known as *light verbs* [7], and the scoring function tends to assign low scores to light verbs. In other words, our method can recognize that the light verbs are frequently used to form idiomatic (i.e. non-compositional) phrases. To verify the assumption, we calculated the average compositionality score for each verb by averaging the compositionality scores paired with its candidate objects. Here we used 135 verbs which take more than 30 types of objects in the BNC data. Table 3 shows the 10 highest and lowest average scores with the corresponding verbs. We see that relatively low scores are assigned to the light verbs as well as other verbs which often form idiomatic phrases. As shown in the group (B) in Table 2, however, light verb phrases are not always non-compositional. Despite this, the learned function assigns low scores to compositional phrases formed by the light verbs. These results suggest that using a more flexible scoring function may further strengthen our method.

Context dependence Both our method and the two datasets, VJ'05 and MC'07, assume that the compositionality score can be computed for each phrase with no contextual information. However, in general, the compositionality level of a phrase depends on its contextual information. For example, the meaning of the idiomatic phrase "bear

[7] In Section 5.2.2 in Newton (2006), the term *light verb* is used to refer to verbs which can be used in combination with some other element where their contribution to the meaning of the whole construction is reduced in some way.

fruit" can be compositionaly interpreted as "to yield fruit" for a plant or tree. We manually inspected the BNC data to check whether the phrase "bear fruit" is used as the compositional meaning or the idiomatic meaning ("to yield results"). As a result, we have found that most of the usage was its idiomatic meaning. In the model training, our method is affected by the majority usage and fits the evaluation datasets where the phrase "bear fruit" is regarded as highly non-compositional. Incorporating contextual information into the compositionality scoring function is a promising direction of future work.

5.2.3 Effects of Ensemble

We used the two different corpora for constructing the training data, and our method achieves the state-of-the-art results in all settings. To inspect the results on VJ'05, we calculated the correlation score between the outputs from our results of the BNC and Wikipedia data. The correlation score is 0.674 and that is, the two different corpora lead to reasonably consistent results, which indicates the robustness of our method. However, the correlation score is still much lower than perfect correlation; in other words, there are disagreements between the outputs learned with the corpora. The group (C) in Table 2 shows such two examples. In these cases, the ensemble technique is helpful in improving the results as shown in the examples.

Another interesting observation in our results is that the result of the ensemble technique outperforms that of the BNC-Wikipedia data as shown in Table 1. This shows that separately using the training corpora of different nature and then performing the ensemble technique can yield better results. By contrast, many of the previous studies on embedding-based methods combine different corpora into a single dataset, or use multiple corpora just separately and compare them (Hashimoto and Tsuruoka, 2015; Muraoka et al., 2014; Pennington et al., 2014). It would be worth investigating whether the results in the previous work can be improved by ensemble techniques.

6 Evaluation on the Phrase Embeddings

6.1 Evaluation Settings

Dataset Next, we evaluated the learned embeddings on the transitive verb disambiguation dataset **GS'11**[8] provided by Grefenstette and Sadrzadeh (2011). GS'11 consists of 200 pairs of transitive verbs and each verb pair takes the same subject and object. For example, the transitive verb "run" is known as a polysemous word and this task requires one to identify the meanings of "run" and "operate" as similar to each other when taking "people" as their subject and "company" as their object. In the same setting, however, the meanings of "run" and "move" are not similar to each other. Each pair has multiple human ratings indicating how similar the phrases of the pair are.

Evaluation metric The evaluation was performed by calculating Spearman's rank correlation scores between the human ratings and the cosine similarity scores of $\boldsymbol{v}(SVO)$ in Eq. (12). Following the previous studies, we used the gold-standard ratings in two ways: averaging the human ratings for each SVO tuple (GS'11a) and treating each human rating separately (GS'11b).

Ensemble technique We used the same ensemble technique described in Section 5.1. In this task we produced two ensemble results: *Ensemble A* and *Ensemble B*. The former used the averaged cosine similarity from the results of the BNC and Wikipedia data, and the latter further incorporated the result of the BNC-Wikipedia data.

Baselines We compared our adaptive joint learning method with two baseline methods. One is the method in Hashimoto and Tsuruoka (2015) and it is equivalent to fixing $\alpha(VO)$ to 1 in our method. The other is fixing $\alpha(VO)$ to 0.5 in our method, which serves as a baseline to evaluate how effective the proposed adaptive weighting method is.

6.2 Results and Discussion

6.2.1 Result Overview

Table 4 shows our results and the state of the art, and our method outperforms almost all of the previous methods in both datasets. Again, the ensemble technique further improves the results, and overall, Ensemble B yields the best results.

The scores in Hashimoto and Tsuruoka (2015), the baseline results with $\alpha(VO) = 1$ in our method, have been the best to date. As shown in Table 4, our method outperforms the baseline results with $\alpha(VO) = 0.5$ as well as those

[8] http://www.cs.ox.ac.uk/activities/compdistmeaning/GS2011data.txt

	Proposed method		$\alpha(VO) = 1$	$\alpha(VO) = 0.5$
take toll	$\alpha(\text{take toll}) = 0.11$	put strain place strain cause strain have affect exacerbate injury	deplete division necessitate monitoring deplete pool create pollution deplete field	put strain cause lack befall army exacerbate weakness cause strain
catch eye	$\alpha(\text{catch eye}) = 0.14$	catch attention grab attention make impression lift spirit become favorite	catch ear catch heart catch e-mail catch imagination catch attention	grab attention make impression catch attention become legend inspire playing
bear fruit	$\alpha(\text{bear fruit}) = 0.19$	accentuate effect enhance beauty enhance atmosphere rejuvenate earth enhance habitat	bear herb bear grain bear spore bear variety bear seed	increase richness reduce biodiversity fuel boom enhance atmosphere worsen violence
make noise	$\alpha(\text{make noise}) = 0.33$	attack intruder attack trespasser avoid predator attack diver attack pedestrian	make sound do beating get bounce get pulse lose bit	burn can kill monster wash machine lightn flash cook raman
buy car	$\alpha(\text{buy car}) = 0.71$	buy bike buy machine buy motorcycle buy automobile purchase coins	buy truck buy bike buy automobile buy motorcycle buy vehicle	buy bike buy instrument buy chip buy scooter buy motorcycle

Table 5: Examples of the closest neighbors in the learned embedding space. All of the results were obtained by using the Wikipedia data, and the values of $\alpha(VO)$ are the same as those in Table 2.

Method	GS'11a	GS'11b
Proposed method (Wikipedia)	0.598	0.461
Proposed method (BNC)	0.595	0.463
Proposed method (BNC-Wikipedia)	0.623	0.483
Proposed method (Ensemble A)	0.661	0.511
Proposed method (Ensemble B)	0.680	0.524
$\alpha(VO) = 0.5$ (Wikipedia)	0.491	0.386
$\alpha(VO) = 0.5$ (BNC)	0.599	0.462
$\alpha(VO) = 0.5$ (BNC-Wikipedia)	0.610	0.477
$\alpha(VO) = 0.5$ (Ensemble A)	0.612	0.474
$\alpha(VO) = 0.5$ (Ensemble B)	0.638	0.495
$\alpha(VO) = 1$ (Wikipedia)	0.576	n/a
$\alpha(VO) = 1$ (BNC)	0.574	n/a
Milajevs et al. (2014)	0.456	n/a
Polajnar et al. (2014)	n/a	0.370
Hashimoto et al. (2014)	0.420	0.340
Polajnar et al. (2015)	n/a	0.330
Grefenstette and Sadrzadeh (2011)	n/a	0.210
Human agreement	0.750	0.620

Table 4: Transitive verb disambiguation task. The results for $\alpha(VO) = 1$ are reported in Hashimoto and Tsuruoka (2015).

with $\alpha(VO) = 1$. We see that our method improves the baseline scores by adaptively combining compositional and non-compositional embeddings. Along with the results in Table 1, these results show that our method allows us to improve the composition function by jointly learning non-compositional embeddings and the scoring function for compositionality detection.

6.2.2 Analysis of the Learned Embeddings

We inspected the effects of adaptively weighting the compositional and non-compositional embeddings. Table 5 shows the five closest neighbor phrases in terms of the cosine similarity for the three idiomatic phrases "take toll", "catch eye", and "bear fruit" as well as the two non-idiomatic phrases "make noise" and "buy car". The examples trained with the Wikipedia data are shown for our method and the two baselines, i.e., $\alpha(VO) = 1$ and $\alpha(VO) = 0.5$. As shown in Table 2, the compositionality levels of the first three phrases are low and their non-compositional embeddings are dominantly used to represent their meaning.

One observation with $\alpha(VO) = 1$ is that head words (i.e. verbs) are emphasized in the shown examples except "take toll" and "make noise". As with other embedding-based methods, the compositional embeddings are highly affected by their component words. As a result, the phrases consisting of the same verb and the similar objects are often listed as the closest neighbors. By contrast, our method flexibly allows us to adaptively omit the information about the component words. Therefore, our method puts more weight on capturing the idiomatic aspects of the example phrases by

adaptively using the non-compositional embeddings.

The results of $\alpha(VO) = 0.5$ are similar to those with our proposed method, but we can see some differences. For example, the phrase list for "make noise" of our proposed method captures offensive meanings, whereas that of $\alpha(VO) = 0.5$ is somewhat ambiguous. As another example, the phrase lists for "buy car" show that our method better captures the semantic similarity between the objects than $\alpha(VO) = 0.5$. This is achieved by adaptively assigning a relatively large compositionality score (0.71) to the phrase to use the information about the object "car".

We should note that "make noise" is highly compositional but our method outputs $\alpha(\text{make noise}) = 0.33$, and the phrase list of $\alpha(VO) = 1$ is the most appropriate in this case. Improving the compositionality detection function should thus further improve the learned embeddings.

7 Related Work

Learning embeddings of words and phrases has been widely studied, and the phrase embeddings have proven effective in many language processing tasks, such as machine translation (Cho et al., 2014; Sutskever et al., 2014), sentiment analysis and semantic textual similarity (Tai et al., 2015). Most of the phrase embeddings are constructed by word-level information via various kinds of composition functions like long short-term memory (Hochreiter and Schmidhuber, 1997) recurrent neural networks. Such composition functions should be powerful enough to efficiently encode information about all the words into the phrase embeddings. By simultaneously considering the compositionality of the phrases, our method would be helpful in saving the composition models from having to be powerful enough to perfectly encode the non-compositional phrases. As a first step towards this purpose, in this paper we have shown the effectiveness of our method on the task of learning verb phrase embeddings.

Many studies have focused on detecting the compositionality of a variety of phrases (Lin, 1999), including the ones on verb phrases (Diab and Bhutada, 2009; McCarthy et al., 2003) and compound nouns (Farahmand et al., 2015; Reddy et al., 2011). Compared to statistical feature-based methods (McCarthy et al., 2007; Venkatapathy and Joshi, 2005), recent methods use word and phrase embeddings (Kiela and Clark, 2013; Yazdani et al., 2015). The embedding-based methods assume that word embeddings are given in advance and as a post-processing step, learn or simply employ composition functions to compute phrase embeddings. In other words, there is no distinction between compositional and non-compositional phrases. Yazdani et al. (2015) further proposed to incorporate latent annotations (binary labels) for the compositionality of the phrases. However, binary judgments cannot consider numerical scores of the compositionality. By contrast, our method adaptively weights the compositional and non-compositional embeddings using the compositionality scoring function.

8 Conclusion and Future Work

We have presented a method for adaptively learning compositional and non-compositional phrase embeddings by jointly detecting compositionality levels of phrases. Our method achieves the state of the art on a compositionality detection task of verb-object pairs, and also improves upon the previous state-of-the-art method on a transitive verb disambiguation task. In future work, we will apply our method to other kinds of phrases and tasks.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by CREST, JST.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring Continuous Word Representations for Dependency Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 809–815.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Kenneth Church and Patrick Hanks. 1990. Word Association Norms, Mutual Information and Lexicography. *Computational Linguistics*, 19(2):263–312.

Mona Diab and Pravin Bhutada. 2009. Verb Noun Construction MWE Token Classification. In *Proceedings of the Workshop on Multiword Expressions: Identification, Interpretation, Disambiguation and Applications*, pages 17–22.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Meghdad Farahmand, Aaron Smith, and Joakim Nivre. 2015. A Multiword Expression Data Set: Annotating Non-Compositionality and Conventionalization for English Noun Compounds. In *Proceedings of the 11th Workshop on Multiword Expressions*, pages 29–33.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental Support for a Categorical Compositional Distributional Model of Meaning. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1394–1404.

Kazuma Hashimoto and Yoshimasa Tsuruoka. 2015. Learning Embeddings for Transitive Verb Disambiguation by Implicit Tensor Factorization. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 1–11.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly Learning Word Representations and Composition Functions Using Predicate-Argument Structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1544–1555.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A Unified Sentence Space for Categorical Distributional-Compositional Semantics: Theory and Experiments. In *Proceedings of the 24th International Conference on Computational Linguistics*, pages 549–558.

Dimitri Kartsaklis, Nal Kalchbrenner, and Mehrnoosh Sadrzadeh. 2014. Resolving Lexical Ambiguity in Tensor Regression Models of Meaning. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 212–217.

Douwe Kiela and Stephen Clark. 2013. Detecting Compositionality of Multi-Word Expressions using Nearest Neighbours in Vector Space Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1427–1432.

Geoffrey Leech. 1992. 100 Million Words of English: the British National Corpus. *Language Research*, 28(1):1–13.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 302–308.

Dekang Lin. 1999. Automatic Identification of Non-compositional Phrases. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 317–324.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a Continuum of Compositionality in Phrasal Verbs. In *Proceedings of the ACL 2003 Workshop on Multiword Expressions: Analysis, Acquisition and Treatment*, pages 73–80.

Diana McCarthy, Sriram Venkatapathy, and Aravind Joshi. 2007. Detecting Compositionality of Verb-Object Combinations using Selectional Preferences. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 369–379.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating Neural Word Representations in Tensor-Based Compositional Settings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 708–719.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based Models of Semantic Composition. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 236–244.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature Forest Models for Probabilistic HPSG Parsing. *Computational Linguistics*, 34(1):35–80, March.

Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. 2014. Finding The Best Model Among Representative Compositional Models. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation*, pages 65–74.

Mark Newton. 2006. *Basic English Syntax with Exercises*. Bölcsész Konzorcium.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the C-PHRASE model. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 971–981.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2014. Using Sentence Plausibility to Learn the Semantics of Transitive Verbs. In *Proceedings of Workshop on Learning Semantics at the 2014 Conference on Neural Information Processing Systems*.

Tamara Polajnar, Laura Rimell, and Stephen Clark. 2015. An Exploration of Discourse-Based Sentence Spaces for Compositional Distributional Semantics. In *Proceedings of the First Workshop on Linking Computational Models of Lexical, Sentential and Discourse-level Semantics*, pages 1–11.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An Empirical Study on Compositionality in Compound Nouns. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 210–218.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Tim Van de Cruys. 2014. A Neural Network Approach to Selectional Preference Acquisition. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–35.

Sriram Venkatapathy and Aravind Joshi. 2005. Measuring the Relative Compositionality of Verb-Noun (V-N) Collocations by Integrating Features. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 899–906.

Majid Yazdani, Meghdad Farahmand, and James Henderson. 2015. Learning Semantic Composition to Detect Non-compositionality of Multiword Expressions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1733–1742.

Metaphor Detection with Topic Transition, Emotion and Cognition in Context

Hyeju Jang, Yohan Jo, Qinlan Shen, Michael Miller,
Seungwhan Moon, Carolyn P. Rosé
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213, USA
{hyejuj,yohanj,qinlans,millerm,seungwhm,cprose}@cs.cmu.edu

Abstract

Metaphor is a common linguistic tool in communication, making its detection in discourse a crucial task for natural language understanding. One popular approach to this challenge is to capture semantic incohesion between a metaphor and the dominant topic of the surrounding text. While these methods are effective, they tend to overclassify target words as metaphorical when they deviate in meaning from its context. We present a new approach that (1) distinguishes literal and non-literal use of target words by examining sentence-level topic transitions and (2) captures the motivation of speakers to express emotions and abstract concepts metaphorically. Experiments on an online breast cancer discussion forum dataset demonstrate a significant improvement in metaphor detection over the state-of-the-art. These experimental results also reveal a tendency toward metaphor usage in personal topics and certain emotional contexts.

1 Introduction

Figurative language is commonly used in human communication ranging from literature to everyday speech. One of the most common forms of non-literal language is *metaphor*, in which two dissimilar concepts are compared. In the utterance, "Time is money" (Lakoff and Johnson, 1980), for example, the concept of "time" is compared to "money" to emphasize that time is valuable. Bringing in information from another domain allows more effective ways of expressing thoughts, feelings, and ideas than only using literal language.

Previous approaches to modeling metaphor have used either the semantic and syntactic information in just the sentence that contains a metaphor (Turney et al., 2011; Tsvetkov et al., 2014), or the context beyond a single sentence (Broadwell et al., 2013; Strzalkowski et al., 2013; Schulder and Hovy, 2014; Klebanov et al., 2015; Jang et al., 2015) to detect topical discrepancy between a candidate metaphor and the dominant theme (See Section 2 for more detailed literature review).

Although previous approaches were effective at capturing some aspects of the governing context of a metaphor, the space of how to best use the contextual information is still wide open. Previous context-based models tend to overclassify literal words as metaphorical if they find semantic contrast with the governing context. These cases manifested in the work by Schulder and Hovy (2014) and Jang et al. (2015) as high recall but low precision for metaphorical instances.

We present a new approach that uses lexical and topical context to resolve the problem of low precision on metaphor detection. To better capture the relevant context surrounding a metaphor, we approach the problem in two directions. First, we hypothesize that topic transition patterns between sentences containing metaphors and their contexts are different from that of literal sentences. To this end, we incorporate several indicators of sentence-level topic transitions as features, such as topic similarity between a sentence and its neighboring sentences, measured by Sentence LDA. Second, we observe that metaphor is often used to express speakers' emotional experiences; we therefore model a speaker's motivation in using metaphor by detecting emotion and cognition words in metaphorical and literal sentences and their contexts.

To demonstrate the efficacy of our approach, we

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 216–225,
Berlin, Germany, August 7-12, 2016.

evaluate our system on the metaphor detection task presented by Jang et al. (2015) using a breast cancer discussion forum dataset. This dataset is distinct in that it features metaphors occurring in conversational text, unlike news corpora or other formal texts typical in computational linguistics.

Our contributions are three-fold: (1) We extend the previous approaches for contextually detecting metaphor by exploring topic transitions between a metaphor and its context rather than only detecting lexical discrepancies. In addition, (2) we propose to capture emotional and cognitive content to better uncover speakers' motivation for using metaphors. Lastly, (3) through our empirical evaluation, we find that metaphor occurs more frequently around personal topics.

2 Relation to Prior Work

Research in automatic metaphor detection has spanned from detecting metaphor in limited sets of syntactic constructions to studying the use of metaphor in discourse, with approaches ranging from rule-based methods using lexical resources to statistical machine learning models. Here, we focus in particular on approaches that use context wider than a sentence for metaphor detection. For a more thorough review of metaphor processing systems, refer to Shutova (2015).

The main idea behind using context in metaphor detection is that metaphorically used words tend to violate lexical cohesion in text. Different methods, however, approach the problem of detecting semantic outliers in different ways.

Li and Sporleder (2009; 2010) identify metaphorical idioms using the idea that non-literal expressions break lexical cohesion of a text. Li and Sporleder (2009) approached the problem by constructing a *lexical cohesion graph*. In the graph, content words in a text are represented as vertices, which are connected by edges representing semantic relatedness. The intuition behind their approach was that non-literal expressions would lower the average semantic relatedness of the graph. To classify a word as literal or metaphorical, Li and Sporleder (2010) use Gaussian Mixture Models with semantic similarity features, such as the relatedness between this target word and words in its context.

Broadwell et al. (2013) and Strzalkowski et al. (2013) base their approach on the idea that metaphors are likely to be concrete words that are not semantically associated with the surrounding context. Broadwell et al. (2013) implemented this idea using *topic chains*, which consist of noun phrases that are connected by pronominal mention, repetition, synonym, or hyponym relations. Strzalkowski et al. (2013) build on this idea by taking nouns and adjectives around the target concept as candidate source relations. They filtered out candidate sources that were in the same topical chain as the target concept or were not linked to the word being classified by a direct dependency path.

Schulder and Hovy (2014) also hypothesize that novel metaphors are marked by their unusualness in a given context. They use a domain-specific *term relevance* metric, which measures how typical a term is for the domain associated with the literal usage of a word, and *common relevance*, which measures how common a word is across domains. If a term is neither typical for a text's domain nor common, it is taken as a metaphor candidate. A particular strength of this approach is its accommodation of common words without discriminative power, which often confuse context-based models.

Jang et al. (2015) model context by using both *global context*, the context of an entire post, and *local context*, the context within a sentence, in relationship to a word being classified as metaphorical or literal. They used word categories from FrameNet, topic distribution, and lexical chain information (similar in concept to the topic chain information in (Broadwell et al., 2013)) to model the contrast between a word and its global context. To model the contrast between a word and its local context, they used lexical concreteness, word categories and semantic relatedness features.

Mohler et al. (2013) built a domain-aware semantic signature for a text to capture the context surrounding a metaphorical candidate. Unlike other approaches that try to discriminate metaphors from their context, their approach uses binary classifiers to compare the semantic signature for a text with that of known metaphors.

The above approaches attempted to capture governing context in various ways and were effective when applied to the problem of metaphor detection. However, these methods tend to overclassify literal instances as metaphorical when semantic cohesion is violated within their governing contexts. Additionally, these methods could

fail to detect extended metaphors, which span over wider contexts. In this paper, we specifically focus on the problem of discriminating literal instances from metaphorical instances by expanding the scope of what is captured within a context. Like (Mohler et al., 2013), we share the intuition that there could be associations between specific metaphors and their contexts, but we relax the assumption that metaphors must be similar to known metaphors.

3 Our Approach

To better capture the distinctions between metaphorical and literal usages of the same word (target word), we approach the task in two directions. First, we model how topics in context change for both metaphorical and literal instances of a target word (Section 3.1). Second, we consider the situational context for why individuals choose to use metaphor (Section 3.2). We use multi-level modeling to combine these two types of features with the specific target word to model interactions between the features and a particular metaphor (Section 3.3).

3.1 Topic Transition

In writing, cohesion refers to the presence or absence of explicit cues in the text that allow the reader to make connections between ideas (Crossley and McNamara, 2010). For example, overlapping words and concepts between sentences indicate that the same ideas are being referred to across these sentences. Metaphorically used words tend to be semantically incohesive with the governing context. Therefore, determining semantic or topical cohesion is important for metaphor detection.

However, even if a text is literal and cohesive, not all words within the text are semantically related. In example (1), a human could easily determine that "pillows", "music", "flickering candles", and "a foot massage" share the theme of relaxation. But it is difficult to define their relatedness computationally – these terms are not synonyms, hypernyms, antonyms, or in any other well-defined lexical relation. Additionally, even if the whole sentence is correctly interpreted as ways of indulging oneself, it is still semantically contrasted with the surrounding sentences about medicine. In this example, the target word "candle" is used literally, but the contrast between the sentence containing the target word and its context makes it computationally difficult to determine that it is not metaphorical:

> (1) ... yet encouraged to hear you have a diagnosis and it's being treated. Since you have to give up your scented stuff you'll just have to figure out some very creative ways to indulge yourself. *Soft pillows, relaxing music, flickering candles, maybe a foot massage*. Let's hope your new pain relief strategy works and the Neulasta shot is not so bad . I never had Taxotere, but have read it can be much easier than AC for many people. ...

Example (2) also shows semantic inconsistency between the candidate metaphor "boat" and the surrounding sentences about medicine. However, in this example, "boat" is metaphorically used. Thus, it is difficult to determine whether a word is metaphorical or literal when there is semantic contrast because both example (1) and example (2) show semantic contrast.

> (2) When my brain mets were discovered last year, I had to see a neurosurgeon. He asked if I understood that my treatment was pallative care. *Boy, did it rock my boat to hear that phrase!* I agree with Fitz, pallative treatment is to help with pain and alleviate symptoms.....but definitely different than hospice care.

The primary difference between these two examples is in the nature of the semantic contrast. In example (1), the topic of the sentence containing "candle" is relaxation, while the topic of the previous and following sentences is medicine. The transition between medicine and relaxation tends to be more literal, whereas the transition between the topic in the sentence containing "boat" and the surrounding medical topic sentences tends to be more metaphorical.

We use these differences in the topic transition for metaphor detection. We consider topic transitions at the sentence level, rather than the word level, because people often represent an idea at or above the sentence level. Thus, topic is better-represented at the sentence level.

To model context at the sentence level, we first assign topics to each sentence using Sentence Latent Dirichlet Allocation (LDA) (Jo and Oh, 2011). Sentence LDA has two main advantages over standard LDA for our work. First, while standard LDA assumes that each word is assigned a topic derived from the topic distribution of a document, Sentence LDA makes the constraint that all words in the same sentence must be assigned the same topic. Due to this property, the generated topics are better aligned with the role or purpose of a sentence, compared to topics generated from LDA. Additionally, having each sentence assigned to one topic helps us avoid using heuristics for representing the topic of each sentence. [1]

Using Sentence LDA, we modeled four features to capture how the topic changes around the sentence where a target word resides. We refer to this sentence as the *target sentence*.

Target Sentence Topic (TargetTopic): We hypothesize that sentences containing a metaphor may prefer topics that are different from those of sentences where the same word is used literally. Hence, TargetTopic is a T-dimensional binary feature, where T is the number of topics, that indicates the topic assigned to the sentence containing the target word.

Topic Difference (TopicDiff): We hypothesize that a metaphorical sentence is more likely to be different from its neighboring sentences, in terms of topic, than a literal sentence. Therefore, TopicDiff is a two-dimensional binary feature that indicates whether the topic assigned to the target sentence is different from that of the previous and next sentences.

Topic Similarity (TopicSim): Under the same hypothesis as TopicDiff, TopicSim is a two-dimensional feature that represents the similarity between the topic of the target sentence and its previous and next sentences. Unlike TopicDiff, which is binary, TopicSim has continuous values between 0 and 1, as we use the cosine similarity between each topic's word distributions as topic similarity. Note that in Sentence LDA, all topics share the same vocabulary, but assign different probabilities to different words as in LDA although all tokens in a sentence are assigned to the same topic in Sentence LDA.

Topic Transition (TopicTrans): The topic of a metaphorical sentence may extend over multiple sentences, so a topic transition may occur a few sentences ahead or behind the target sentence. TopicTrans looks for the nearest sentences with a different topic before and after the current target sentence and encodes the topics of the different-topic sentences. Hence, TopicTrans is a $2T$-dimensional feature, where T is the number of topics, that indicates the topics of the nearest sentences that have a different topic from the target sentence.

Topic Transition Similarity (TopicTransSim): The topics before and after a transition, even in the extended case for TopicTrans, are still expected to be more different in metaphorical cases than in literal cases, as we assume for TopicSim. Therefore, TopicTransSim is a two-dimensional continuous feature that encodes the cosine similarity between the topic of the target sentence and the topics of the nearest sentences that have a different topic before and after the target sentence.

3.2 Emotion and Cognition

Metaphors are often used to explain or describe abstract ideas, such as difficult concepts or emotions (Meier and Robinson, 2005). (Fainsilber and Ortony, 1987) showed that descriptions of feelings contain more metaphorical language than descriptions of behavior.

In our domain, writers are searching for support through the emotionally tumultuous experience of breast cancer and often turn to metaphor to express this emotion. For example, the word "road" can be used as a metaphor to express the emotional experiences of waiting for or passing through steps in treatment. A similar phenomenon is that markers of cognition, such as "I think", can occur to introduce the abstract source of the metaphor. In example (3), one breast cancer patient in our data describes her speculation about her condition metaphorically, writing,

(3) i have such a long *road* i just wonder what to do with myself.

To encode these emotional and cognitive elements as features, we use Linguistic Inquiry Word Count (LIWC) (Tausczik and Pennebaker, 2010). LIWC is a tool that counts word use in certain

[1] We also tried standard LDA for assigning topics to sentences, by representing each sentence as a topic distribution over its words. However, this representation was not as informative as Sentence LDA in our task, so we leave out the LDA topics in further discussion.

psychologically relevant categories. Focusing on emotional and cognitive processes, we use the LIWC term lists for categories seen in Table 1.

LIWC category	Example Terms
affect	ache, like, sweet
positive emotion	passion, agree, giving
negative emotion	agony, annoy, miss
anxiety	embarrass, avoid
anger	assault, offend
sadness	despair, grim
cognitive mechanisms	if, could
insight	believe, aware
cause	make, pick
discrep	would, hope
tentativeness	anyone, suppose
certainty	never, true

Table 1: Selected LIWC categories.

We count the number of words that fall into each category within either an immediate or global context. For these LIWC features, we take the target sentence and its neighboring sentences as the immediate context and the entire post as the global context for a candidate metaphor instance. The counts for each category in either the immediate or global context are used as features encoded by what degree the immediate or global context expresses the emotional or cognitive category.

We expect words indicative of emotion and cognition to appear more frequently in metaphorical cases. Our preliminary statistical analysis on the development set revealed that this holds true within the target sentence and shows a tendency in the surrounding sentences.

3.3 Multi-Level Modeling

Our topical and emotion and cognition context features are general across target words. However, the specific features that are informative for metaphor identification may depend on the target word. To account for the specificity of target words, we use multi-level modeling (Daume III, 2007). The idea of multi-level modeling is to pair each of our features with every target word while keeping one set of features independent of the target words. There are then multiple copies of each topic transition and emotion/cognition feature, all paired with a different target word. Thus, if there are N target words, our feature space becomes $N + 1$ times larger.

4 Experiments

Our main experimental task is metaphor detection or disambiguation – given a post containing a candidate metaphor word, we aim to determine whether the word is used literally or metaphorically in context.

4.1 Data

We conducted experiments on a dataset of posts from a public breast cancer support group discussion forum, annotated by Jang et al. (2015). We chose to work on this dataset because it features metaphors occurring in naturalistic language.

In this dataset, posts are restricted to those containing one of seven candidate metaphors that appear either metaphorically or literally: "boat", "candle", "light", "ride", "road", "spice", and "train". We split the data randomly into a development set of 800 posts for preliminary analysis and a cross-validation set of 1,870 posts for classification as in (Jang et al., 2015).

4.2 Metrics

We report five evaluation metrics for every model: kappa, F1 score, precision, recall, and accuracy. Kappa, which corrects for agreement by chance, was calculated between predicted results and actual results. Because the dataset is skewed towards metaphorical instances, we rely on the first four measures over accuracy for our evaluation.

4.3 Baselines

We use the following two baselines: the feature set of (Jang et al., 2015) and a context unigram model.

Jang et al. (2015): We use the best configuration of features from Jang et al. (2015), the state-of-the-art model on our dataset, as a baseline. This feature set consists of all of their local context features (word category, semantic relatedness, concreteness), all of their global context features except lexical chaining (word category, global topic distribution), and context unigrams.

Context Unigram Model: All the words in a post, including the target word, are used as context features.

4.4 Settings

We ran Sentence LDA, setting the number of topics to 10, 20, 30, 50, and 100. α and β determine the sparsity of the topic distribution of each document and the word distribution of each topic,

Model	κ	F1	P-L	R-L	P-M	R-M	A
Unigram	.435	.714	.701	.434	.845	.943	.824
Unigram + AllTopic + AllLIWC***	.533	.765	.728	.550	.872	.937	.847
Unigram + MM AllTopic + MM AllLIWC***	.543	.770	.754	.546	.872	.946	.852
J	.575	.786	.758	.587	.882	.943	.859
J + AllTopic + AllLIWC*	.609	.804	.772	.626	.892	.943	.869
J + MM AllTopic**	.619	.809	.784	.630	.893	.947	.873
J + MM AllLIWC	.575	.787	.757	.589	.882	.942	.859
J + MM AllTopic + MM AllLIWC***	**.631**	**.815**	**.792**	**.642**	**.896**	**.948**	**.876**

Table 2: Performance on metaphor identification task. (**Models**) J: Jang et al. (2015), MM - Multilevel Modeling (**Metrics**) κ: Cohen's kappa, F1: average F1 score on M/L, P-L: precision on literals, R-L: recall on literals, P-M: precision on metaphors, R-M: recall on metaphors, A: accuracy, *: marginally statistically significant ($p < 0.1$), **: statistically significant ($p < 0.05$), ***: highly statistically significant ($p < 0.01$) improvement over corresponding baseline by Student's t-test.

respectively; the lower the sparser. Following convention, we set these parameters to 0.1 and 0.001, respectively, to enforce sparsity. We also removed the 37 most frequent words in the corpus, drawing the threshold at the point where content words and pronouns started to appear in the ranked list. The models with 10 topics performed the best on the development set, with performance degrading as the number of topics increased. We suspect that poorer performance on the models with more topics is due to feature sparsity.

We used the support vector machine (SVM) classifier provided in the `LightSIDE` toolkit (Mayfield and Rosé, 2010) with sequential minimal optimization (SMO) and a polynomial kernel of exponent 2. For each experiment, we performed 10-fold cross-validation. We also trained the baselines with the same SVM settings.

4.5 Results

The results of our classification experiment are shown in Table 2. We tested our topical and emotion and cognition features in combination with lexical features from our baselines: unigram and Jang et al. (2015).

Adding our topical and emotion/cognition features to the baselines improved performance in predicting metaphor detection. We see that our features combined with the unigram features improved over the Unigram baseline although they do not beat the Jang et al. (2015) baseline. However, when our features are combined with the features from Jang et al. (2015), we see large gains in performance. Additionally, our multi-level modeling significantly improved performance by taking into account the effects of specific metaphors. The topical features added to the baseline led to a significant improvement in accuracy, while emotion and cognition features only slightly improved the accuracy without statistical significance. However, the combination of these emotion and cognition features with topical features (in the last row of Table 2) leads to improvement. We performed a Student's t-test for calculating statistical significance.

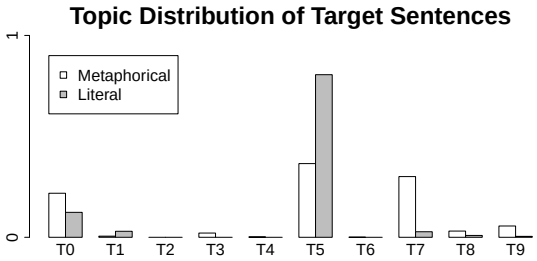


Figure 1: Proportions of topics assigned to target sentences, when target words were used metaphorically vs. literally. The proportions of metaphorical and literal cases are different with statistical significance of $p < 0.01$ by Pearson's chi-square test.

5 Discussion

Metaphorical instances tend to have personal topics. An author was more likely to use target words metaphorically when the target sentence relates more closely to their own experience of disease and treatment. Specifically, metaphors were relatively frequent when people shared their own disease experience (Topic 0, Topic 9) or sympa-

	Topic	Top Words	Example Sentences
0	Disease/ Treatment	get, chemo, if, they, as, out, can, like, now, she, feel, did, up, know, think, been, good, time, or, when	I'm scared of chemo and ct scans because it makes cancer come back and you become more resistance to treatment with drugs like these later.
1	Food	good, they, gt, can, like, eat, fat, or, if, some, one, as, them, get, up, fiber, think, more, what	*Martha's Way* Stuff a miniature marshmallow in the bottom of a sugar cone to prevent ice cream drips.
2	Emotions	love, great, laura, good, hope, like, debbie, amy, up, happy, too, everyone, day, glad, look, fun, mary, what, kelly, how	Too funny. / You're so cute! / ene23...the photo in the locket idea sounds great!
3	Time	chemo, week, go, last, then, next, weeks, taxol, good, done, treatment, first, start, one, more, rads, after, today, 'll, now	I am now 45, and just had my ONE year anniversary from finishing chemo last week!!
4	Greetings/ Thanks	thanks, hugs, hi, here, carrie, thank, welcome, love, us, glad, know, greg, good, everyone, thread, ladies, there, how, sorry, mags	Thank you so much for the story!! / Big Hugs!
5	People	she, he, they, out, get, up, her, when, like, one, as, from, there, our, time, did, if, can, go, what	She has three children and her twin sister has taken her and her 3 children in.
6	Support	good, hope, well, happy, everyone, doing, glad, luck, hear, better, take, jen, care, great, liz, birthday, hugs, lol, watson, feeling	YAY! / lol. / I wish you all good luck and peace.
7	Relation	what, know, she, as, can, her, cancer, if, there, has, think, been, how, like, our, who, when, they, would, us	She knows that she has BC but does not know that it has spread. / I just read your message and I wondered about you.
8	Religion	god, love, lord, us, prayers, our, bless, dear, her, lu, may, day, patti, thank, know, comfort, amen, xoxo, he, pray	Dear Lord, I come to you with a friend that is not doing well, Please bless her that her hands will reach for you threw the last part of her breast cancer fight.
9	Diagnosis	diagnosed, when, chemo, she, breast, years, stage, cancer, dx, now, found, nodes, no, after, lump, they, age, then, year, mastectomy	I was 64 when diagnosed wtth pure DCIS.....I had my ninght radiation treatment today. / I was diagnosed Nov 2007 at age 45.

Table 3: Topics learned by Sentence LDA.

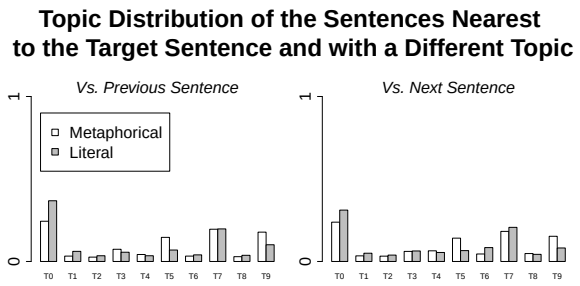


Figure 2: Proportions of the topics of the sentences that are nearest to the target sentence and have a different topic from the target sentence. The proportions of metaphorical and literal cases are different with statistical significance of $p < 0.01$ by Pearson's chi-square test.

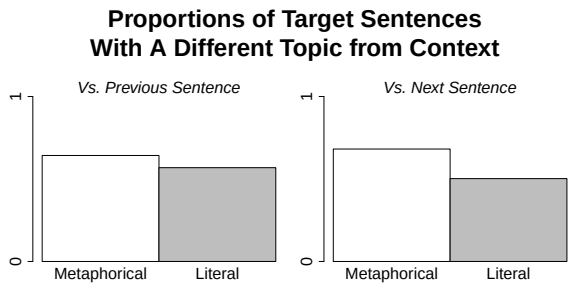


Figure 3: Proportions of target sentences whose topic is different from that of the previous/next sentence, when target words were used metaphorically vs. literally. The proportions of metaphorical and literal cases are different with statistical significance of $p < 0.01$ by Pearson's chi-square test.

thized with other people's experiences (Topic 7), but were more infrequent when they simply talked about other people in Topic 5 (Figure 1). According to our closer examination of sample sentences, Topic 0 had many personal stories about disease and treatment, and Topic 7 was about learning and relating to other people's experiences. Example metaphorical expressions include "There is *light* during chemo." (Topic 0) and "Hi Dianne - I am glad I found your post as I am sort of in the same *boat*." (Topic 7). Analysis of our LIWC features also supports the reflective nature of metaphors: "insight" and "discrepancy" words such as "wish", "seem", and "feel" occur more frequently around metaphorical uses of target terms.

The topics of the surrounding context (TopicTrans) were also informative for metaphor detection (Figure 2). However, the topics of the surrounding sentences followed an opposite pattern to the topics of the target sentence; talking

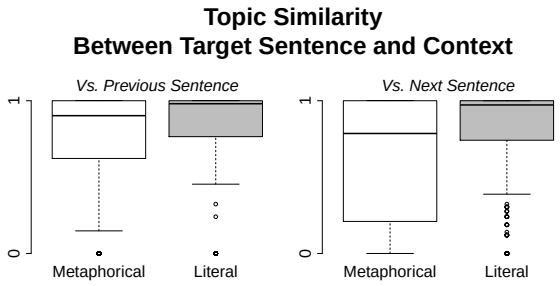


Figure 4: Cosine similarity between the topic of a target sentence and the topic of its previous/next sentence, when target words were used metaphorically vs. literally. The means of the metaphorical and literal cases are different with statistical significance of $p < 0.01$ by Welch's t-test.

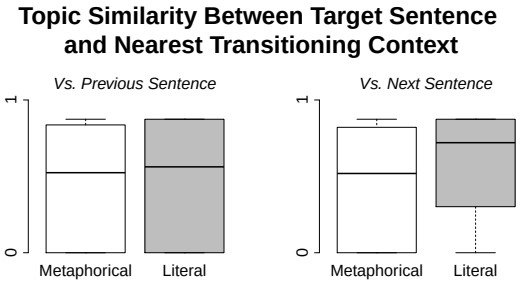


Figure 5: Cosine similarity of the topic of a target sentence and the topic of the sentences that are nearest to the target sentence and have a different topic from the target sentence. The means of metaphorical and literal cases are different with statistical significance only for the next sentence, with $p < 0.01$ by Welch's t-test.

about other people (Topic 5) in the context of a target sentence led to more metaphorical usage of target words. Similarly, writers used target words more literally before or after they shared their personal stories (Topic 0). This pattern could be because the topic of the target sentence differs from the topics of the surrounding sentences in these instances, which would mean that the target sentence is a topic that is more likely to be literal. Topic 9, however, does not follow the same pattern. One possible reason is that Topic 9 and Topic 0 tend to frequently co-occur and be metaphorical. Thus, if a target word comes after or before Topic 9 and it is Topic 0, then this word may more likely be metaphorical.

Topic transitions are effective indicators of metaphor. Metaphorical instances accompanied more drastic topic transitions than literal instances. This tendency, which matched our hypothesis, was shown in all our topic features. The immediately neighboring sentences of metaphorical instances were more likely to have a different topic from the target sentence than those of literal instances (Figure 3). Additionally, differences in topic between the target sentence and the neighboring sentences were greater for metaphorical instances (Figure 4). The nearest sentences with topics different from the target sentence (TopicTransSim) also showed this pattern (Figure 5). An interesting finding was that a topic transition after the target sentence was more indicative of metaphor than a transition before.

Emotion and cognitive words are discriminative depending on the metaphor. Emotion and cognition in the surrounding contexts, which were captured by the LIWC features, helped identify metaphors when combined with topical features. This result supports the claim in (Fainsilber and Ortony, 1987) that descriptions of feelings contain more metaphorical language than descriptions of behavior.

This effect, however, was limited to specific target words and emotions. For example, we saw a higher number of anxiety words in the immediate and global contexts of metaphors, but the trend was the opposite for anger words. This may be because our target words, "boat", "candle", "light", "ride", "road", "spice" and "train", relate more to anxiety in metaphors such as "bumpy road" and "rollercoaster ride", than to anger. On the other hand, cognitive words had more consistency, as words marking insight and discrepancy were seen significantly higher around metaphorical uses of the target words. These patterns, nevertheless, could be limited to our domain. It would be interesting to explore other patterns in different domains.

A multi-level model captures word-specific effects. Our features in context helped recognize metaphors in different ways for different target words, captured by the multi-level model. The paucity of general trends across metaphorical terms does not mean a limited applicability of our method, though, as our features do not suppose any specific trends. Rather, our method only assumes the existence of a correlation between metaphors and the theme of their context, and our multi-level model effectively identifies the interaction between metaphorical terms and their con-

texts as useful information.

For all the figures in this section, most target words have a similar pattern. See our supplemental material for graphs by target word.

6 Conclusion

We propose a new, effective method for metaphor detection using (1) sentence level topic transitions between target sentences and surrounding contexts and (2) emotion and cognition words. Both types of features showed significant improvement over the state-of-the-art. In particular, our system made significant gains in solving the problem of over-classification in metaphor detection.

We also find that personal topics are markers of metaphor, as well as certain patterns in topic transition. Additionally, language expressing emotion and cognition relates to metaphor, but in ways specific to particular candidate words. For our breast cancer forum dataset, we find more words related to anxiety around metaphors.

Our proposed features can be expanded to other domains. Though in other domains, the specific topic transition and emotion/cognition patterns would likely be different, these features would still be relevant to metaphor detection.

Acknowledgments

This research was supported in part by NSF Grant IIS-1302522.

References

George Aaron Broadwell, Umit Boz, Ignacio Cases, Tomek Strzalkowski, Laurie Feldman, Sarah Taylor, Samira Shaikh, Ting Liu, Kit Cho, and Nick Webb. 2013. Using imageability and topic chaining to locate metaphors in linguistic corpora. In *Social Computing, Behavioral-Cultural Modeling and Prediction*, pages 102–110. Springer.

Scott A Crossley and Danielle S McNamara. 2010. Cohesion, coherence, and expert evaluations of writing proficiency. In *Proceedings of the 32nd annual conference of the Cognitive Science Society*, pages 984–989.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

Lynn Fainsilber and Andrew Ortony. 1987. Metaphorical uses of language in the expression of emotions. *Metaphor and Symbolic Activity*, 2(4):239–250.

Hyeju Jang, Seunghwan Moon, Yohan Jo, and Carolyn Penstein Rosé. 2015. Metaphor detection in discourse. In *16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 384.

Yohan Jo and Alice Oh. 2011. Aspect and Sentiment Unification Model for Online Review Analysis. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 815–824.

Beata Beigman Klebanov, Chee Wee Leong, and Michael Flor. 2015. Supervised word-level metaphor detection: Experiments with concreteness and reweighting of examples. *NAACL HLT 2015 3rd Metaphor Workshop*, page 11.

George Lakoff and M. Johnson. 1980. Metaphors we live by. *Chicago/London*.

Linlin Li and Caroline Sporleder. 2010. Using gaussian mixture models to detect figurative language in context. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 297–300, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elijah Mayfield and Carolyn Rosé. 2010. An interactive tool for supporting error analysis for text mining. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 25–28. Association for Computational Linguistics.

Brian P Meier and Michael D Robinson. 2005. The metaphorical representation of affect. *Metaphor and symbol*, 20(4):239–257.

Michael Mohler, David Bracewell, David Hinote, and Marc Tomlinson. 2013. Semantic signatures for example-based linguistic metaphor detection. In *Proceedings of the First Workshop on Metaphor in NLP*, pages 27–35.

Marc Schulder and Eduard Hovy. 2014. Metaphor detection through term relevance. *ACL 2014*, page 18.

Ekaterina Shutova. 2015. Design and evaluation of metaphor processing systems. *Computational Linguistics*.

Caroline Sporleder and Linlin Li. 2009. Unsupervised recognition of literal and non-literal use of idiomatic expressions. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 754–762. Association for Computational Linguistics.

Tomek Strzalkowski, George Aaron Broadwell, Sarah Taylor, Laurie Feldman, Boris Yamrom, Samira Shaikh, Ting Liu, Kit Cho, Umit Boz, Ignacio Cases, et al. 2013. Robust extraction of metaphors from novel data. *Meta4NLP 2013*, page 67.

Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of language and social psychology*, 29(1):24–54.

Yulia Tsvetkov, Leonid Boytsov, Anatole Gershman, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on the Empirical Methods in Natural Language Processing*, pages 680–690.

Compressing Neural Language Models by Sparse Word Representations

Yunchuan Chen,[1,2] **Lili Mou,**[1,3] **Yan Xu,**[1,3] **Ge Li,**[1,3] **Zhi Jin**[1,3,*]
[1]Key Laboratory of High Confidence Software Technologies (Peking University), MoE, China
[2]University of Chinese Academy of Sciences, chenyunchuan11@mails.ucas.ac.cn
[3]Institute of Software, Peking University, doublepower.mou@gmail.com,
{xuyan14,lige,zhijin}@pku.edu.cn *Corresponding author

Abstract

Neural networks are among the state-of-the-art techniques for language modeling. Existing neural language models typically map discrete words to distributed, dense vector representations. After information processing of the preceding context words by hidden layers, an output layer estimates the probability of the next word. Such approaches are time- and memory-intensive because of the large numbers of parameters for word embeddings and the output layer. In this paper, we propose to compress neural language models by sparse word representations. In the experiments, the number of parameters in our model increases very slowly with the growth of the vocabulary size, which is almost imperceptible. Moreover, our approach not only reduces the parameter space to a large extent, but also improves the performance in terms of the perplexity measure.[1]

1 Introduction

Language models (LMs) play an important role in a variety of applications in natural language processing (NLP), including speech recognition and document recognition. In recent years, neural network-based LMs have achieved significant breakthroughs: they can model language more precisely than traditional n-gram statistics (Mikolov et al., 2011); it is even possible to generate new sentences from a neural LM, benefiting various downstream tasks like machine translation, summarization, and dialogue systems (Devlin et al., 2014; Rush et al., 2015; Sordoni et al., 2015; Mou et al., 2015b).

[1]Code released on https://github.com/chenych11/lm

Existing neural LMs typically map a discrete word to a distributed, real-valued vector representation (called *embedding*) and use a neural model to predict the probability of each word in a sentence. Such approaches necessitate a large number of parameters to represent the embeddings and the output layer's weights, which is unfavorable in many scenarios. First, with a wider application of neural networks in resource-restricted systems (Hinton et al., 2015), such approach is too memory-consuming and may fail to be deployed in mobile phones or embedded systems. Second, as each word is assigned with a dense vector—which is tuned by gradient-based methods—neural LMs are unlikely to learn meaningful representations for infrequent words. The reason is that infrequent words' gradient is only occasionally computed during training; thus their vector representations can hardly been tuned adequately.

In this paper, we propose a compressed neural language model where we can reduce the number of parameters to a large extent. To accomplish this, we first represent infrequent words' embeddings with frequent words' by sparse linear combinations. This is inspired by the observation that, in a dictionary, an unfamiliar word is typically defined by common words. We therefore propose an optimization objective to compute the sparse codes of infrequent words. The property of sparseness (only 4–8 values for each word) ensures the efficiency of our model.

Based on the pre-computed sparse codes, we design our compressed language model as follows. A dense embedding is assigned to each common word; an infrequent word, on the other hand, computes its vector representation by a sparse combination of common words' embeddings. We use the long short term memory (LSTM)-based recurrent neural network (RNN) as the hidden layer of

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 226–235,
Berlin, Germany, August 7-12, 2016.

our model. The weights of the output layer are also compressed in a same way as embeddings. Consequently, the number of trainable neural parameters is a constant regardless of the vocabulary size if we ignore the biases of words. Even considering sparse codes (which are very small), we find the memory consumption grows imperceptibly with respect to the vocabulary.

We evaluate our LM on the Wikipedia corpus containing up to 1.6 billion words. During training, we adopt noise-contrastive estimation (NCE) (Gutmann and Hyvärinen, 2012) to estimate the parameters of our neural LMs. However, different from Mnih and Teh (2012), we tailor the NCE method by adding a regression layer (called `ZRegressoion`) to predict the normalization factor, which stabilizes the training process. Experimental results show that, our compressed LM not only reduces the memory consumption, but also improves the performance in terms of the perplexity measure.

To sum up, the main contributions of this paper are three-fold. (1) We propose an approach to represent uncommon words' embeddings by a sparse linear combination of common ones'. (2) We propose a compressed neural language model based on the pre-computed sparse codes. The memory increases very slowly with the vocabulary size (4–8 values for each word). (3) We further introduce a `ZRegression` mechanism to stabilize the NCE algorithm, which is potentially applicable to other LMs in general.

2 Background

2.1 Standard Neural LMs

Language modeling aims to minimize the joint probability of a corpus (Jurafsky and Martin, 2014). Traditional n-gram models impose a Markov assumption that a word is only dependent on previous $n-1$ words and independent of its position. When estimating the parameters, researchers have proposed various smoothing techniques including back-off models to alleviate the problem of data sparsity.

Bengio et al. (2003) propose to use a feed-forward neural network (FFNN) to replace the multinomial parameter estimation in n-gram models. Recurrent neural networks (RNNs) can also be used for language modeling; they are especially capable of capturing long range dependencies in sentences (Mikolov et al., 2010; Sundermeyer et al., 2015).

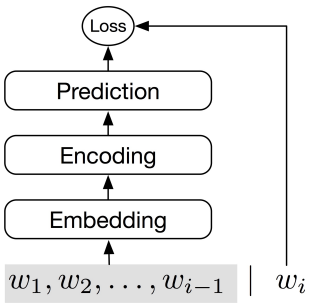


Figure 1: The architecture of a neural network-based language model.

In the above models, we can view that a neural LM is composed of three main parts, namely the `Embedding`, `Encoding`, and `Prediction` subnets, as shown in Figure 1.

The `Embedding` subnet maps a word to a dense vector, representing some abstract features of the word (Mikolov et al., 2013). Note that this subnet usually accepts a list of words (known as history or context words) and outputs a sequence of word embeddings.

The `Encoding` subnet encodes the history of a target word into a dense vector (known as *context* or *history* representation). We may either leverage FFNNs (Bengio et al., 2003) or RNNs (Mikolov et al., 2010) as the `Encoding` subnet, but RNNs typically yield a better performance (Sundermeyer et al., 2015).

The `Prediction` subnet outputs a distribution of target words as

$$p(w = w_i|h) = \frac{\exp(s(h, w_i))}{\sum_j \exp(s(h, w_j))}, \quad (1)$$

$$s(h, w_i) = \boldsymbol{W}_i^\top \boldsymbol{h} + b_i, \quad (2)$$

where $\boldsymbol{h}$ is the vector representation of context/history h, obtained by the `Encoding` subnet. $\boldsymbol{W} = (\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_V) \in \mathbb{R}^{C \times V}$ is the *output weights* of `Prediction`; $\boldsymbol{b} = (b_1, b_2, \ldots, b_V) \in \mathbb{R}^C$ is the bias (the prior). $s(h, w_i)$ is a scoring function indicating the degree to which the context h matches a target word w_i. (V is the size of vocabulary $\mathcal{V}$; C is the dimension of context/history, given by the `Encoding` subnet.)

2.2 Complexity Concerns of Neural LMs

Neural network-based LMs can capture more precise semantics of natural language than n-gram models because the regularity of the `Embedding` subnet extracts meaningful semantics of a word

and the high capacity of `Encoding` subnet enables complicated information processing.

Despite these, neural LMs also suffer from several disadvantages mainly out of complexity concerns.

Time complexity. Training neural LMs is typically time-consuming especially when the vocabulary size is large. The normalization factor in Equation (1) contributes most to time complexity. Morin and Bengio (2005) propose hierarchical softmax by using a Bayesian network so that the probability is self-normalized. Sampling techniques—for example, importance sampling (Bengio and Senécal, 2003), noise-contrastive estimation (Gutmann and Hyvärinen, 2012), and target sampling (Jean et al., 2014)—are applied to avoid computation over the entire vocabulary. Infrequent normalization maximizes the unnormalized likelihood with a penalty term that favors normalized predictions (Andreas and Klein, 2014).

Memory complexity and model complexity. The number of parameters in the `Embedding` and `Prediction` subnets in neural LMs increases linearly with respect to the vocabulary size, which is large (Table 1). As said in Section 1, this is sometimes unfavorable in memory-restricted systems. Even with sufficient hardware resources, it is problematic because we are unlikely to fully tune these parameters. Chen et al. (2015) propose the differentiated softmax model by assigning fewer parameters to rare words than to frequent words. However, their approach only handles the output weights, i.e., $\boldsymbol{W}$ in Equation (2); the input embeddings remain uncompressed in their approach.

In this work, we mainly focus on memory and model complexity, i.e., we propose a novel method to compress the `Embedding` and `Prediction` subnets in neural language models.

2.3 Related Work

Existing work on model compression for neural networks. Buciluǎ et al. (2006) and Hinton et al. (2015) use a well-trained large network to guide the training of a small network for model compression. Jaderberg et al. (2014) compress neural models by matrix factorization, Gong et al. (2014) by quantization. In NLP, Mou et al. (2015a) learn an embedding subspace by supervised training. Our work resembles little, if any, to the above methods as we compress embeddings and output weights using sparse word representations. Existing model compression typically works with a compromise of performance. On the contrary, our model improves the perplexity measure after compression.

Sub-nets	RNN-LSTM	FFNN
`Embedding`	VE	VE
`Encoding`	$4(CE + C^2 + C)$	$nCE + C$
`Prediction`	$V(C + 1)$	$V(C + 1)$
TOTAL†	$\mathcal{O}((C + E)V)$	$\mathcal{O}((E + C)V)$

Table 1: Number of parameters in different neural network-based LMs. E: embedding dimension; C: context dimension; V: vocabulary size. †Note that $V \gg C$ (or E).

Sparse word representations. We leverage sparse codes of words to compress neural LMs. Faruqui et al. (2015) propose a sparse coding method to represent each word with a sparse vector. They solve an optimization problem to obtain the sparse vectors of words as well as a dictionary matrix simultaneously. By contrast, we do not estimate any dictionary matrix when learning sparse codes, which results in a simple and easy-to-optimize model.

3 Our Proposed Model

In this section, we describe our compressed language model in detail. Subsection 3.1 formalizes the sparse representation of words, serving as the premise of our model. On such a basis, we compress the `Embedding` and `Prediction` subnets in Subsections 3.2 and 3.3, respectively. Finally, Subsection 3.4 introduces NCE for parameter estimation where we further propose the `ZRegression` mechanism to stabilize our model.

3.1 Sparse Representations of Words

We split the vocabulary $\mathcal{V}$ into two disjoint subsets ($\mathcal{B}$ and $\mathcal{C}$). The first subset $\mathcal{B}$ is a base set, containing a fixed number of common words (8k in our experiments). $\mathcal{C} = \mathcal{V} \backslash \mathcal{B}$ is a set of uncommon words. We would like to use $\mathcal{B}$'s word embeddings to encode $\mathcal{C}$'s.

Our intuition is that oftentimes a word can be defined by a few other words, and that rare words should be defined by common ones. Therefore, it is reasonable to use a few common words' embeddings to represent that of a rare word. Following most work in the literature (Lee et al., 2006; Yang et al., 2011), we represent each uncommon word with a sparse, linear combination of com-

mon ones' embeddings. The sparse coefficients are called a *sparse code* for a given word.

We first train a word representation model like SkipGram (Mikolov et al., 2013) to obtain a set of embeddings for each word in the vocabulary, including both common words and rare words. Suppose $\boldsymbol{U} = (\boldsymbol{U}_1, \boldsymbol{U}_2, \ldots, \boldsymbol{U}_B) \in \mathbb{R}^{E \times B}$ is the (learned) embedding matrix of common words, i.e., $\boldsymbol{U}_i$ is the embedding of i-th word in $\mathcal{B}$. (Here, $B = |\mathcal{B}|$.)

Each word in $\mathcal{B}$ has a natural sparse code (denoted as $\boldsymbol{x}$): it is a one-hot vector with B elements, the i-th dimension being on for the i-th word in $\mathcal{B}$.

For a word $w \in \mathcal{C}$, we shall learn a sparse vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_B)$ as the sparse code of the word. Provided that $\boldsymbol{x}$ has been learned (which will be introduced shortly), the embedding of w is

$$\hat{\boldsymbol{w}} = \sum_{j=1}^{B} x_j \boldsymbol{U}_j = \boldsymbol{U}\boldsymbol{x}, \tag{3}$$

To learn the sparse representation of a certain word w, we propose the following optimization objective

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{U}\boldsymbol{x} - \boldsymbol{w}\|_2^2 + \alpha\|\boldsymbol{x}\|_1 + \beta|\boldsymbol{1}^\top \boldsymbol{x} - 1| + \gamma \boldsymbol{1}^\top \max\{\boldsymbol{0}, -\boldsymbol{x}\}, \tag{4}$$

where $\max$ denotes the component-wise maximum; $\boldsymbol{w}$ is the embedding for a rare word $w \in \mathcal{C}$.

The first term (called *fitting loss* afterwards) evaluates the closeness between a word's coded vector representation and its "true" representation $\boldsymbol{w}$, which is the general goal of sparse coding.

The second term is an ℓ_1 regularizer, which encourages a sparse solution. The last two regularization terms favor a solution that sums to 1 and that is nonnegative, respectively. The nonnegative regularizer is applied as in He et al. (2012) due to psychological interpretation concerns.

It is difficult to determine the hyperparameters α, β, and γ. Therefore we perform several tricks. First, we drop the last term in the problem (4), but clip each element in $\boldsymbol{x}$ so that all the sparse codes are nonnegative during each update of training.

Second, we re-parametrize α and β by balancing the fitting loss and regularization terms dynamically during training. Concretely, we solve the following optimization problem, which is slightly different but closely related to the conceptual objective (4):

$$\min_{\boldsymbol{x}} \ L(\boldsymbol{x}) + \alpha_t R_1(\boldsymbol{x}) + \beta_t R_2(\boldsymbol{x}), \tag{5}$$

where $L(\boldsymbol{x}) = \|\boldsymbol{U}\boldsymbol{x} - \boldsymbol{w}\|_2^2$, $R_1(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$, and $R_2(\boldsymbol{x}) = |\boldsymbol{1}^\top \boldsymbol{x} - 1|$. α_t and β_t are adaptive parameters that are resolved during training time. Suppose $\boldsymbol{x}_t$ is the value we obtain after the update of the t-th step, we expect the importance of fitness and regularization remain unchanged during training. This is equivalent to

$$\frac{\alpha_t R_1(\boldsymbol{x}_t)}{L(\boldsymbol{x}_t)} = w_\alpha \equiv \text{const}, \tag{6}$$

$$\frac{\beta_t R_2(\boldsymbol{x}_t)}{L(\boldsymbol{x}_t)} = w_\beta \equiv \text{const}. \tag{7}$$

or

$$\alpha_t = \frac{L(\boldsymbol{x}_t)}{R_1(\boldsymbol{x}_t)} w_\alpha \text{ and } \beta_t = \frac{L(\boldsymbol{x}_t)}{R_2(\boldsymbol{x}_t)} w_\beta,$$

where w_α and w_β are the ratios between the regularization loss and the fitting loss. They are much easier to specify than α or β in the problem (4).

We have two remarks as follows.

- To learn the sparse codes, we first train the "true" embeddings by `word2vec`[2] for both common words and rare words. However, these true embeddings are slacked during our language modeling.
- As the codes are pre-computed and remain unchanged during language modeling, they are not tunable parameters of our neural model. Considering the learned sparse codes, we need only 4–8 values for each word on average, as the codes contain 0.05–0.1% nonzero values, which are almost negligible.

3.2 Parameter Compression for the `Embedding` Subnet

One main source of LM parameters is the `Embedding` subnet, which takes a list of words (history/context) as input, and outputs dense, low-dimensional vector representations of the words.

We leverage the sparse representation of words mentioned above to construct a compressed `Embedding` subnet, where the number of parameters is independent of the vocabulary size.

By solving the optimization problem (5) for each word, we obtain a non-negative sparse code $\boldsymbol{x} \in \mathbb{R}^B$ for each word, indicating the degree to which the word is related to common words in $\mathcal{B}$. Then the embedding of a word is given by $\hat{\boldsymbol{w}} = \boldsymbol{U}\boldsymbol{x}$.

[2] https://code.google.com/archive/p/word2vec

We would like to point out that the embedding of a word $\hat{\boldsymbol{w}}$ is not sparse because $\boldsymbol{U}$ is a dense matrix, which serves as a shared parameter of learning all words' vector representations.

3.3 Parameter Compression for the `Prediction` Subnet

Another main source of parameters is the `Prediction` subnet. As Table 1 shows, the output layer contains V target-word weight vectors and biases; the number increases with the vocabulary size. To compress this part of a neural LM, we propose a weight-sharing method that uses words' sparse representations again. Similar to the compression of word embeddings, we define a base set of weight vectors, and use them to represent the rest weights by sparse linear combinations.

Without loss of generality, we let $\boldsymbol{D} = \boldsymbol{W}_{:,1:B}$ be the output weights of B base target words, and $\boldsymbol{c} = \boldsymbol{b}_{1:B}$ be bias of the B target words.[3] The goal is to use $\boldsymbol{D}$ and $\boldsymbol{c}$ to represent $\boldsymbol{W}$ and $\boldsymbol{b}$. However, as the values of $\boldsymbol{W}$ and $\boldsymbol{b}$ are unknown before the training of LM, we cannot obtain their sparse codes in advance.

We claim that it is reasonable to share the same set of sparse codes to represent word vectors in `Embedding` and the output weights in the `Prediction` subnet. In a given corpus, an occurrence of a word is always companied by its context. The co-occurrence statistics about a word or corresponding context are the same. As both word embedding and context vectors capture these co-occurrence statistics (Levy and Goldberg, 2014), we can expect that context vectors share the same internal structure as embeddings. Moreover, for a fine-trained network, given any word w and its context h, the output layer's weight vector corresponding to w should specify a large inner-product score for the context h; thus these context vectors should approximate the weight vector of w. Therefore, word embeddings and the output weight vectors should share the same internal structures and it is plausible to use a same set of sparse representations for both words and target-word weight vectors. As we shall show in Section 4, our treatment of compressing the `Prediction` subnet does make sense and achieves high performance.

Formally, the i-th output weight vector is estimated by

$$\hat{\boldsymbol{W}}_i = \boldsymbol{D}\boldsymbol{x}_i, \tag{8}$$

[3] $\boldsymbol{W}_{:,1:B}$ is the first B columns of $\boldsymbol{W}$.

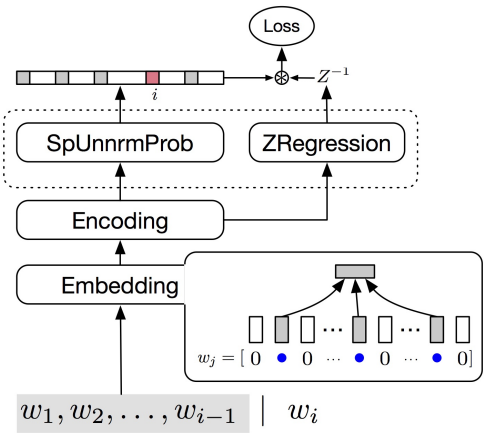


Figure 2: Compressing the output of neural LM. We apply NCE to estimate the parameters of the `Prediction` sub-network (dashed round rectangle). The SpUnnrmProb layer outputs a **sp**arse, **unnorm**alized **prob**ability of the next word. By "sparsity," we mean that, in NCE, the probability is computed for only the "true" next word (red) and a few generated negative samples.

The biases can also be compressed as

$$\hat{b}_i = \boldsymbol{c}\boldsymbol{x}_i. \tag{9}$$

where $\boldsymbol{x}_i$ is the sparse representation of the i-th word. (It is shared in the compression of weights and biases.)

In the above model, we have managed to compressed a language model whose number of parameters is irrelevant to the vocabulary size.

To better estimate a "prior" distribution of words, we may alternatively assign an independent bias to each word, i.e., $\boldsymbol{b}$ is not compressed. In this variant, the number of model parameters grows very slowly and is also negligible because each word needs only one extra parameter. Experimental results show that by not compressing the bias vector, we can even improve the performance while compressing LMs.

3.4 Noise-Contrastive Estimation with `ZRegression`

We adopt the noise-contrastive estimation (NCE) method to train our model. Compared with the maximum likelihood estimation of softmax, NCE reduces computational complexity to a large degree. We further propose the `ZRegression` mechanism to stablize training.

NCE generates a few negative samples for each positive data sample. During training, we only

need to compute the unnormalized probability of these positive and negative samples. Interested readers are referred to (Gutmann and Hyvärinen, 2012) for more information.

Formally, the estimated probability of the word w_i with history/context h is

$$P(w|h;\boldsymbol{\theta}) = \frac{1}{Z_h} P^0(w_i|h;\boldsymbol{\theta}) = \frac{1}{Z_h} \exp(s(w_i, h;\boldsymbol{\theta})), \quad (10)$$

where $\boldsymbol{\theta}$ is the parameters and Z_h is a context-dependent normalization factor. $P^0(w_i|h;\boldsymbol{\theta})$ is the unnormalized probability of the w (given by the SpUnnrmProb layer in Figure 2).

The NCE algorithm suggests to take Z_h as parameters to optimize along with $\boldsymbol{\theta}$, but it is intractable for context with variable lengths or large sizes in language modeling. Following Mnih and Teh (2012), we set $Z_h = 1$ for all h in the base model (without `ZRegression`).

The objective for each occurrence of context/history h is

$$J(\boldsymbol{\theta}|h) = \log \frac{P(w_i|h;\boldsymbol{\theta})}{P(w_i|h;\boldsymbol{\theta}) + kP_n(w_i)} + \sum_{j=1}^{k} \log \frac{kP_n(w_j)}{P(w_j|h;\boldsymbol{\theta}) + kP_n(w_j)},$$

where $P_n(w)$ is the probability of drawing a negative sample w; k is the number of negative samples that we draw for each positive sample.

The overall objective of NCE is

$$J(\boldsymbol{\theta}) = \mathbb{E}_h[J(\boldsymbol{\theta}|h)] \approx \frac{1}{M} \sum_{i=1}^{M} J(\boldsymbol{\theta}|h_i),$$

where h_i is an occurrence of the context and M is the total number of context occurrences.

Although setting Z_h to 1 generally works well in our experiment, we find that in certain scenarios, the model is unstable. Experiments show that when the true normalization factor is far away from 1, the cost function may vibrate. To comply with NCE in general, we therefore propose a `ZRegression` layer to predict the normalization constant Z_h dependent on h, instead of treating it as a constant.

The regression layer is computed by

$$Z_h^{-1} = \exp(\boldsymbol{W}_Z^\top \boldsymbol{h} + b_Z),$$

Partitions	Running words
Train (n-gram)	1.6 B
Train (neural LMs)	100 M
Dev	100 K
Test	5 M

Table 2: Statistics of our corpus.

where $\boldsymbol{W}_Z \in \mathbb{R}^C$ and $b_Z \in \mathbb{R}$ are weights and bias for `ZRegression`. Hence, the estimated probability by NCE with `ZRegression` is given by

$$P(w|h) = \exp(s(h, w)) \cdot \exp(\boldsymbol{W}_Z^\top \boldsymbol{h} + b_Z).$$

Note that the `ZRegression` layer does not guarantee normalized probabilities. During validation and testing, we explicitly normalize the probabilities by Equation (1).

4 Evaluation

In this part, we first describe our dataset in Subsection 4.1. We evaluate our learned sparse codes of rare words in Subsection 4.2 and the compressed language model in Subsection 4.3. Subsection 4.4 provides in-depth analysis of the `ZRegression` mechanism.

4.1 Dataset

We used the freely available Wikipedia[4] dump (2014) as our dataset. We extracted plain sentences from the dump and removed all markups. We further performed several steps of preprocessing such as text normalization, sentence splitting, and tokenization. Sentences were randomly shuffled, so that no information across sentences could be used, i.e., we did not consider cached language models. The resulting corpus contains about 1.6 billion running words.

The corpus was split into three parts for training, validation, and testing. As it is typically time-consuming to train neural networks, we sampled a subset of 100 million running words to train neural LMs, but the full training set was used to train the backoff n-gram models. We chose hyperparameters by the validation set and reported model performance on the test set. Table 2 presents some statistics of our dataset.

4.2 Qualitative Analysis of Sparse Codes

To obtain words' sparse codes, we chose 8k common words as the "dictionary," i.e., $B = 8000$.

[4] http://en.wikipedia.org

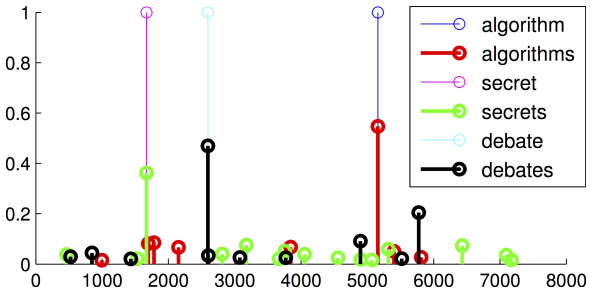


Figure 3: The sparse representations of selected words. The x-axis is the dictionary of 8k common words; the y-axis is the coefficient of sparse coding. Note that *algorithm*, *secret*, and *debate* are common words, each being coded by itself with a coefficient of 1.

We had 2k–42k uncommon words in different settings. We first pretrained word embeddings of both rare and common words, and obtained 200d vectors $\boldsymbol{U}$ and $\boldsymbol{w}$ in Equation (5). The dimension was specified in advance and not tuned. As there is no analytic solution to the objective, we optimized it by Adam (Kingma and Ba, 2014), which is a gradient-based method. To filter out small coefficients around zero, we simply set a value to 0 if it is less than $0.015 \cdot \max\{v \in \boldsymbol{x}\}$. w_α in Equation (6) was set to 1 because we deemed fitting loss and sparsity penalty are equally important. We set w_β in Equation (7) to 0.1, and this hyperparameter is insensitive.

Figure 3 plots the sparse codes of a few selected words. As we see, *algorithm*, *secret*, and *debate* are common words, and each is (sparsely) coded by itself with a coefficient of 1. We further notice that a rare word like *algorithms* has a sparse representation with only a few non-zero coefficient.

Moreover, the coefficient in the code of *algorithms*—corresponding to the base word *algorithm*—is large (~ 0.6), showing that the words *algorithm* and *algorithms* are similar. Such phenomena are also observed with *secret* and *debate*.

The qualitative analysis demonstrates that our approach can indeed learn a sparse code of a word, and that the codes are meaningful.

4.3 Quantitative Analysis of Compressed Language Models

We then used the pre-computed sparse codes to compress neural LMs, which provides quantitative analysis of the learned sparse representations of words. We take perplexity as the performance measurement of a language model, which is defined by

$$\text{PPL} = 2^{-\frac{1}{N}\sum_{i=1}^{N} \log_2 p(w_i|h_i)}$$

where N is the number of running words in the test corpus.

4.3.1 Settings

We leveraged LSTM-RNN as the `Encoding` subnet, which is a prevailing class of neural networks for language modeling (Sundermeyer et al., 2015; Karpathy et al., 2015). The hidden layer was 200d. We used the Adam algorithm to train our neural models. The learning rate was chosen by validation from $\{0.001, 0.002, 0.004, 0.006, 0.008\}$. Parameters were updated with a mini-batch size of 256 words. We trained neural LMs by NCE, where we generated 50 negative samples for each positive data sample in the corpus. All our model variants and baselines were trained with the same pre-defined hyperparameters or tuned over a same candidate set; thus our comparison is fair.

We list our compressed LMs and competing methods as follows.

- **KN3**. We adopted the modified Kneser-Ney smoothing technique to train a 3-gram LM; we used the SRILM toolkit (Stolcke and others, 2002) in out experiment.
- **LBL5**. A **L**og-**B**i**L**inear model introduced in Mnih and Hinton (2007). We used 5 preceding words as context.
- **LSTM-s**. A standard LSTM-RNN language model which is applied in Sundermeyer et al. (2015) and Karpathy et al. (2015). We implemented the LM ourselves based on Theano (Theano Development Team, 2016) and also used NCE for training.
- **LSTM-z**. An LSTM-RNN enhanced with the `ZRegression` mechanism described in Section 3.4.
- **LSTM-z,wb**. Based on LSTM-z, we compressed word embeddings in `Embedding` and the output weights and biases in `Prediction`.
- **LSTM-z,w**. In this variant, we did not compress the bias term in the output layer. For each word in $\mathcal{C}$, we assigned an independent bias parameter.

4.3.2 Performance

Tables 3 shows the perplexity of our compressed model and baselines. As we see, LSTM-based LMs significantly outperform the log-bilinear

Vocabulary	10k	22k	36k	50k
KN3†	90. 4	125.3	146.4	159.9
LBL5	116. 6	167.0	199.5	220.3
LSTM-s	107. 3	159.5	189.4	222.1
LSTM-z	75. 1	104.4	119.6	130.6
LSTM-z,wb	73. 7	103.4	122.9	138.2
LSTM-z,w	72. 9	101.9	119.3	129.2

Table 3: Perplexity of our compressed language models and baselines. †Trained with the full corpus of 1.6 billion running words.

Vocabulary	10k	22k	36k	50k
LSTM-z,w	17.76	59.28	73.42	79.75
LSTM-z,wb	17.80	59.44	73.61	79.95

Table 4: Memory reduction (%) by our proposed methods in comparison with the uncompressed model LSTM-z. The memory of sparse codes are included.

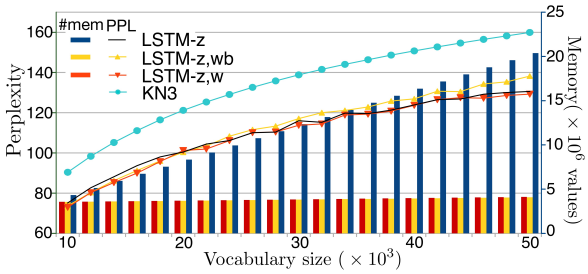


Figure 4: Fine-grained plot of performance (perplexity) and memory consumption (including sparse codes) versus the vocabulary size.

model as well as the backoff 3-gram LM, even if the 3-gram LM is trained on a much larger corpus with 1.6 billion words. The `ZRegression` mechanism improves the performance of LSTM to a large extent, which is unexpected. Subsection 4.4 will provide more in-depth analysis.

Regarding the compression method proposed in this paper, we notice that LSTM-z,wb and LSTM-z,w yield similar performance to LSTM-z. In particular, LSTM-z,w outperforms LSTM-z in all scenarios of different vocabulary sizes. Moreover, both LSTM-z,wb and LSTM-z,w can reduce the memory consumption by up to 80% (Table 4).

We further plot in Figure 4 the model performance (lines) and memory consumption (bars) in a fine-grained granularity of vocabulary sizes. We see such a tendency that compressed LMs (LSTM-z,wb and LSTM-z,w, yellow and red lines) are generally better than LSTM-z (black line) when we have a small vocabulary. However, LSTM-z,wb is slightly worse than LSTM-z if the vocabulary size is greater than, say, 20k. The LSTM-z,w remains comparable to LSTM-z as the vocabulary grows.

To explain this phenomenon, we may imagine that the compression using sparse codes has two effects: it loses information, but it also enables more accurate estimation of parameters especially for rare words. When the second factor dominates, we can reasonably expect a high performance of the compressed LM.

From the bars in Figure 4, we observe that traditional LMs have a parameter space growing linearly with the vocabulary size. But the number of parameters in our compressed models does not increase—or strictly speaking, increases at an extremely small rate—with vocabulary.

These experiments show that our method can largely reduce the parameter space with even performance improvement. The results also verify that the sparse codes induced by our model indeed capture meaningful semantics and are potentially useful for other downstream tasks.

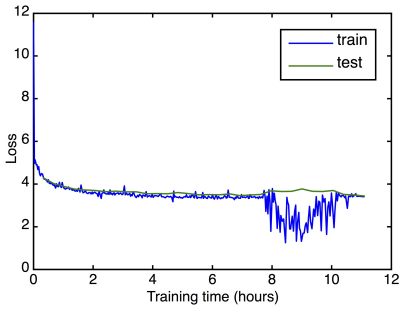
4.4 Effect of `ZRegression`

We next analyze the effect of `ZRegression` for NCE training. As shown in Figure 5a, the training process becomes unstable after processing 70% of the dataset: the training loss vibrates significantly, whereas the test loss increases.

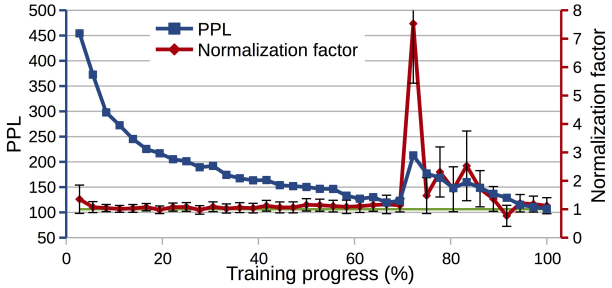
We find a strong correlation between unstableness and the Z_h factor in Equation (10), i.e., the sum of unnormalized probability (Figure 5b). Theoretical analysis shows that the Z_h factor tends to be self-normalized even though it is not forced to (Gutmann and Hyvärinen, 2012). However, problems would occur, should it fail.

In traditional methods, NCE jointly estimates normalization factor Z and model parameters (Gutmann and Hyvärinen, 2012). For language modeling, Z_h dependents on context h. Mnih and Teh (2012) propose to estimate a separate Z_h based on two history words (analogous to 3-gram), but their approach hardly scales to RNNs because of the exponential number of different combinations of history words.

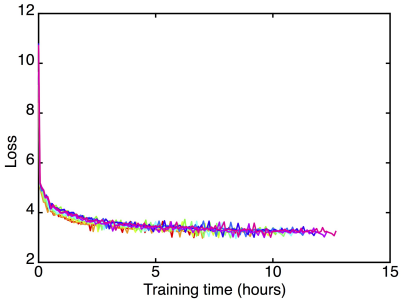
We propose the `ZRegression` mechanism in Section 3.4, which can estimate the Z_h factor well (Figure 5d) based on the history vector $\boldsymbol{h}$. In this way, we manage to stabilize the training process (Figure 5c) and improve the performance by



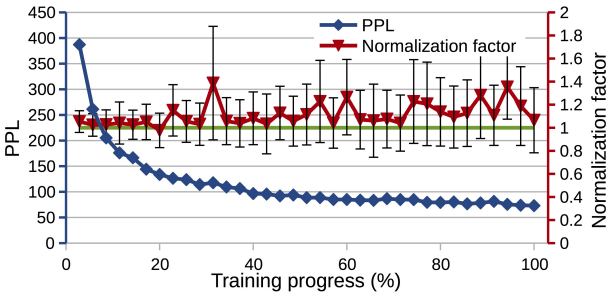
(a) Training/test loss vs. training time w/o `ZRegression`.



(b) The validation perplexity and normalization factor Z_h w/o `ZRegression`.



(c) Training loss vs. training time w/ `ZRegression` of different runs.



(d) The validation perplexity and normalization factor Z_h w/ `ZRegression`.

Figure 5: Analysis of `ZRegression`.

a large margin, as has shown in Table 3.

It should be mentioned that `ZRegression` is not specific to model compression and is generally applicable to other neural LMs trained by NCE.

5 Conclusion

In this paper, we proposed an approach to represent rare words by sparse linear combinations of common ones. Based on such combinations, we managed to compress an LSTM language model (LM), where memory does not increase with the vocabulary size except a bias and a sparse code for each word. Our experimental results also show that the compressed LM has yielded a better performance than the uncompressed base LM.

Acknowledgments

This research is supported by the National Basic Research Program of China (the 973 Program) under Grant No. 2015CB352201, the National Natural Science Foundation of China under Grant Nos. 61232015, 91318301, 61421091 and 61502014, and the China Post-Doctoral Foundation under Grant No. 2015M580927.

References

Jacob Andreas and Dan Klein. 2014. When and why are log-linear models self-normalizing. In *Proceedings of the Annual Meeting of the North American Chapter of the Association for Computational Linguistics*, pages 244–249.

Yoshua Bengio and Jean-Sébastien Senécal. 2003. Quick training of probabilistic neural nets by importance sampling. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics*.

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

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–541.

Welin Chen, David Grangier, and Michael Auli. 2015. Strategies for training large vocabulary neural language models. *arXiv preprint arXiv:1512.04906*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52rd Annual Meeting of the Association for Computational Linguistics*, pages 1370–1380.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1491–1500.

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*.

Michael Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *The Journal of Machine Learning Research*, 13(1):307–361.

Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document summarization based on data reconstruction. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, pages 620–626.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. 2014. Speeding up convolutional neural networks with low rank expansions. In *Proceedings of the British Machine Vision Conference*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*.

Dan Jurafsky and James H. Martin. 2014. *Speech and Language Processing*. Pearson.

Andrej Karpathy, Justin Johnson, and Fei-Fei Li. 2015. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. 2006. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems*, pages 801–808.

Omer Levy and Yoav Goldberg. 2014. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the Eighteenth Conference on Natural Language Learning*, pages 171–180.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, pages 1045–1048.

Tomas Mikolov, Anoop Deoras, Daniel Povey, Lukas Burget, and Jan Cernocký. 2011. Strategies for training large scale neural network language models. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 196–201.

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

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

Andriy Mnih and Yee-Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*.

Fréderic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 246–252.

Lili Mou, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2015a. Distilling word embeddings: An encoding approach. *arXiv preprint arXiv:1506.04488*.

Lili Mou, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. 2015b. Backward and forward language modeling for constrained natural language generation. *arXiv preprint arXiv:1512.06612*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205.

Andreas Stolcke et al. 2002. SRILM—An extensible language modeling toolkit. In *INTERSPEECH*, pages 901–904.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent LSTM neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, 23(3):517–529.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

Meng Yang, Lei Zhang, Jian Yang, and David Zhang. 2011. Robust sparse coding for face recognition. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, pages 625–632.

Intrinsic Subspace Evaluation of Word Embedding Representations

Yadollah Yaghoobzadeh and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
yadollah@cis.lmu.de

Abstract

We introduce a new methodology for intrinsic evaluation of word representations. Specifically, we identify four fundamental criteria based on the characteristics of natural language that pose difficulties to NLP systems; and develop tests that directly show whether or not representations contain *the subspaces necessary to satisfy these criteria*. Current intrinsic evaluations are mostly based on the overall similarity or *full-space similarity* of words and thus view vector representations as *points*. We show the limits of these point-based intrinsic evaluations. We apply our evaluation methodology to the comparison of a count vector model and several neural network models and demonstrate important properties of these models.

1 Introduction

Distributional word representations or *embeddings* are currently an active area of research in natural language processing (NLP). The motivation for embeddings is that knowledge about words is helpful in NLP. Representing words as vocabulary indexes may be a good approach if large training sets allow us to learn everything we need to know about a word to solve a particular task; but in most cases it helps to have a representation that contains distributional information and allows inferences like: "above" and "below" have similar syntactic behavior or "engine" and "motor" have similar meaning.

Several methods have been introduced to assess the quality of word embeddings. We distinguish two different types of evaluation in this paper: (i) *extrinsic evaluation* evaluates embeddings in an NLP application or task and (ii) *intrinsic evaluation* tests the quality of representations independent of a specific NLP task.

Each single word is a combination of a large number of morphological, lexical, syntactic, semantic, discourse and other features. Its embedding should accurately and consistently represent these features, and ideally a good evaluation method must clarify this and give a way to analyze the results. The goal of this paper is to build such an evaluation.

Extrinsic evaluation is a valid methodology, but it does not allow us to understand the properties of representations without further analysis; e.g., if an evaluation shows that embedding A works better than embedding B on a task, then that is not an analysis of the causes of the improvement. Therefore, extrinsic evaluations do not satisfy our goals.

Intrinsic evaluation analyzes the generic quality of embeddings. Currently, this evaluation mostly is done by testing overall distance/similarity of words in the embedding space, i.e., it is based on viewing word representations as points and then computing *full-space similarity*. The assumption is that the high dimensional space is smooth and similar words are close to each other. Several datasets have been developed for this purpose, mostly the result of human judgement; see (Baroni et al., 2014) for an overview. We refer to these evaluations as *point-based* and as *full-space* because they consider embeddings as points in the space – sub-similarities in subspaces are generally ignored.

Point-based intrinsic evaluation computes a score based on the full-space similarity of two words: a single number that generally does not say anything about the underlying reasons for a lower or higher value of full-space similarity. This makes it hard to interpret the results of point-based evaluation and may be the reason that contradictory results have been published; e.g., based on

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 236–246,
Berlin, Germany, August 7-12, 2016.

point-based evaluation, some papers have claimed that count-based representations perform as well as learning-based representations (Levy and Goldberg, 2014a). Others have claimed the opposite (e.g., Mikolov et al. (2013), Pennington et al. (2014), Baroni et al. (2014)).

Given the limits of current evaluations, we propose a new methodology for intrinsic evaluation of embeddings by identifying generic fundamental criteria for embedding models that are important for representing features of words accurately and consistently. We develop corpus-based tests using supervised classification that directly show whether the representations contain the information necessary to meet the criteria or not. The fine-grained corpus-based supervision makes the sub-similarities of words important by looking at the subspaces of word embeddings relevant to the criteria, and this enables us to give direct insights into properties of representation models.

2 Related Work

Baroni et al. (2014) evaluate embeddings on different intrinsic tests: similarity, analogy, synonym detection, categorization and selectional preference. Schnabel et al. (2015) introduce tasks with more fine-grained datasets. These tasks are unsupervised and generally based on cosine similarity; this means that only the overall direction of vectors is considered or, equivalently, that *words are modeled as points* in a space and only their full-space distance/closeness is considered. In contrast, we test embeddings in a classification setting and *different subspaces of embeddings are analyzed*. Tsvetkov et al. (2015) evaluate embeddings based on their correlations with WordNet-based linguistic embeddings. However, correlation does not directly evaluate how accurately and completely an application can extract a particular piece of information from an embedding.

Extrinsic evaluations are also common (cf. (Li and Jurafsky, 2015; Köhn, 2015; Lai et al., 2015)). Li and Jurafsky (2015) conclude that embedding evaluation must go beyond human-judgement tasks like similarity and analogy. They suggest to evaluate on NLP tasks. Köhn (2015) gives similar suggestions and also recommends the use of supervised methods for evaluation. Lai et al. (2015) evaluate embeddings in different tasks with different setups and show the contradictory results of embedding models on different tasks. Idiosyncrasies of different downstream tasks can affect extrinsic evaluations and result in contradictions.

3 Criteria for word representations

Each word is a combination of different properties. Depending on the language, these properties include lexical, syntactic, semantic, world knowledge and other features. We call these properties *facets*. The ultimate goal is to learn representations for words that accurately and consistently contain these facets. Take the facet gender (GEN) as an example. We call a representation 100% *accurate* for GEN if information it contains about GEN is always accurate; we call the representation 100% *consistent* for GEN if the representation of every word that has a GEN facet contains this information.

We now introduce four important criteria that a representation must satisfy to represent facets accurately and consistently. These criteria are applied across different problems that NLP applications face in the effective use of embeddings.

Nonconflation. A word embedding must keep the evidence from different local contexts separate – "do not conflate" – because each context can infer specific facets of the word. Embeddings for different word forms with the same stem, like plural and singular forms or different verb tenses, are examples vulnerable to conflation because they occur in similar contexts.

Robustness against sparseness. One aspect of natural language that poses great difficulty for statistical modeling is sparseness. Rare words are common in natural language and embedding models must learn useful representations based on a small number of contexts.

Robustness against ambiguity. Another central problem when processing words in NLP is lexical ambiguity (Cruse, 1986; Zhong and Ng, 2010). Polysemy and homonymy of words can make it difficult for a statistical approach to generalize and infer well. Embeddings should fully represent all senses of an ambiguous word. This criterion becomes more difficult to satisfy as distributions of senses become more skewed, but a robust model must be able to overcome this.

Accurate and consistent representation of multifacetedness. This criterion addresses settings with large numbers of facets. It is based on the following linguistic phenomenon, a phenomenon that occurs frequently crosslinguistically

(Comrie, 1989). (i) Words have a large number of facets, including phonetic, morphological, syntactic, semantic and topical properties. (ii) Each facet by itself constitutes a small part of the overall information that a representation should capture about a word.

4 Experimental setup and results

We now design experiments to directly evaluate embeddings on the four criteria. We proceed as follows. First, we design a probabilistic context free grammar (PCFG) that generates a corpus that is a manifestation of the underlying phenomenon. Then we train our embedding models on the corpus. The embeddings obtained are then evaluated in a classification setting, in which we apply a linear SVM (Fan et al., 2008) to classify embeddings. Finally, we compare the classification results for different embedding models and analyze and summarize them.

Selecting embedding models. Since this paper is about developing a new evaluation methodology, the choice of models is not important as long as the models can serve to show that the proposed methodology reveals interesting differences with respect to the criteria.

On the highest level, we can distinguish two types of distributional representations. *Count vectors* (Sahlgren, 2006; Baroni and Lenci, 2010; Turney and Pantel, 2010) live in a high-dimensional vector space in which each dimension roughly corresponds to a (weighted) count of cooccurrence in a large corpus. *Learned vectors* are learned from large corpora using machine learning methods: unsupervised methods such as LSI (e.g., Deerwester et al. (1990), Levy and Goldberg (2014b)) and supervised methods such as neural networks (e.g., Mikolov et al. (2013)) and regression (e.g., Pennington et al. (2014)). Because of the recent popularity of learning-based methods, we consider one count-based and five learning-based distributional representation models.

The learning-based models are: (i) vLBL (henceforth: LBL) (vectorized log-bilinear language model) (Mnih and Kavukcuoglu, 2013), (ii) SkipGram (henceforth: SKIP) (skipgram bag-of-word model), (iii) CBOW (continuous bag-of-word model (Mikolov et al., 2013), (iv) Structured SkipGram (henceforth SSKIP), (Ling et al., 2015) and CWindow (henceforth CWIN) (continuous window model) (Ling et al., 2015). These models learn word embeddings for input and target spaces using neural network models.

1	$P(aVb\|S)$	=	1/4	
2	$P(bVa\|S)$	=	1/4	
3	$P(aWa\|S)$	=	1/8	
4	$P(aWb\|S)$	=	1/8	
5	$P(bWa\|S)$	=	1/8	
6	$P(bWb\|S)$	=	1/8	
7	$P(v_i\|V)$	=	1/5	$0 \leq i \leq 4$
8	$P(w_i\|W)$	=	1/5	$0 \leq i \leq 4$

Figure 1: Global conflation grammar. Words v_i occur in a subset of the contexts of words w_i, but the global count vector signatures are the same.

For a given context, represented by the input space representations of the left and right neighbors $\vec{v}_{i-1}$ and $\vec{v}_{i+1}$, LBL, CBOW and CWIN predict the target space $\vec{v}_i$ by combining the contexts. LBL combines $\vec{v}_{i-1}$ and $\vec{v}_{i+1}$ linearly with position dependent weights and CBOW (resp. CWIN) combines them by adding (resp. concatenation). SKIP and SSKIP predict the context words v_{i-1} or v_{i+1} given the input space $\vec{v}_i$. For SSKIP, context words are in different spaces depending on their position to the input word. In summary, CBOW and SKIP are learning embeddings using bag-of-word (BoW) models, but the other three, CWIN, SSKIP and LBL, are using position dependent models. We use `word2vec`[1] for SKIP and CBOW, `wang2vec`[2] for SSKIP and CWIN, and Lai et al. (2015)'s implementation[3] for LBL.

The count-based model is position-sensitive PPMI, Levy and Goldberg (2014a)'s explicit vector space representation model.[4] For a vocabulary of size V, the representation $\vec{w}$ of w is a vector of size $4V$, consisting of four parts corresponding to the relative positions $r \in \{-2, -1, 1, 2\}$ with respect to occurrences of w in the corpus. The entry for dimension word v in the part of $\vec{w}$ corresponding to relative position r is the PPMI (positive pointwise mutual information) weight of w and v for that relative position. The four parts of the vector are length normalized. In this paper, we use only two relative positions: $r \in \{-1, 1\}$, so each $\vec{w}$ has two parts, corresponding to immediate left and right neighbors.

[1] `code.google.com/archive/p/word2vec`
[2] `github.com/wlin12/wang2vec`
[3] `github.com/licstar/compare`
[4] `bitbucket.org/omerlevy/hyperwords`

4.1 Nonconflation

Grammar. The PCFG grammar shown in Figure 1 generates v_i words that occur in two types of contexts: a-b (line 1) and b-a (line 2); and w_i words that also occur in these two contexts (lines 4 and 5), but in addition occur in a-a (line 3) and b-b (line 6) contexts. As a result, the set of contexts in which v_i and w_i occur is different, but if we simply count the number of occurrences in the contexts, then v_i and w_i cannot be distinguished.

Dataset. We generated a corpus of 100,000 sentences. Words that can occur in a-a and b-b contexts constitute the positive class, all other words the negative class. The words v_3, v_4, w_3, w_4 were assigned to the test set, all other words to the training set.

Results. We learn representations of words by our six models and train one SVM per model; it takes a word representation as input and outputs +1 (word can occur in a-a/b-b) or -1 (it cannot). The SVMs trained on PPMI and CBOW representations assigned all four test set words to the negative class; in particular, w_3 and w_4 were incorrectly classified. Thus, the accuracy of classification for these models (50%) was not better than random. The SVMs trained on LBL, SSKIP, SSKIP and CWIN representations assigned all four test set words to the correct class: v_3 and v_4 were assigned to the negative class and w_3 and w_4 were assigned to the positive class.

Discussion. The property of embedding models that is relevant here is that PPMI is an *aggregation model*, which means it calculates aggregate statistics for each word and then computes the final word embedding from these aggregate statistics. In contrast, all our learning-based models are *iterative models*: they iterate over the corpus and each local context of a word is used as a training instance for learning its embedding.

For iterative models, it is common to use composition of words in the context, as in LBL, CBOW and CWIN. Non-compositional iterative models like SKIP and SSKIP are also popular. Aggregation models can also use composite features from context words, but these features are too sparse to be useful. The reason that the model of Agirre et al. (2009) is rarely used is precisely its inability to deal with sparseness. All widely used distributional models employ individual word occurrences as basic features.

The bad PPMI results are explained by the fact

1	$P(AVB\|S)$	=	1/2	
2	$P(CWD\|S)$	=	1/2	
3	$P(a_i\|A)$	=	1/10	$0 \leq i \leq 9$
4	$P(b_i\|B)$	=	1/10	$0 \leq i \leq 9$
5	$P(c_i\|C)$	=	1/10	$0 \leq i \leq 9$
6	$P(d_i\|D)$	=	1/10	$0 \leq i \leq 9$
7	$P(v_i\|V)$	=	1/10	$0 \leq i \leq 9$
8	$P(w_i\|W)$	=	1/10	$0 \leq i \leq 9$
9	$L' =$	$L(S)$		
10		$\cup \{a_i u_i b_i \| 0 \leq i \leq 9\}$		
11		$\cup \{c_i x_i d_i \| 0 \leq i \leq 9\}$		

Figure 2: In language L', frequent v_i and rare u_i occur in a-b contexts; frequent w_i and rare x_i occur in c-d contexts. Word representations should encode possible contexts (a-b vs. c-d) for both frequent and rare words.

that it is an aggregation model: the PPMI model cannot distinguish two words with the same global statistics – as is the case for, say, v_3 and w_3. The bad result of CBOW is probably connected to its weak (addition) composition of context, although it is an iterative compositional model. Simple representation of context words with iterative updating (through backpropagation in each training instance), can influence the embeddings in a way that SKIP and SSKIP get good results, although they are non-compositional.

As an example of conflation occurring in the English Wikipedia, consider this simple example. We replace all single digits by "7" in tokenization. We learn PPMI embeddings for the tokens and see that among the one hundred nearest neighbors of "7" are the days of the week, e.g., "Friday". As an example of a conflated feature consider the word "falls" occurring immediately to the right of the target word. The weekdays as well as single digits often have the immediate right neighbor "falls" in contexts like "Friday falls on a public holiday" and "2 out of 3 falls match" – tokenized as "7 out of 7 falls match" – in World Wrestling Entertainment (WWE). The left contexts of "Friday" and "7" are different in these contexts, but the PPMI model does not record this information in a way that would make the link to "falls" clear.

4.2 Robustness against sparseness

Grammar. The grammar shown in Figure 2 generates frequent v_i and rare u_i in a-b contexts (lines 1 and 9); and frequent w_i and rare x_i in c-d contexts (lines 2 and 10). The language generated by the PCFG on lines 1–8 is merged on lines 9–11 with the ten contexts $a_0u_0b_0 \ldots a_9u_9b_9$ (line 9)

and the ten contexts $c_0x_0d_0 \ldots c_9x_9d_9$ (line 10); that is, each of the u_i and x_i occurs exactly once in the merged language L', thus modeling the phenomenon of sparseness.

Dataset. We generated a corpus of 100,000 sentences using the PCFG (lines 1–8) and added the 20 rare sentences (lines 9–11). We label all words that can occur in c-d contexts as positive and all other words as negative. The singleton words u_i and x_i were assigned to the test set, all other words to the training set.

Results. After learning embeddings with different models, the SVM trained on PPMI representations assigned all twenty test words to the negative class. This is the correct decision for the ten u_i (since they cannot occur in a c-d context), but the incorrect decision for the x_i (since they can occur in a c-d context). Thus, the accuracy of classification was 50% and not better than random. The SVMs trained on learning-based representations classified all twenty test words correctly.

Discussion. Representations of rare words in the PPMI model are sparse. The PPMI representations of the u_i and x_i only contain two nonzero entries, one entry for an a_i or c_i (left context) and one entry for a b_i or d_i (right context). Given this sparseness, it is not surprising that representations are not a good basis for generalization and PPMI accuracy is random.

In contrast, learning-based models learn that the a_i, b_i, c_i and d_i form four different distributional classes. The final embeddings of the a_i after learning is completed are all close to each other and the same is true for the other three classes. Once the similarity of two words in the same distributional class (say, the similarity of a_5 and a_7) has been learned, the contexts for the u_i (resp. x_i) look essentially the same to embedding models as the contexts of the v_i (resp. w_i). Thus, the embeddings learned for the u_i will be similar to those learned for the v_i. This explains why learning-based representations achieve perfect classification accuracy.

This sparseness experiment highlights an important difference between count vectors and learned vectors. Count vector models are less robust in the face of sparseness and noise because they base their representations on individual contexts; the overall corpus distribution is only weakly taken into account, by way of PPMI weighting. In contrast, learned vector models make much better use of the overall corpus distribution and they can leverage second-order effects for learning improved representations. In our example, the second order effect is that the model first learns representations for the a_i, b_i, c_i and d_i and then uses these as a basis for inferring the similarity of u_i to v_i and of x_i to w_i.

1	$P(AV_1B\|S)$	=10/20	
2	$P(CW_1D\|S)$	=9/20	
3	$P(CW_2D\|S)$	$=\beta\cdot$1/20	
4	$P(AW_2B\|S)$	$=(1-\beta)\cdot$1/20	
5	$P(a_i\|A)$	=1/10	$0 \leq i \leq 9$
6	$P(b_i\|B)$	=1/10	$0 \leq i \leq 9$
7	$P(c_i\|C)$	=1/10	$0 \leq i \leq 9$
8	$P(d_i\|D)$	=1/10	$0 \leq i \leq 9$
9	$P(v_i\|V_1)$	=1/50	$0 \leq i \leq 49$
10	$P(w_i\|W_1)$	=1/45	$5 \leq i \leq 49$
11	$P(w_i\|W_2)$	=1/5	$0 \leq i \leq 4$

Figure 3: Ambiguity grammar. v_i and $w_5 \ldots w_{49}$ occur in a-b and c-d contexts only, respectively. $w_0 \ldots w_4$ are ambiguous and occur in both contexts.

4.3 Robustness against ambiguity

Grammar. The grammar in Figure 3 generates two types of contexts that we interpret as two different meanings: a-b contexts (lines 1,4) and c-d contexts (lines 2, 3). v_i occur only in a-b contexts (line 1), $w_5 \ldots w_{49}$ occur only in c-d contexts (line 2); thus, they are unambiguous. $w_0 \ldots w_4$ are ambiguous and occur with probability β in c-d contexts (line 3) and with probability $(1-\beta)$ in a-b contexts (lines 3, 4). The parameter β controls the skewedness of the sense distribution; e.g., the two senses are equiprobable for $\beta = 0.5$ and the second sense (line 4) is three times as probable as the first sense (line 3) for $\beta = 0.25$.

Dataset. The grammar specified in Figure 3 was used to generate a training corpus of 100,000 sentences. Label criterion: A word is labeled positive if it can occur in a c-d context, as negative otherwise. The test set consists of the five ambiguous words $w_0 \ldots w_4$. All other words are assigned to the training set.

Linear SVMs were trained for the binary classification task on the train set. 50 trials of this experiment were run for each of eleven values of β: $\beta = 2^{-\alpha}$ where $\alpha \in \{1.0, 1.1, 1.2, \ldots, 2.0\}$. Thus, for the smallest value of α, $\alpha = 1.0$, the two senses have the same frequency; for the largest value of α, $\alpha = 2.0$, the dominant sense is three times as frequent as the less frequent sense.

Results. Figure 4 shows accuracy of the classi-

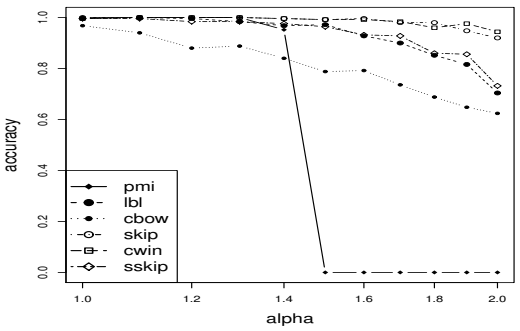


Figure 4: SVM classification results for the ambiguity dataset. X-axis: $\alpha = -\log_2 \beta$. Y-axis: classification accuracy:

fication on the test set: the proportion of correctly classified words out of a total of 250 (five words each in 50 trials).

All models perform well for balanced sense frequencies; e.g., for $\alpha = 1.0, \beta = 0.5$, the SVMs were all close to 100% accurate in predicting that the w_i can occur in a c-d context. PPMI accuracy falls steeply when α is increased from 1.4 to 1.5. It has a 100% error rate for $\alpha \geq 1.5$. Learning-based models perform better in the order CBOW (least robust), LBL, SSKIP, SKIP, CWIN (most robust). Even for $\alpha = 2.0$, CWIN and SKIP are still close to 100% accurate.

Discussion. The evaluation criterion we have used here is a classification task. The classifier attempts to answer a question that may occur in an application – can this word be used in this context? Thus, the evaluation criterion is: does the word representation contain a specific type of information that is needed for the application.

Another approach to ambiguity is to compute multiple representations for a word, one for each sense. We generally do not yet know what the sense of a word is when we want to use its word representation, so data-driven approaches like clustering have been used to create representations for different usage clusters of words that may capture some of its senses. For example, Reisinger and Mooney (2010) and Huang et al. (2012) cluster the contexts of each word and then learn a different representation for each cluster. The main motivation for this approach is the assumption that single-word distributional representations cannot represent all senses of a word well (Huang et al., 2012). However, Li and Jurafsky (2015) show that simply increasing the dimensionality of single-representation gets comparable results to using multiple-representation. Our results confirm that a single embedding can be robust against ambiguity, but also show the main challenge: skewness of sense distribution.

1	$P(NF_n\|S)$	=1/4	
2	$P(AF_a\|S)$	=1/4	
3	$P(NM_n\|S)$	=1/4	
4	$P(AM_f\|S)$	=1/4	
5	$P(n_i\|N)$	=1/5	$0 \leq i \leq 4$
6	$P(a_i\|A)$	=1/5	$0 \leq i \leq 4$
7	$P(x_i^{\text{nf}} U_i^{\text{nf}}\|F_n)$	=1/5	$0 \leq i \leq 4$
8	$P(f\|U_i^{\text{nf}})$	=1/2	
9	$P(\mu(U_i^{\text{nf}})\|U_i^{\text{nf}})$	=1/2	
10	$P(x_i^{\text{af}} U_i^{\text{af}}\|F_a)$	=1/5	$0 \leq i \leq 4$
11	$P(f\|U_i^{\text{af}})$	=1/2	
12	$P(\mu(U_i^{\text{af}})\|U_i^{\text{af}})$	=1/2	
13	$P(x_i^{\text{nm}} U_i^{\text{nm}}\|M_n)$	=1/5	$0 \leq i \leq 4$
14	$P(m\|U_i^{\text{nm}})$	=1/2	
15	$P(\mu(U_i^{\text{nm}})\|U_i^{\text{nm}})$	=1/2	
16	$P(x_i^{\text{am}} U_i^{\text{am}}\|M_f)$	=1/5	$0 \leq i \leq 4$
17	$P(m\|U_i^{\text{am}})$	=1/2	
18	$P(\mu(U_i^{\text{am}})\|U_i^{\text{am}})$	=1/2	

Figure 5: This grammar generates nouns ($x_i^{\text{n}\cdot}$) and adjectives ($x_i^{\text{a}\cdot}$) with masculine (x_i^{m}) and feminine (x_i^{f}) gender as well as paradigm features u_i. μ maps each U to one of $\{u_0 \ldots u_4\}$. μ is randomly initialized and then kept fixed.

4.4 Accurate and consistent representation of multifacetedness

Grammar. The grammar shown in Figure 5 models two syntactic categories, nouns and adjectives, whose left context is highly predictable: it is one of five left context words n_i (resp. a_i) for nouns, see lines 1, 3, 5 (resp. for adjectives, see lines 2, 4, 6). There are two grammatical genders: feminine (corresponding to the two symbols F_n and F_a) and masculine (corresponding to the two symbols M_n and M_a). The four combinations of syntactic category and gender are equally probable (lines 1–4). In addition to *gender*, nouns and adjectives are distinguished with respect to *morphological paradigm*. Line 7 generates one of five feminine nouns (x_i^{nf}) and the corresponding paradigm marker U_i^{nf}. A noun has two equally probable right contexts: a context indicating its gender (line 8) and a context indicating its paradigm (line 9). μ is a function that maps each U to one of five morphological paradigms $\{u_0 \ldots u_4\}$. μ is randomly initialized before a corpus is generated and kept fixed.

The function μ models the assignment of

paradigms to nouns and adjectives. Nouns and adjectives can have different (or the same) paradigms, but for a given noun or adjective the paradigm is fixed and does not change. Lines 7–9 generate gender and paradigm markers for feminine nouns, for which we use the symbols x_i^{nf}. Lines 10–18 cover the three other cases: masculine nouns (x_i^{nm}), feminine adjectives (x_i^{af}) and masculine adjectives (x_i^{am}).

Dataset. We perform 10 trials. In each trial, μ is initialized randomly and a corpus of 100,000 sentences is generated. The train set consists of the feminine nouns (x_i^{nf}, line 7) and the masculine nouns (x_i^{nm}, line 13). The test set consists of the feminine (x_i^{af}) and masculine (x_i^{am}) adjectives.

Results. Embeddings have been learned, SVMs are trained on the binary classification task feminine vs. masculine and evaluated on test. There was not a single error: accuracy of classifications is 100% for all embedding models.

Discussion. The facet gender is indicated directly by the distribution and easy to learn. For a noun or adjective x, we simply have to check whether f or m occurs to its right anywhere in the corpus. PPMI stores this information in two dimensions of the vectors and the SVM learns this fact perfectly. The encoding of "f or m occurs to the right" is less direct in the learning-based representation of x, but the experiment demonstrates that they also reliably encode it and the SVM reliably picks it up.

It would be possible to encode the facet in just one bit in a manually designed representation. While all representations are less compact than a one-bit representation – PPMI uses two real dimensions, learning-based models use an activation pattern over several dimensions – it is still true that most of the capacity of the embeddings is used for encoding facets other than gender: syntactic categories and paradigms. Note that there are five different instances each of feminine/masculine adjectives, feminine/masculine nouns and u_i words, but only two gender indicators: f and m. This is a typical scenario across languages: words are distinguished on a large number of morphological, grammatical, semantic and other dimensions and each of these dimensions corresponds to a small fraction of the overall knowledge we have about a given word.

Point-based tests do not directly evaluate specific facets of words. In similarity datasets, there is no individual test on facets – only full-space similarity is considered. There are test cases in analogy that hypothetically evaluate specific facets like gender of words, as in king-man+woman=queen. However, it does not consider the impact of other facets and assumes the only difference of "king" and "queen" is gender. A clear example that words usually differ on many facets, not just one, is the analogy: London:England $\sim$ Ankara:Turkey. *political-capital-of* applies to both, *cultural-capital-of* only to London:England since Istanbul is the cultural capital of Turkey.

To make our argument more clear, we designed an additional experiment that tries to evaluate gender in our dataset based on similarity and analogy methods. In the *similarity evaluation*, we search for the nearest neighbor of each word and accuracy is the proportion of nearest neighbors that have the same gender as the search word. In the *analogy evaluation*, we randomly select triples of the form $<x_i^{c_1 g_1}, x_j^{c_1 g_2}, x_k^{c_2 g_2}>$ where $(c_1, c_2) \in \{(\text{noun}, \text{adjective}), (\text{adjective}, \text{noun})\}$ and $(g_1, g_2) \in \{(\text{masculine}, \text{feminine}), (\text{feminine}, \text{masculine})\}$. We then compute $\vec{s} = \vec{x}_i^{c_1 g_1} - \vec{x}_j^{c_1 g_2} + \vec{x}_k^{c_2 g_2}$ and identify the word whose vector is closest to $\vec{s}$ where the three vectors $\vec{x}_i^{c_1 g_1}$, $\vec{x}_j^{c_1 g_2}$, $\vec{x}_k^{c_2 g_2}$ are excluded. If the nearest neighbor of $\vec{s}$ is of type $\vec{x}_l^{c_2 g_1}$, then the search is successful; e.g., for $\vec{s} = \vec{x}_i^{\text{nf}} - \vec{x}_j^{\text{nm}} + \vec{x}_k^{\text{am}}$, the search is successful if the nearest neighbor is feminine. We did this evaluation on the same test set for PPMI and LBL embedding models. Error rates were 29% for PPMI and 25% for LBL (similarity) and 16% for PPMI and 14% for LBL (analogy). This high error, compared to 0% error for SVM classification, indicates it is not possible to determine the presence of a low entropy facet accurately and consistently when full-space similarity and analogy are used as test criteria.

5 Analysis

In this section, we first summarize and analyze the lessons we learned through experiments in Section 4. After that, we show how these lessons are supported by a real natural-language corpus.

5.1 Learned lessons

(i) Two words with clearly different context distributions should receive different representations. Aggregation models fail to do so by calculating

	all entities		head entities		tail entities	
	MLP	1NN	MLP	1NN	MLP	1NN
PPMI	61.6	44.0	69.2	63.8	43.0	28.5
LBL	63.5	51.7	72.7	66.4	44.1	32.8
CBOW	63.0	53.5	71.7	69.4	39.1	29.9
CWIN	66.1	53.0	73.5	68.6	46.8	31.4
SKIP	64.5	**57.1**	69.9	**71.5**	**49.8**	**34.0**
SSKIP	**66.2**	52.8	**73.9**	68.5	45.5	31.4

Table 1: Entity typing results using embeddings learned with different models.

global statistics.

(ii) Embedding learning can have different effectiveness for sparse vs. non-sparse events. Thus, models of representations should be evaluated with respect to their ability to deal with sparseness; evaluation data sets should include rare as well as frequent words.

(iii) Our results in Section 4.3 suggest that single-representation approaches can indeed represent different senses of a word. We did a classification task that roughly corresponds to the question: does this word have a particular meaning? A representation can fail on similarity judgement computations because less frequent senses occupy a small part of the capacity of the representation and therefore have little impact on full-space similarity values. Such a failure does not necessarily mean that a particular sense is not present in the representation and it does not necessarily mean that single-representation approaches perform poor on real-world tasks. However, we saw that even though single-representations do well on balanced senses, they can pose a challenge for ambiguous words with skewed senses.

(iv) Lexical information is complex and multifaceted. In point-based tests, all dimensions are considered together and their ability to evaluate specific facets or properties of a word is limited. The full-space similarity of a word may be highest to a word that has a different value on a low-entropy facet. Any good or bad result on these tasks is not sufficient to conclude that the representation is weak. The valid criterion of quality is whether information about the facet is consistently and accurately stored.

5.2 Extrinsic evaluation: entity typing

To support the case for sub-space evaluation and also to introduce a new extrinsic task that uses the embeddings directly in supervised classification, we address a *fine-grained entity typing* task.

Learning taxonomic properties or types of words has been used as an evaluation method for word embeddings (Rubinstein et al., 2015). Since available word typing datasets are quite small (cf. Baroni et al. (2014), Rubinstein et al. (2015)), entity typing can be a promising alternative, which enables to do supervised classification instead of unsupervised clustering. Entities, like other words, have many properties and therefore belong to several semantic types, e.g., "Barack Obama" is a POLITICIAN, AUTHOR and AWARD_WINNER. We perform entity typing by learning types of knowledge base entities from their embeddings; this requires looking at subspaces because each entity can belong to multiple types.

We adopt the setup of Yaghoobzadeh and Schütze (2015) who present a dataset of Freebase entities;[5] there are 102 types (e.g., POLITICIAN FOOD, LOCATION-CEMETERY) and most entities have several. More specifically, we use a multilayer-perceptron (MLP) with one hidden layer to classify entity embeddings to 102 FIGER types. To show the limit of point-based evaluation, we also experimentally test an entity typing model based on *cosine similarity* of entity embeddings. To each test entity, we assign all types of the entity closest to it in the train set. We call this approach 1NN (kNN for $k = 1$).[6]

We take part of ClueWeb, which is annotated with Freebase entities using automatic annotation of FACC1[7] (Gabrilovich et al., 2013), as our corpus. We then replace all mentions of entities with their Freebase identifier and learn embeddings of words and entities in the same space. Our corpus has around 6 million sentences with at least one annotated entity. We calculate embeddings using our different models. Our hyperparameters: for learning-based models: dim=100, neg=10, iterations=20, window=1, sub=10^{-3}; for PPMI: SVD-dim=100, neg=1, window=1, cds=0.75, sub=10^{-3}, eig=0.5. See (Levy et al., 2015) for more information about the meaning of hyperparameters.

Table 1 gives results on test for all (about 60,000 entities), head (freq $>$ 100; about 12,200 entities) and tail (freq $<$ 5; about 10,000 entities). The MLP models consistently outperform 1NN on

[5] cistern.cis.lmu.de/figment

[6] We tried other values of k, but results were not better.

[7] lemurproject.org/clueweb12/FACC1

all and tail entities. This supports our hypothesis that only part of the information about types that is present in the vectors can be determined by similarity-based methods that use the overall direction of vectors, i.e., full-space similarity.

There is little correlation between results of MLP and 1NN in all and head entities, and the correlation between their results in tail entities is high.[8] For example, for all entities, using 1NN, SKIP is 4.3% (4.1%) better, and using MLP is 1.7% (1.6%) worse than SSKIP (CWIN). The good performance of SKIP on 1NN using cosine similarity can be related to its objective function, which maximizes the cosine similarity of cooccuring token embeddings.

The important question is not similarity, but whether the information about a specific type exists in the entity embeddings or not. Our results confirm our previous observation that a classification by looking at subspaces is needed to answer this question. In contrast, based on full-space similarity, one can infer little about the quality of embeddings. Based on our results, SSKIP and CWIN embeddings contain more accurate and consistent information because MLP classifier gives better results for them. However, if we considered 1NN for comparison, SKIP and CBOW would be superior.

6 Conclusion and future work

We have introduced a new way of evaluating distributional representation models. As an alternative to the common evaluation tasks, we proposed to identify generic criteria that are important for an embedding model to represent properties of words accurately and consistently. We suggested four criteria based on fundamental characteristics of natural language and designed tests that evaluate models on the criteria. We developed this evaluation methodology using PCFG-generated corpora and applied it on a case study to compare different models of learning distributional representations.

While we showed important differences of the embedding models, the goal was not to do a comprehensive comparison of them. We proposed an innovative way of doing intrinsic evaluation of embeddings. Our evaluation method gave direct insight about the quality of embeddings. Additionally, while most intrinsic evaluations consider word vectors as points, we used classifiers that identify different small subspaces of the full space. This is an important desideratum when designing evaluation methods because of the multifacetedness of natural language words: they have a large number of properties, each of which only occupies a small proportion of the full-space capacity of the embedding.

Based on this paper, there are serveral lines of investigation we plan to conduct in the future. (i) We will attempt to support our results on artificially generated corpora by conducting experiments on *real natural language data*. (ii) We will study the *coverage of our four criteria* in evaluating word representations. (iii) We modeled the four criteria using separate PCFGs, but they could also be modeled by one single unified PCFG. One question that arises is then to what extent the four criteria are orthogonal and to what extent interdependent. A single unified grammar may make it harder to interpret the results, but may give additional and more fine-grained insights as to how the performance of embedding models is influenced by different fundamental properties of natural language and their interactions.

Finally, we have made the simplifying assumption in this paper that the best conceptual framework for thinking about embeddings is that the embedding space can be *decomposed into subspaces*: either into completely orthogonal subspaces or – less radically – into partially "overlapping" subspaces. Furthermore, we have made the assumption that the smoothness and robustness properties that are the main reasons why embeddings are used in NLP can be reduced to *similarities in subspaces*. See Rothe et al. (2016) and Rothe and Schütze (2016) for work that makes similar assumptions.

The fundamental assumptions here are decomposability and linearity. The smoothness properties could be much more complicated. However even if this was the case, then much of the general framework of what we have presented in this paper would still apply; e.g., the criterion that a particular facet be fully and correctly represented is as important as before. But the validity of the assumption that embedding spaces can be decomposed into "linear" subspaces should be investigated in the future.

Acknowledgments. This work was supported by DFG (SCHU 2246/8-2).

[8] The spearman correlation between MLP and 1NN for all=0.31, head=0.03, tail=0.75.

References

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA*, pages 19–27.

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

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pages 238–247.

Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. Blackwell, 2nd edition.

D. A. Cruse. 1986. *Lexical Semantics*. Cambridge University Press, Cambridge, MA.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 873–882.

Arne Köhn. 2015. What?s in an embedding? analyzing word embeddings through multilingual evaluation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2067–2073, Lisbon, Portugal, September.

Siwei Lai, Kang Liu, Liheng Xu, and Jun Zhao. 2015. How to generate a good word embedding? *CoRR*, abs/1507.05523.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *CoNLL*.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal, September.

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 1299–1304.

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

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *NIPS*, pages 2265–2273.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Joseph Reisinger and Raymond J Mooney. 2010. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze. 2016. Word embedding calculus in meaningful ultradense subspaces. In *ACL*.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense embeddings by orthogonal transformation. In *NAACL*.

Dana Rubinstein, Effi Levi, Roy Schwartz, and Ari Rappoport. 2015. How well do distributional models capture different types of semantic knowledge? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 2: Short Papers*, pages 726–730.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 298–307, Lisbon, Portugal, September.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2049–2054, Lisbon, Portugal, September.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res. (JAIR)*, 37:141–188.

Yadollah Yaghoobzadeh and Hinrich Schütze. 2015. Corpus-level fine-grained entity typing using contextual information. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 715–725, Lisbon, Portugal, September.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, System Demonstrations*, pages 78–83.

On the Role of Seed Lexicons in Learning Bilingual Word Embeddings

Ivan Vulić and **Anna Korhonen**
Language Technology Lab
DTAL, University of Cambridge
{iv250, alk23}@cam.ac.uk

Abstract

A shared bilingual word embedding space (SBWES) is an indispensable resource in a variety of cross-language NLP and IR tasks. A common approach to the SBWES induction is to learn a mapping function between monolingual semantic spaces, where the mapping critically relies on a seed word lexicon used in the learning process. In this work, we analyze the importance and properties of seed lexicons for the SBWES induction across different dimensions (i.e., lexicon source, lexicon size, translation method, translation pair reliability). On the basis of our analysis, we propose a simple but effective hybrid bilingual word embedding (BWE) model. This model (HYBWE) learns the mapping between two monolingual embedding spaces using only highly reliable symmetric translation pairs from a seed document-level embedding space. We perform bilingual lexicon learning (BLL) with 3 language pairs and show that by carefully selecting reliable translation pairs our new HYBWE model outperforms benchmarking BWE learning models, all of which use more expensive bilingual signals. Effectively, we demonstrate that a SBWES may be induced by leveraging only a very weak bilingual signal (document alignments) along with monolingual data.

1 Introduction

Dense real-valued vector representations of words or word embeddings (WEs) have recently gained increasing popularity in natural language processing (NLP), serving as invaluable features in a broad range of NLP tasks, e.g., (Turian et al., 2010; Collobert et al., 2011; Chen and Manning, 2014). Several studies have showcased a direct link and comparable performance to "more traditional" distributional models (Turney and Pantel, 2010). Yet the widely used skip-gram model with negative sampling (SGNS) (Mikolov et al., 2013b) is considered as the state-of-the-art word representation model, due to its simplicity, fast training, as well as its solid and robust performance across a wide variety of semantic tasks (Baroni et al., 2014; Levy and Goldberg, 2014b; Levy et al., 2015).

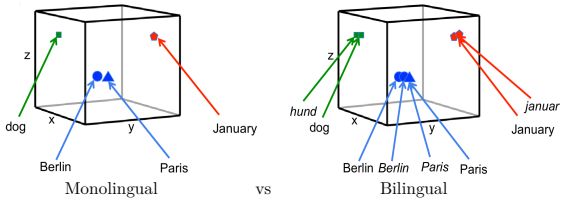


Figure 1: A toy example of a 3-dimensional monolingual vs shared bilingual word embedding space (further SBWES) from Gouws et al. (2015).

Research interest has recently extended to bilingual word embeddings (BWEs). BWE learning models focus on the induction of a *shared bilingual word embedding space* (SBWES) where words from both languages are represented in a uniform language-independent manner such that similar words (regardless of the actual language) have similar representations (see Fig. 1). A variety of BWE learning models have been proposed, differing in the essential requirement of a *bilingual signal* necessary to construct such a SBWES (discussed later in Sect. 2). SBWES may be used to support many tasks, e.g., computing cross-lingual/multilingual semantic word similarity (Faruqui and Dyer, 2014), learning bilingual word lexicons (Mikolov et al., 2013a; Gouws et al., 2015; Vulić et al., 2016), cross-lingual entity linking (Tsai and Roth, 2016),

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 247–257,
Berlin, Germany, August 7-12, 2016.

parsing (Guo et al., 2015; Johannsen et al., 2015), machine translation (Zou et al., 2013), or cross-lingual information retrieval (Vulić and Moens, 2015; Mitra et al., 2016).

BWE models should have two desirable properties: **(P1)** leverage (large) monolingual training sets tied together through a bilingual signal, **(P2)** use as inexpensive bilingual signal as possible in order to learn a SBWES in a scalable and widely applicable manner across languages and domains.

While we provide a classification of related work, that is, different BWE models according to these properties in Sect. 2.1, the focus of this work is on a popular class of models labeled *Post-Hoc Mapping with Seed Lexicons*. These models operate as follows (Mikolov et al., 2013a; Dinu et al., 2015; Lazaridou et al., 2015; Ammar et al., 2016): (1) two separate non-aligned monolingual embedding spaces are induced using any monolingual WE learning model (SGNS is the typical choice), (2) given a *seed lexicon* of word translation pairs as the bilingual signal for training, a mapping function is learned which ties the two monolingual spaces together into a SBWES.

All existing work on this class of models assumes that high-quality training seed lexicons are readily available. In reality, little is understood regarding what constitutes a high quality seed lexicon, even with "traditional" distributional models (Gaussier et al., 2004; Holmlund et al., 2005; Vulić and Moens, 2013). Therefore, in this work we ask *whether BWE learning could be improved by making more intelligent choices when deciding over seed lexicon entries*. In order to do this we delve deeper into the cross-lingual mapping problem by analyzing a spectrum of seed lexicons with respect to controllable parameters such as lexicon source, its size, translation method, and translation pair reliability.

The contributions of this paper are as follows:

(C1) We present a systematic study on the importance of seed lexicons for learning mapping functions between monolingual WE spaces.

(C2) Given the insights gained, we propose a simple yet effective hybrid BWE model HYBWE that removes the need for readily available seed lexicons, and satisfies properties P1 and P2. HYBWE relies on an inexpensive seed lexicon of highly reliable word translation pairs obtained by a document-level BWE model (Vulić and Moens, 2016) from document-aligned comparable data.

(C3) Using a careful pair selection process when constructing a seed lexicon, we show that in the BLL task HYBWE outperforms a BWE model of Mikolov et al. (2013a) which relies on readily available seed lexicons. HYBWE also outperforms state-of-the-art models of (Hermann and Blunsom, 2014b; Gouws et al., 2015) which require sentence-aligned parallel data.

2 Learning SBWES using Seed Lexicons

Given source and target language vocabularies V^S and V^T, all BWE models learn a representation of each word $w \in V^S \sqcup V^T$ in a SBWES as a real-valued vector: $\mathbf{w} = [f_1, \ldots, f_d]$, where $f_k \in \mathbb{R}$ denotes the value for the k-th cross-lingual feature for w within a d-dimensional SBWES. Semantic similarity $sim(w, v)$ between two words $w, v \in V^S \sqcup V^T$ is then computed by applying a similarity function (SF), e.g. cosine (*cos*) on their representations in the SBWES: $sim(w, v) = SF(\mathbf{w}, \mathbf{v}) = cos(\mathbf{w}, \mathbf{v})$.

2.1 Related Work: BWE Models and Bilingual Signals

BWE models may be clustered into four different types according to bilingual signals used in training, and properties P1 and P2 (see Sect. 1). Upadhyay et al. (2016) provide a similar overview of recent bilingual embedding learning architectures regarding different bilingual signals required for the embedding induction.

(Type 1) *Parallel-Only*: This group of BWE models relies on sentence-aligned and/or word-aligned parallel data as the only data source (Zou et al., 2013; Hermann and Blunsom, 2014a; Kočiský et al., 2014; Hermann and Blunsom, 2014b; Chandar et al., 2014). In addition to an expensive bilingual signal (colliding with P2), these models do not leverage larger monolingual datasets for training (not satisfying P1).

(Type 2) *Joint Bilingual Training*: These models jointly optimize two monolingual objectives, with the cross-lingual objective acting as a cross-lingual regularizer during training (Klementiev et al., 2012; Gouws et al., 2015; Soyer et al., 2015; Shi et al., 2015; Coulmance et al., 2015). The idea may be summarized by the simplified formulation (Luong et al., 2015): $\gamma(Mono_S + Mono_T) + \delta Bi$. The monolingual objectives $Mono_S$ and $Mono_T$ ensure that similar words in each language are assigned similar

embeddings and aim to capture the semantic structure of each language, whereas the cross-lingual objective Bi ensures that similar words across languages are assigned similar embeddings. It ties the two monolingual spaces together into a SBWES (thus satisfying P1). Parameters γ and δ govern the influence of the monolingual and bilingual components.[1] The main disadvantage of Type 2 models is the costly parallel data needed for the bilingual signal (thus colliding with P2).

(Type 3) *Pseudo-Bilingual Training*: This set of models requires *document alignments* as bilingual signal to induce a SBWES. Vulić and Moens (2016) create a collection of pseudo-bilingual documents by merging every pair of aligned documents in training data, in a way that preserves important local information: words that appeared next to other words within the same language and those that appeared in the same region of the document across different languages. This collection is then used to train word embeddings with monolingual SGNS from `word2vec`.

With pseudo-bilingual documents, the "context" of a word is redefined as a mixture of neighbouring words (in the original language) and words that appeared in the same region of the document (in the "foreign" language). The bilingual contexts for each word in each document steer the final model towards constructing a SBWES. The advantage over other BWE model types lies in exploiting weaker document-level bilingual signals (satisfying P2), but these models are unable to exploit monolingual corpora during training (unlike Type 2 or Type 4; thus colliding with P1).

(Type 4) *Post-Hoc Mapping with Seed Lexicons*: These models learn post-hoc mapping functions between monolingual WE spaces induced separately for two different languages (e.g., by SGNS). All Type 4 models (Mikolov et al., 2013a; Faruqui and Dyer, 2014; Dinu et al., 2015; Lazaridou et al., 2015) rely on readily available seed lexicons of highly frequent words obtained by e.g. *Google Translate* (GT) to learn the mapping (again colliding with P2), but they are able to satisfy P1.

[1]Type 1 models may be considered a special case of Type 2 models: Setting $\gamma = 0$ reduces Type 2 models to Type 1 models trained solely on parallel data, e.g., (Hermann and Blunsom, 2014b; Chandar et al., 2014). $\gamma = 1$ results in the models from (Klementiev et al., 2012; Gouws et al., 2015; Soyer et al., 2015; Coulmance et al., 2015).

2.2 Post-Hoc Mapping with Seed Lexicons: Methodology and Lexicons

Key Intuition One may infer that a type-*hybrid* procedure which would retain only highly reliable translation pairs obtained by a Type 3 model as a seed lexicon for Type 4 models effectively satisfies both requirements: (P1) unlike Type 1 and Type 3, it can learn from monolingual data and tie two monolingual spaces using the highly reliable translation pairs, (P2) unlike Type 1 and Type 2, it does not require parallel data; unlike Type 4, it does not require external lexicons and translation systems. The only bilingual signal required are document alignments. Therefore, our focus is on novel less expensive Type 4 models.

Overview The standard learning setup we use is as follows: First, two monolingual embedding spaces, $\mathbb{R}^{d_S}$ and $\mathbb{R}^{d_T}$, are induced separately in each of the two languages using a standard monolingual WE model such as CBOW or SGNS. d_S and d_T denote the dimensionality of monolingual WE spaces. The bilingual signal is a seed lexicon, i.e., a list of word translation pairs (x_i, y_i), where $x_i \in V^S$, $y_i \in V^T$, and $\mathbf{x_i} \in \mathbb{R}^{d_S}$, $\mathbf{y_i} \in \mathbb{R}^{d_T}$.

Learning Objectives Training is cast as a multivariate regression problem: it implies learning a function that maps the source language vectors from the training data to their corresponding target language vectors. A standard approach (Mikolov et al., 2013a; Dinu et al., 2015) is to assume a linear map $\mathbf{W} \in \mathbb{R}^{d_S \times d_T}$, where a L_2-regularized least-squares error objective (i.e., ridge regression) is used to learn the map $\mathbf{W}$. The map is learned by solving the following optimization problem (typically by stochastic gradient descent (SGD)):

$$\min_{\mathbf{W} \in \mathbb{R}^{d_S \times d_T}} ||\mathbf{XW} - \mathbf{Y}||_F^2 + \lambda ||\mathbf{W}||_F^2 \quad (1)$$

$\mathbf{X}$ and $\mathbf{Y}$ are matrices obtained through the respective concatenation of source language and target language vectors from training pairs. Once the linear map $\mathbf{W}$ is estimated, any previously unseen source language word vector $\mathbf{x_u}$ may be straightforwardly mapped into the target language embedding space $\mathbb{R}^{d_T}$ as $\mathbf{W x_u}$. After mapping all vectors $\mathbf{x}$, $x \in V^S$, the target embedding space $\mathbb{R}^{d_T}$ in fact serves as SBWES.[2]

[2]Another possible objective (found in the zero-shot learning literature) is a margin-based ranking loss (Weston et al., 2011; Lazaridou et al., 2015). We omit the results with this objective for brevity, and due to the fact that similar trends are observed as with (more standard) linear maps.

Seed Lexicon Source and Translation Method Prior work on post-hoc mapping with seed lexicons used a translation system (i.e., GT) to translate highly frequent English words to other languages such as Czech, Spanish (Mikolov et al., 2013a; Gouws et al., 2015) or Italian (Dinu et al., 2015; Lazaridou et al., 2015). This method presupposes the availability and high quality of such an external translation system. To simulate this setup, we take as a starting point the BNC word frequency list from Kilgarriff (1997) containing 6,318 most frequent English lemmas. The list is then translated to other languages via GT. We call the BNC-based lexicons obtained by employing Google Translate BNC+GT.

In this paper, we propose another option: first, we learn the "first" SBWES (i.e., SBWES-1) using another BWE model (see Sect. 2.1), and then translate the BNC list through SBWES-1 by retaining the nearest cross-lingual neighbor $y_i \in V^T$ for each x_i in the BNC list which is represented in SBWES-1. The pairs (x_i, y_i) constitute the seed lexicon needed for learning the mapping between monolingual spaces, that is, to induce the final SBWES-2.

Although in theory any BWE induction model may be used to induce SBWES-1, we rely on a document-level Type 3 BWE induction model from (Vulić and Moens, 2016), since it requires only document alignments as (weak) bilingual signal. The resulting hybrid BWE induction model (HYBWE) combines the output of a Type 3 model (SBWES-1) and a Type 4 model (SBWES-2). This seed lexicon and BWE learning variant is called BNC+HYB.

Our new hybrid model allows us to also use source language words occurring in SBWES-1 sorted by frequency as seed lexicon source, again leaning on the intuition that higher frequency phenomena are more reliably translated using statistical models. Their translations can also be found through SBWES-1 to obtain seed lexicon pairs (x_i, y_i). This variant is called HFQ+HYB.

Another possibility, recently introduced by Kiros et al. (2015) for vocabulary expansion in monolingual settings, relies on all words shared between two vocabularies to learn the mapping. In this work, we test the ability and limits of such orthographic evidence in cross-lingual settings: seed lexicon pairs are (x_i, x_i), where $x_i \in V^S$ and $x_i \in V^T$. This seed lexicon variant is called ORTHO.

Seed Lexicon Size While all prior reported only results with restricted seed lexicon sizes only (i.e., 1K, 2K and 5K lexicon pairs are used as standard), in this work we provide a full-fledged analysis of the influence of seed lexicon size on the SBWES performance in cross-lingual tasks. More extreme settings are also investigated, in the attempt to answer two important questions: (1) Can a Type 4 SBWES be induced in a limited setting with only a few hundred lexicon pairs available (e.g., 100-500)? (2) Can the Type 4 models profit from the inclusion of more seed lexicon pairs (e.g., more than 5K, even up to 40K-50K lexicon pairs)?

Translation Pair Reliability When building seed lexicons through SBWES-1 (i.e., BNC+HYB and HFQ+HYB methods), it is possible to control for the reliability of translation pairs to be included in the final lexicon, with the idea that the use of only highly reliable pairs can potentially lead to an improved SBWES-2. A simple yet effective reliability reliability feature for translation pairs is the *symmetry constraint* (Peirsman and Padó, 2010; Vulić and Moens, 2013) : two words $x_i \in V^S$ and $y_i \in V^S$ are used as seed lexicon pairs only if they are mutual nearest neighbours given their representations in SBWES-1. The two variants of seed lexicons with only symmetric pairs are BNC+HYB+SYM and HFREQ+HYB+SYM. We also test the variants without the symmetry constraint (i.e., BNC+HYB+ASYM and HFQ+HYB+ASYM).

Even more conservative reliability measures may be applied by exploiting the scores in the lists of translation candidates ranked by their similarity to the cue word x_i. We investigate a symmetry constraint with a *threshold*: two words $x_i \in V^S$ and $y_i \in V^S$ are included as seed lexicon pair (x_i, y_i) iff they are mutual nearest neighbours in SBWES-1 and it holds:

$$sim(x_i, y_i) - sim(x_i, z_i) > THR \quad (2)$$
$$sim(y_i, x_i) - sim(y_i, w_i) > THR \quad (3)$$

where $z_i \in V^T$ is the second best translation candidate for x_i, and $w_i \in V^S$ for y_i. THR is a parameter which specifies the margin between the two best translation candidates. The intuition is that highly unambiguous and monosemous translation pairs (which is reflected in higher score margins) are also highly reliable.[3]

[3]Other (more elaborate) reliability measures exist in the

3 Experimental Setup

Task: Bilingual Lexicon Learning (BLL) After the final SBWES is induced, given a list of n source language words $x_{u1}, \ldots, x_{un}$, the task is to find a target language word t for each x_u in the list using the SBWES. t is the target language word closest to the source language word x_u in the induced SBWES, also known as the *cross-lingual nearest neighbor*. The set of learned n (x_u, t) pairs is then run against a gold standard BLL test set. Following the standard practice (Mikolov et al., 2013a; Dinu et al., 2015), for all Type 4 models, all pairs containing any of the test words $x_{u1}, \ldots, x_{un}$ are removed from training seed lexicons.

Test Sets For each language pair, we evaluate on standard 1,000 ground truth one-to-one translation pairs built for three language pairs: Spanish (ES)-, Dutch (NL)-, Italian (IT)-English (EN) by Vulić and Moens (2013). The dataset is generally considered a benchmarking test set for BLL models that learn from non-parallel data, and is available online.[4] We have also experimented with two other benchmarking BLL test sets (Bergsma and Durme, 2011; Leviant and Reichart, 2015) observing a very similar relative performance of all the models in our comparison.

Evaluation Metrics We measure the BLL performance using the standard *Top 1* accuracy (Acc_1) metric (Gaussier et al., 2004; Mikolov et al., 2013a; Gouws et al., 2015).[5]

Baseline Models To induce SBWES-1, we resort to document-level embeddings of Vulić and Moens (2016) (Type 3). We also compare to results obtained directly by their model (BWESG) to measure the performance gains with HYBWE.

To compare with a representative Type 2 model, we opt for the BilBOWA model of Gouws et al. (2015) due to its solid performance and robustness in the BLL task when trained on general-domain corpora such as Wikipedia (Luong et al., 2015), its reduced complexity reflected in fast computations on massive datasets, as well as its public availability.[6] In short, BilBOWA combines the adapted SGNS for monolingual objectives together with a cross-lingual objective that minimizes the L_2-loss between the bag-of-word vectors of parallel sentences. BilBOWA uses the same training setup as HYBWE (monolingual datasets plus a bilingual signal), but relies on a stronger bilingual signal (sentence alignments as opposed to HYBWE's document alignments).

We also compare with a benchmarking Type 1 model from sentence-aligned parallel data called BiCVM (Hermann and Blunsom, 2014b). Finally, a SGNS-based BWE model with the BNC+GT seed lexicon is taken as a baseline Type 4 model (Mikolov et al., 2013a).[7]

Training Data and Setup We use standard training data and suggested settings to obtain BWEs for all models involved in comparison. We retain the 100K most frequent words in each language for all models. To induce monolingual WE spaces, two monolingual SGNS models were trained on the cleaned and tokenized Wikipedias from the Polyglot website (Al-Rfou et al., 2013) using SGD with a global learning rate of 0.025. For BilBOWA, as in the original work (Gouws et al., 2015), the bilingual signal for the cross-lingual regularization is provided by the first 500K sentences from Europarl.v7 (Tiedemann, 2012). We use SGD with a global rate of 0.15.[8] The window size is varied from 2 to 16 in steps of 2, and the best scoring model is always reported in all comparisons.

BWESG was trained on the cleaned and tokenized document-aligned Wikipedias available online[9], SGD on pseudo-bilingual documents with a global rate 0.025. For BiCVM, we use the tool released by its authors[10] and train on the whole Europarl.v7 for each language pair: we train an additive model, with hinge loss margin set to d (i.e., dimensionality) as in the original paper, batch size of 50, and noise parameter of 10. All BiCVM models are trained with 200 iterations.

For all models, we obtain BWEs with $d = 40, 64, 300, 500$, but we report only results with 300-dimensional BWEs as similar trends were observed with other d-s. Other parameters are: 15 epochs, 15 negatives, subsampling rate $1e-4$.

literature (Smith and Eisner, 2007; Tu and Honavar, 2012; Vulić and Moens, 2013), but we do not observe any significant gains when resorting to the more complex reliability estimates.

[4] http://people.cs.kuleuven.be/~ivan.vulic/

[5] Similar trends are observed within a more lenient setting with Acc_5 and Acc_{10} scores, but we omit these results for clarity and the fact that the actual BLL performance is best reflected in Acc_1 scores (i.e., best translation only).

[6] https://github.com/gouwsmeister/bilbowa

[7] For details concerning all baseline models, the reader is encouraged to check the relevant literature.

[8] Suggested by the authors (personal correspondence).

[9] http://linguatools.org/tools/corpora/

[10] https://github.com/karlmoritz/bicvm

BNC+GT	BNC+HYB+ASYM	BNC+HYB+SYM	HFQ+HYB+ASYM	HFQ+HYB+SYM	ORTHO
casamiento	*casamiento*	*casamiento*	*casamiento*	*casamiento*	*casamiento*
marriage	marry	*marriage*	*marriage*	*marriage*	maría
marry	*marriage*	marry	marry	marry	señor
marrying	marrying	marrying	betrothal	betrothal	doña
betrothal	wed	wedding	marrying	marrying	juana
wedding	wedding	betrothal	wedding	wedding	noche
wed	betrothal	wed	daughter	wed	amor
elopement	remarry	marriages	betrothed	elopement	guerra

Table 1: Nearest EN neighbours of the Spanish word *casamiento (marriage)* with different seed lexicons.

Model	ES-EN	NL-EN	IT-EN
BICVM (TYPE 1)	0.532	0.583	0.569
BILBOWA (TYPE 2)	0.632	0.636	0.647
BWESG (TYPE 3)	0.676	0.626	0.643
BNC+GT (Type 4)	0.677	0.641	0.646
ORTHO	0.233	0.506	0.224
BNC+HYB+ASYM	0.673	0.626	0.644
BNC+HYB+SYM (3388; 2738; 3145)	0.681	**0.658***	0.663*
HFQ+HYB+ASYM	0.673	0.596	0.635
HFQ+HYB+SYM	**0.695***	0.657*	**0.667***

Table 2: Acc_1 scores in a standard BLL setup (for Type 4 models): all seed lexicons contain 5K translation pairs, except for BNC+HYB+SYM (its sizes provided in parentheses). * denotes a statistically significant improvement over baselines and BNC+GT using McNemar's statistical significance test with the Bonferroni correction, $p < 0.05$.

4 Results and Discussion

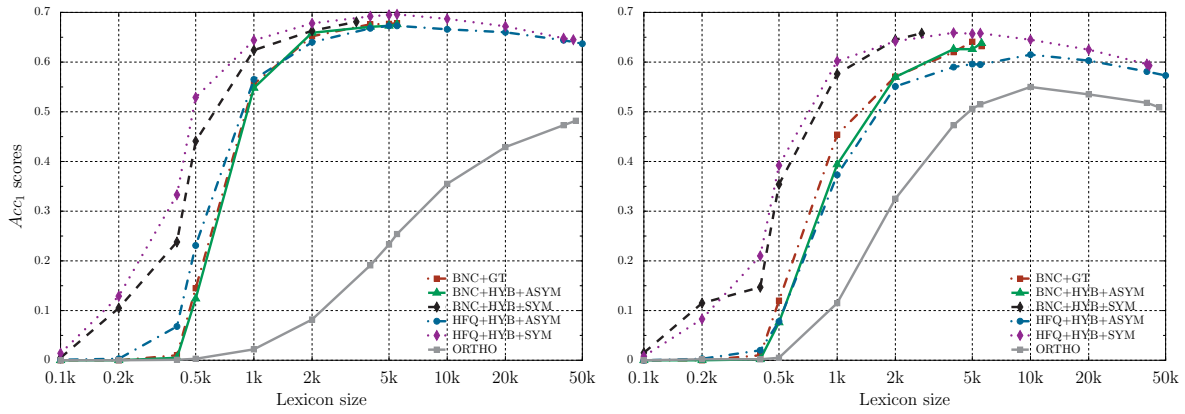
Exp. I: Standard BLL Setting First, we replicate the previous BLL setups with Type 4 models from (Mikolov et al., 2013a; Dinu et al., 2015) by relying on seed lexicons of exactly 5K word pairs (except for BNC+HYB+SYM which exhausts all possible pairs before the 5K limit) sorted by frequency of the source language word. Results with different lexicons for the three language pairs are summarized in Table 2, while Table 1 shows examples of nearest neighbour words for a Spanish word not present in any of the training lexicons.

Table 1 provides evidence for our first insight: Type 4 models do not necessarily require external lexicons (such as the BNC+GT model) to learn a semantically plausible SBWES (i.e., the lists of nearest neighbours are similar for all lexicons excluding ORTHO). Table 1 also suggests that the choice of seed lexicon pairs may strongly influence the properties of the resulting SBWES. Due to its design, ORTHO finds a mapping which naturally brings foreign words appearing in the English vocabulary closer in the induced SBWES.

This first batch of quantitative results already shows that Type 4 models with inexpensive automatically induced lexicons (i.e., HYBWE) are on a par with or even better than Type 4 models relying on external resources or translation systems. In addition, the best reported scores using the more constrained symmetric BNC/HFQ+HYB+SYM lexicon variants are higher than those for three baseline models (of Type 1, Type 2, and Type 3) that previously held highest scores on the BLL test sets (Vulić and Moens, 2016). These improvements over the baseline models and BNC+GT are statistically significant (using McNemar's statistical significance test, $p < 0.05$). Table 2 also suggests that a careful selection of reliable pairs can lead to peak performances even with a lower number of pairs, i.e., see the results of BNC+HYB+SYM.

Exp. II: Lexicon Size BLL results for ES-EN and NL-EN obtained by varying the seed lexicon sizes are displayed in Fig. 2(a) and 2(b). Results for IT-EN closely follow the patterns observed with ES-EN. BNC+HYB+SYM and HFQ+HYB+ASYM – the two models that do not blindly use all potential training pairs, but rely on sets of symmetric pairs (i.e., they include the simple measure of translation pair reliability) – display the best performance across all lexicon sizes. The finding confirms the intuition that a more intelligent pair selection strategy is essential for Type 4 BWE models. HFQ+HYB+SYM – a simple hybrid BWE model (HYBWE) combining a document-level Type 3 model with a Type 4 model and translation reliability detection – is the strongest BWE model overall (see also Table 2 again).

HYBWE-based models which do not perform any pair selection (i.e., BNC/HFQ+HYB+ASYM) closely follow the behaviour of the GT-based model. This demonstrates that an external lexicon or translation system may be safely replaced



(a) Spanish-English (b) Dutch-English

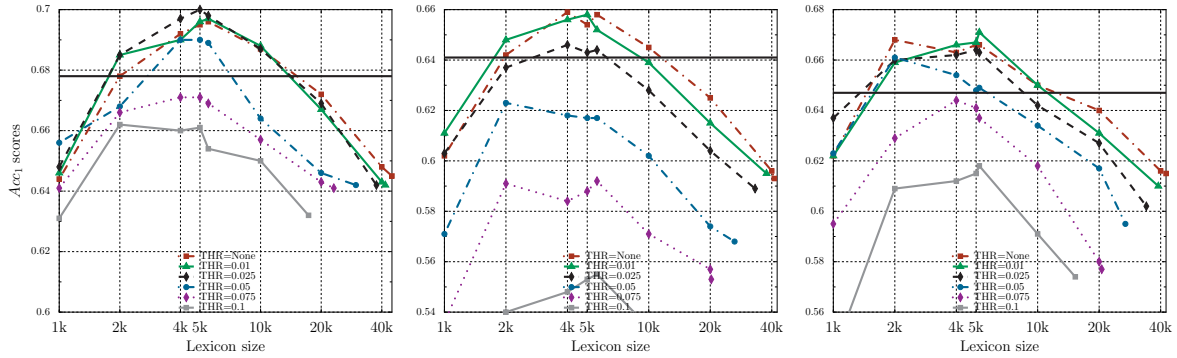
Figure 2: BLL results (Acc_1) across different seed lexicon sizes for all lexicons. x axes are in log scale.

by a document-level embedding model without any significant performance loss in the BLL task. The ORTHO-based model falls short of its competitors. However, we observe that even this model with the learning setting relying on the cheapest bilingual signal may lead to reasonable BLL scores, especially for the more related NL-EN pair.

The two models with the symmetry constraint display a particularly strong performance with settings relying on scarce resources (i.e., only a small portion of training pairs is available). For instance, HFQ+HYB+SYM scores 0.129 for ES-EN with only 200 training pairs (vs 0.002 with BNC+GT), and 0.529 with 500 pairs (vs 0.145 with BNC+GT). On the other hand, adding more pairs does not lead to an improved BLL performance. In fact, we observe a slow and steady decrease in performance with lexicons containing $10,000$ and more training pairs for all HYBWE variants. The phenomenon may be attributed to the fact that highly frequent words receive more accurate representations in SBWES-1, and adding less frequent and, consequently, less accurate training pairs to the SBWES-2 learning process brings in additional noise. In plain language, when it comes to seed lexicons Type 4 models prefer quality over quantity.

Exp. III: Translation Pair Reliability In the next experiment, we vary the threshold value THR (see sect. 2.2) in the HFQ+HYB+SYM variant with the following values in comparison: 0.0 (None), $0.01, 0.025, 0.05, 0.075, 0.1$. We investigate whether retaining only highly unambiguous pairs would lead to even better BLL performance. The results for all three language pairs are summarized in Fig. 3(a)-3(c). The results for all variant models again decrease when employing larger lexicons (due to the usage of less frequent word pairs in training). We observe that a slightly stricter selection criterion (i.e., THR $= 0.01, 0.025$) also leads to slightly improved peak BLL scores for ES-EN and IT-EN around the 5K region. The improvements, however, are not statistically significant. On the other hand, a too conservative pair selection criterion with higher threshold values significantly deteriorates the overall performance of HYBWE with HFQ+HYB+SYM. The conservative criteria discard plenty of potentially useful training pairs. Therefore, as one line of future research, we plan to investigate more sophisticated models for the selection of reliable seed lexicon pairs that will lead to a better trade-off between the lexicon size and reliability of the pairs.

Exp. IV: Another Task - Suggesting Word Translations in Context (SWTC) In the final experiment, we test whether the findings originating from the BLL task generalize to another cross-lingual semantic task: *suggesting word translations in context* (SWTC) recently proposed by Vulić and Moens (2014). Given an occurrence of a polysemous word $w \in V^S$, the SWTC task is to choose the correct translation in the target language of that particular occurrence of w from the given set $\mathcal{TC}(w) = \{t_1, \ldots, t_{tq}\}$, $\mathcal{TC}(w) \subseteq V^T$, of its tq possible translations/meanings. Whereas in the BLL task the candidate search is performed over the entire vocabulary V^T, the set $TC(w)$ typically comprises only a few pre-selected words/senses. One may refer to $\mathcal{TC}(w)$ as an inventory of translation candidates for w. The best scoring translation candidate in the ranked list is then the correct translation for that particular occurrence of w observing its local context $Con(w)$. SWTC is an extended



(a) Spanish-English (b) Dutch-English (c) Italian-English

Figure 3: BLL results across different threshold (THR) values with the HFQ+HYB+SYM seed lexicons. Higher thresholds imply less ambiguous word translation pairs. Thicker horizontal lines denote the best score from any of the baseline models. x axes are in log scale.

Model	**ES-EN**	**NL-EN**	**IT-EN**
NO CONTEXT	0.406	0.433	0.408
BEST SYSTEM (Vulić and Moens, 2014)	0.703	0.712	0.789
BICVM (TYPE 1)	0.506	0.586	0.522
BILBOWA (TYPE 2)	0.586	0.656	0.589
BWESG (TYPE 3)	0.783	0.858	0.792
BNC+GT (TYPE 4)	0.794	0.858	0.783
ORTHO	0.647	0.794	0.678
BNC+HYB+ASYM	0.806*	0.872	0.778
BNC+HYB+SYM (3839; 3117; 3693)	**0.808***	**0.875***	**0.814***
HFQ+HYB+ASYM	0.789	0.864	0.781
HFQ+HYB+SYM (THR = None)	0.792	0.869	0.786
HFQ+HYB+SYM (THR=0.01)	0.792	0.858	0.789
HFQ+HYB+SYM (THR=0.025)	0.800	0.853	0.792

Table 3: Acc_1 scores in the SWTC task. All seed lexicons contain 6K translation pairs, except for BNC+HYB+SYM (its sizes provided in parentheses). * denotes a statistically significant improvement over baselines and BNC+GT using McNemar's statistical significance test with the Bonferroni correction, $p < 0.05$.

cross-lingual variant of the task proposed by Huang et al. (2012) which evaluates monolingual context-sensitive semantic similarity of words in sentential context, and it is also very related to cross-lingual lexical substitution (Mihalcea et al., 2010).

To isolate the performance of each BWE induction model from the details of the SWTC setup, we use the same approach with all models: we opt for the SWTC framework proven to yield excellent results with BWEs in the SWTC task (Vulić and Moens, 2016). In short, the context bag $Con(w) = \{cw_1, \ldots, cw_r\}$ is obtained by harvesting all r words that occur with w in the sentence. The vector representation of $Con(w)$ is the d-dimensional embedding computed by aggregating over all word embeddings for each $cw_j \in Con(w)$ using standard *addition* as the compositional operator (Mitchell and Lapata, 2008) which was proven a robust choice (Milajevs et al., 2014):

$$\mathbf{Con(w)} = \mathbf{cw}_1 + \mathbf{cw}_2 + \ldots + \mathbf{cw}_r \quad (4)$$

where $\mathbf{cw}_j$ is the embedding of the j-th context word, and $\mathbf{Con(w)}$ is the resulting embedding of the context bag $Con(w)$. Finally, for each $t_j \in \mathcal{TC}(w)$, the context-sensitive similarity with w is computed as: $sim(w, t_j, Con(w)) = cos(\mathbf{Con(w)}, \mathbf{t}_j)$, where $\mathbf{Con(w)}$ and $\mathbf{t}_j$ are representations of the (sentential) context bag and the candidate translation t_j in the same SBWES.[11]

The evaluation set consists of 360 sentences for 15 polysemous nouns (24 sentences for each noun) in each of the three languages: Spanish, Dutch, Italian, along with the single gold standard single word English translation given the sentential context.[12] Table 3 summarizes the results (Acc_1 scores) in the SWTC task. NO-CONTEXT refers to the context-insensitive majority baseline obtained by BNC+GT (i.e., it always chooses the most semantically similar translation candidate at the word type level). We also report the results of the best SWTC model from Vulić and Moens (2014).

The results largely support the claims established with the BLL evaluation. An exter-

[11] The same ranking of different models (with lower absolute scores) is observed when adapting the monolingual lexical substitution framework of Melamud et al. (2015) to the SWTC task as done by Vulić and Moens (2016).

[12] The SWTC evaluation set is available online at: http://aclweb.org/anthology/attachments/D/D14/D14-1040.Attachment.zip

nal seed lexicon of BNC+GT may be safely replaced by an automatically induced inexpensive seed lexicon (as in HYBWE with BNC+HYB+SYM/ASYM). The best performing models are again BNC+HYB+SYM and HFQ+HYB+SYM. The comparison of ASYM and SYM lexicon variants further suggests that filtering translation pairs using the symmetry constraint again leads to consistent improvements, but stricter selection criteria with higher thresholds do not lead to significant performance boosts, and may even hurt the performance (see the results for NL-EN). Various HYBWE variants significantly improve over baseline BWE models (Types 1-4), also outperforming previous best SWTC results.

5 Conclusions and Future Work

We presented a detailed analysis of the importance and properties of seed bilingual lexicons in learning bilingual word embeddings (BWEs) which are valuable for many cross-lingual/multilingual NLP tasks. On the basis of the analysis, we proposed a simple yet effective hybrid bilingual word embedding model called HYBWE. It learns the mapping between two monolingual embedding spaces using only highly reliable symmetric translation pairs from an inexpensive seed document-level embedding space. The results in the tasks of (1) bilingual lexicon learning and (2) suggesting word translations in context demonstrate that – due to its careful selection of reliable translation pairs for seed lexicons – HYBWE outperforms benchmarking BWE induction models, all of which use more expensive bilingual signals for training.

In future work, we plan to investigate other methods for seed pairs selection, settings with scarce resources (Agić et al., 2015; Zhang et al., 2016), other context types inspired by recent work in the monolingual settings (Levy and Goldberg, 2014a; Melamud et al., 2016), as well as model adaptations that can work with multi-word expressions. Encouraged by the excellent results, we also plan to test the portability of the approach to more language pairs, and other tasks and applications.

Acknowledgments

This work is supported by ERC Consolidator Grant LEXICAL: Lexical Acquisition Across Languages (no 648909). The authors are grateful to Roi Reichart and the anonymous reviewers for their helpful comments and suggestions.

References

Željko Agić, Dirk Hovy, and Anders Søgaard. 2015. If all you have is a bit of the Bible: Learning POS taggers for truly low-resource languages. In *ACL*, pages 268–272.

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. In *CoNLL*, pages 183–192.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A. Smith. 2016. Massively multilingual word embeddings. *CoRR*, abs/1602.01925.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *IJCAI*, pages 1764–1769.

Sarath A.P. Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *NIPS*, pages 1853–1861.

Danqi Chen and Christopher D. Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, fast cross-lingual word embeddings. In *EMNLP*, pages 1109–1113.

Georgiana Dinu, Angeliki Lazaridou, and Marco Baroni. 2015. Improving zero-shot learning by mitigating the hubness problem. In *ICLR Workshop Papers*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *EACL*, pages 462–471.

Éric Gaussier, Jean-Michel Renders, Irina Matveeva, Cyril Goutte, and Hervé Déjean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *ACL*, pages 526–533.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast bilingual distributed representations without word alignments. In *ICML*, pages 748–756.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *ACL*, pages 1234–1244.

Karl Moritz Hermann and Phil Blunsom. 2014a. Multilingual distributed representations without word alignment. In *ICLR*.

Karl Moritz Hermann and Phil Blunsom. 2014b. Multilingual models for compositional distributed semantics. In *ACL*, pages 58–68.

Jon Holmlund, Magnus Sahlgren, and Jussi Karlgren. 2005. Creating bilingual lexica using reference wordlists for alignment of monolingual semantic vector spaces. In *NODALIDA*, pages 71–77.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*, pages 873–882.

Anders Johannsen, Héctor Martínez Alonso, and Anders Søgaard. 2015. Any-language frame-semantic parsing. In *EMNLP*, pages 2062–2066.

Adam Kilgarriff. 1997. Putting frequencies in the dictionary. *International Journal of Lexicography*, 10(2):135–155.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *COLING*, pages 1459–1474.

Tomáš Kočiský, Karl Moritz Hermann, and Phil Blunsom. 2014. Learning bilingual word representations by marginalizing alignments. In *ACL*, pages 224–229.

Angeliki Lazaridou, Georgiana Dinu, and Marco Baroni. 2015. Hubness and pollution: Delving into cross-space mapping for zero-shot learning. In *ACL*, pages 270–280.

Ira Leviant and Roi Reichart. 2015. Judgment language matters: Multilingual vector space models for judgment language aware lexical semantics. *CoRR*, abs/1508.00106.

Omer Levy and Yoav Goldberg. 2014a. Dependency-based word embeddings. In *ACL*, pages 302–308.

Omer Levy and Yoav Goldberg. 2014b. Neural word embedding as implicit matrix factorization. In *NIPS*, pages 2177–2185.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the ACL*, 3:211–225.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Oren Melamud, Omer Levy, and Ido Dagan. 2015. A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 1–7.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *NAACL-HLT*.

Rada Mihalcea, Ravi Sinha, and Diana McCarthy. 2010. SemEval-2010 task 2: Cross-lingual lexical substitution. In *SEMEVAL*, pages 9–14.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Dmitrijs Milajevs, Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Matthew Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *EMNLP*, pages 708–719.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

Bhaskar Mitra, Eric T. Nalisnick, Nick Craswell, and Rich Caruana. 2016. A dual embedding space model for document ranking. *CoRR*, abs/1602.01137.

Yves Peirsman and Sebastian Padó. 2010. Cross-lingual induction of selectional preferences with bilingual vector spaces. In *NAACL*, pages 921–929.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning cross-lingual word embeddings via matrix co-factorization. In *ACL*, pages 567–572.

David A. Smith and Jason Eisner. 2007. Bootstrapping feature-rich dependency parsers with entropic priors. In *EMNLP-CoNLL*, pages 667–677.

Hubert Soyer, Pontus Stenetorp, and Akiko Aizawa. 2015. Leveraging monolingual data for crosslingual compositional word representations. In *ICLR*.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *LREC*, pages 2214–2218.

Chen-Tse Tsai and Dan Roth. 2016. Cross-lingual wikification using multilingual embeddings. In *NAACL-HLT*.

Kewei Tu and Vasant Honavar. 2012. Unambiguity regularization for unsupervised learning of probabilistic grammars. In *EMNLP-CoNLL*, pages 1324–1334.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: vector space models of semantics. *Journal of Artifical Intelligence Research*, 37(1):141–188.

Shyam Upadhyay, Manaal Faruqui, Chris Dyer, and Dan Roth. 2016. Cross-lingual models of word embeddings: An empirical comparison. In *ACL*.

Ivan Vulić and Marie-Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *EMNLP*, pages 1613–1624.

Ivan Vulić and Marie-Francine Moens. 2014. Probabilistic models of cross-lingual semantic similarity in context based on latent cross-lingual concepts induced from comparable data. In *EMNLP*, pages 349–362.

Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *SIGIR*, pages 363–372.

Ivan Vulić and Marie-Francine Moens. 2016. Bilingual distributed word representations from document-aligned comparable data. *Journal of Artificial Intelligence Research*, 55:953–994.

Ivan Vulić, Douwe Kiela, Stephen Clark, and Marie-Francine Moens. 2016. Multi-modal representations for improved bilingual lexicon learning. In *ACL*.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. WSABIE: scaling up to large vocabulary image annotation. In *IJCAI*, pages 2764–2770.

Yuan Zhang, David Gaddy, Regina Barzilay, and Tommi Jaakkola. 2016. Ten pairs to tag - Multilingual POS tagging via coarse mapping between embeddings. In *NAACL-HLT*.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.

Liberal Event Extraction and Event Schema Induction

Lifu Huang[1], **Taylor Cassidy**[2], **Xiaocheng Feng**[3],
Heng Ji[1], **Clare R. Voss**[2], **Jiawei Han**[4], **Avirup Sil**[5]
[1] Rensselaer Polytechnic Institute, [2] US Army Research Lab,
[3] Harbin Institute of Technology, [4] Univerisity of Illinois at Urbana-Champaign,
[5] IBM T.J. Watson Research Center
[1]{huangl7,jih}@rpi.edu,
[2]{taylor.cassidy.civ,clare.r.voss.civ}@rpi.edu,
[3]xcfeng@ir.hit.edu.cn, [4]hanj@illinois.edu, [5]avi@us.ibm.com

Abstract

We propose a brand new "Liberal" Event Extraction paradigm to extract events and discover event schemas from any input corpus simultaneously. We incorporate symbolic (e.g., Abstract Meaning Representation) and distributional semantics to detect and represent event structures and adopt a joint typing framework to simultaneously extract event types and argument roles and discover an event schema. Experiments on general and specific domains demonstrate that this framework can construct high-quality schemas with many event and argument role types, covering a high proportion of event types and argument roles in manually defined schemas. We show that extraction performance using discovered schemas is comparable to supervised models trained from a large amount of data labeled according to predefined event types. The extraction quality of new event types is also promising.

1 Introduction

Event extraction aims at identifying and typing trigger words and participants (arguments). It remains a challenging and costly task. The first question is what to extract? The TIPSTER (Onyshkevych et al., 1993), MUC (Grishman and Sundheim, 1996), CoNLL (Tjong et al., 2003; Pradhan et al., 2011), ACE [1] and TAC-KBP (Ji and Grishman, 2011) programs found that it was feasible to manually define an event schema based on the needs of potential users. An ACE event schema example is shown in Figure 1. This process is very expensive because consumers and expert linguists need to examine a lot of data before specifying the types of events and argument roles and writing detailed annotation guidelines for each type in the schema. Manually-defined event schemas often provide low coverage and fail to generalize to new domains. For example, none of the aforementioned programs include "*donation*" and "*evacuation*" in their schema in spite of their potential relevance to users.

In this paper we propose *Liberal Event Extraction*, a new paradigm to take humans out of the loop and enable systems to extract events in a more liberal fashion. It automatically discovers a complete event schema, customized for a specific input corpus. Figure 1 compares the ACE event extraction paradigm and our proposed Liberal event extraction paradigm.

We use the following examples to explain and motivate our approach, where event triggers are in bold and arguments are in italics and underlined:

E1. *Two Soldiers* were **killed** and *one* **injured** in the close-quarters **fighting** in *Kut*.

E2. *Bill Bennet*'s glam *gambling* **loss** changed my opinion.

E3. Gen. Vincent Brooks announced the **capture** of *Barzan Ibrahim Hasan al-Tikriti*, telling reporters he was an adviser to Saddam.

E4. This was the *Italian ship* that was **captured** by *Palestinian* terrorists back in 1985.

E5. *Ayman Sabawi Ibrahim* was **arrested** in *Tikrit* and was **sentenced** to life in *prison*.

We seek to cluster the event triggers and event arguments so that each cluster represents a type. We rely on distributional similarity for our clustering distance metric. The distributional hypothesis (Harris, 1954) states that words often occurring in similar contexts tend to have similar meanings. We formulate the following distributional

[1] http://www.itl.nist.gov/iad/mig/tests/ace/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 258–268,
Berlin, Germany, August 7-12, 2016.

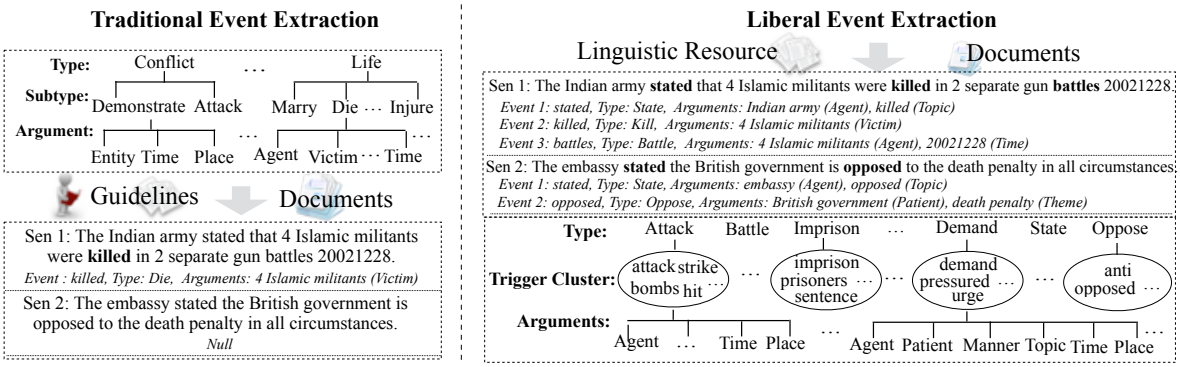


Figure 1: Comparison between ACE Event Extraction and Liberal Event Extraction.

hypotheses specifically for event extraction, and develop our approach accordingly.

Hypothesis 1: *Event triggers that occur in similar contexts and share the same sense tend to have similar types.*

Following the distributional hypothesis, when we simply learn general word embeddings from a large corpus for each word, we obtain similar words like those shown in Table 1. We can see similar words, such as those centered around "*injure*" and "*fight*", are converging to similar types. However, for words with multiple senses such as "*fire*" (shooting or employment termination), similar words may indicate multiple event types. Thus, we propose to apply Word Sense Disambiguation (WSD) and learn a distinct embedding for each sense (Section 2.3).

injure	Score	fight	Score	fire	Score
injures	0.602	fighting	0.792	fires	0.686
hurt	0.593	fights	0.762	aim	0.683
harm	0.592	battle	0.702	enemy	0.601
maim	0.571	fought	0.636	grenades	0.597
injuring	0.561	Fight	0.610	bombs	0.585
endanger	0.543	battles	0.590	blast	0.566
dislocate	0.529	Fighting	0.588	burning	0.562
kill	0.527	bout	0.570	smoke	0.558

Table 1: Top-8 Most Similar Words (in 3 Clusters)

Hypothesis 2: *Beyond the lexical semantics of a particular event trigger, its type is also dependent on its arguments and their roles, as well as other words contextually connected to the trigger.*

For example, in *E4*, the fact that the patient role is a vehicle ("*Italian ship*"), and not a person (as in *E3* and *E5*), suggests that the event trigger "*captured*" has type "*Transfer-Ownership*" as opposed to "*Arrest*". In *E2*, we know the "*loss*" event occurs in a *gambling* scenario, so we can determine its type as loss of money, not loss of life.

We therefore propose to enrich each trigger's representation by incorporating the distributional representations of various words in the trigger's context. Not all context words are relevant to event trigger type prediction, while those that are vary in their predictive value. We propose to use semantic relations, derived from a meaning representation for the text, to carefully select arguments and other words in an event trigger's context. These words are then incorporated into a "global" event structure for a trigger mention. We rely on semantic relations to (1) specify how the distributional semantics of relevant context words contribute to the overall event structure representation; (2) determine the order in which distributional semantics of relevant context words are incorporated into the event structure (Section 2.4).

2 Approach

2.1 Overview

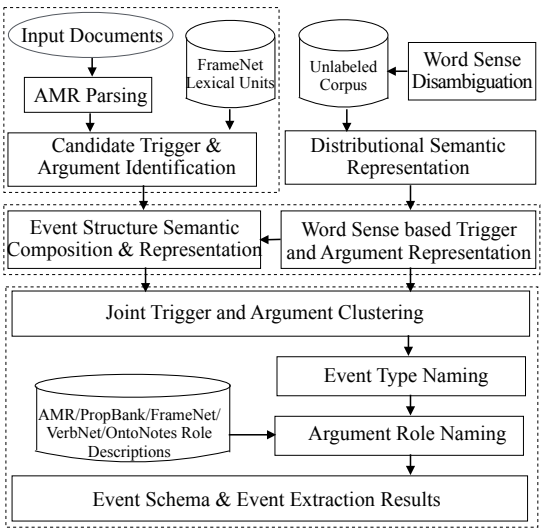


Figure 2: Liberal Event Extraction Overview.

Figure 2 illustrates the overall framework of

Liberal Event Extraction. Given a set of input documents, we first extract semantic relations, apply WSD and learn word sense embeddings. Next, we identify candidate triggers and arguments.

For each event trigger, we apply a series of compositional functions to generate that trigger's event structure representation. Each function is specific to a semantic relation, and operates over vectors in the embedding space. Argument representations are generated as a by-product.

Trigger and argument representations are then passed to a joint constraint clustering framework. Finally, we name each cluster of triggers, and name each trigger's arguments using mappings between the meaning representation and semantic role descriptions in FrameNet, VerbNet (Kipper et al., 2008) and Propbank (Palmer et al., 2005).

We compare settings in which semantic relations connecting triggers to context words are derived from three meaning representations: Abstract Meaning Representation (AMR) (Banarescu et al., 2013), Stanford Typed Dependencies (Marie-Catherine et al., 2006), and FrameNet (Baker and Sato, 2003). We derive semantic relations automatically for these three representations using CAMR (Wang et al., 2015a), Stanford's dependency parser (Manning, 2003), and SEMAFOR (Das et al., 2014), respectively.

2.2 Candidate Trigger and Argument Identification

Given a sentence, we consider all noun and verb concepts that are assigned an OntoNotes (Hovy et al., 2006) sense by WSD as candidate event triggers. Any remaining concepts that match both a verbal and a nominal lexical unit in the FrameNet corpus are considered candidate event triggers as well. This mainly helps to identify more nominal triggers like *"pickpocket"* and *"sin"*.[2]

For each candidate event trigger, we consider as candidate arguments all concepts for which one of a manually-selected set of semantic relations holds between it and the event trigger. For the setting in which AMR serves as our meaning representation, we selected a subset of all AMR relations that specify event arguments, as shown in Table 2. Note that some AMR relations generally do not specify event arguments, e.g. "mode", which can indicate sentence illocutionary force, or "snt" which is used to combine multiple sentences into one AMR graph.[3] When FrameNet is the meaning representation we allow all frame relations to identify arguments. For dependencies, we manually mapped dependency relations to AMR relations and use Table 2.

Categories	Relations
Core roles	ARG0, ARG1, ARG2, ARG3, ARG4
Non-core roles	mod, location, poss, manner, topic, medium, instrument, duration, prep-X
Temporal	year, duration, decade, weekday, time
Spatial	destination, path, location

Table 2: Event-Related AMR Relations.

In E1, for example, "*killed*", "*injured*" and "*fighting*" are identified as candidate triggers, and three concept sets are identified as candidate arguments using AMR relations: "{*Two Soldiers, very large missile*}", "{*one, Kut*}" and "{*Two Soldiers, Kut*}", as shown in Figure 3.

2.3 Trigger Sense and Argument Representation

Based on Hypothesis 1, we learn sense-based embeddings from a large data set, using the Continuous Skip-gram model (Mikolov et al., 2013). Specifically, we first apply WSD to link each word to its sense in WordNet using a state-of-the-art tool (Zhong and Ng, 2010), and map WordNet sense output to OntoNotes senses. [4] We map each trigger candidate to its OntoNotes sense and learn a distinct embedding for each sense. We use general lexical embeddings for arguments.

2.4 Event Structure Composition and Representation

Based on Hypothesis 2, we aim to exploit linguistic knowledge to incorporate inter-dependencies between event and argument role types into our event structure representation. Many meaning representations could provide such information to some degree. We illustrate our method for building event structures using semantic relations from meaning representations using AMR. In Section 3.4 we compare results using Stanford Typed Dependencies and FrameNet in place of AMR.

Let's take *E2* as an example. Based on AMR annotation and Table 2, we extract semantically re-

[2]For consistency, we use the same trigger identification procedure regardless of which meaning representation is used to derive semantic relations.

[3]For relation details, see https://github.com/amrisi/amr-guidelines/blob/master/amr.md

[4]WordNet-OntoNotes mapping from https://catalog.ldc.upenn.edu/LDC2011T03

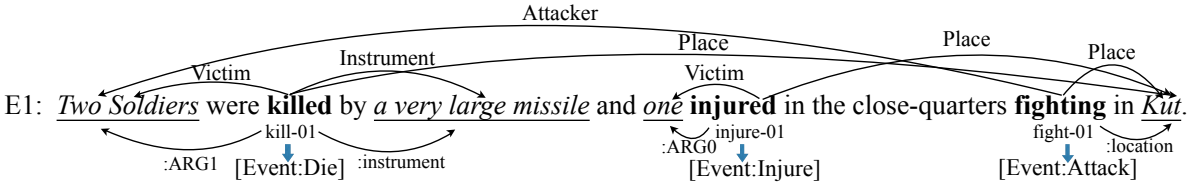


Figure 3: Event Trigger and Argument Annotations and AMR Parsing Results of *E1*.

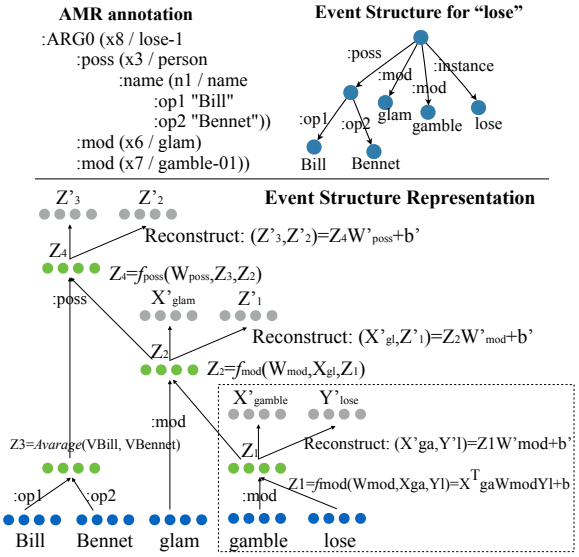


Figure 4: Partial AMR and Event Structure for *E2*.

lated words for the event trigger with sense "*lose-1*" and construct the event structure for the whole event, as shown in Figure 4.

We design a Tensor based Recursive Auto-Encoder (TRAE) (Socher et al., 2011) framework to utilize a tensor based composition function for each of a subset of the AMR semantic relations and compose the event structure representation based on multiple functional applications. This subset was manually selected by the authors as the set of relations that link a trigger to concepts that help to determine its type. Similarly, we selected a subset of dependency and FrameNet relations using the same criteria for experiments using those meaning representations.

Figure 4 shows an instance of a TRAE applied to an event structure to generate its representation. For each semantic relation type r, such as "*:mod*", we define the output of a tensor product Z via the following vectorized notation:

$$Z = f_{mod}(X, Y, W_r^{[1:d]}, b) = [X;Y]^T W_r^{[1:d]} [X;Y] + b$$

where $W_{mod} \in \mathbb{R}^{2d \cdot 2d \cdot d}$ is a 3-order tensor, and $X, Y \in \mathbb{R}^d$ are two input word vectors. $b \in \mathbb{R}^d$ is the bias term. $[X;Y]$ denotes the concatenation of two vectors X and Y. Each slice of the tensor acts as a coefficient matrix for one entry Z_i in Z:

$$Z_i = f_{mod}(X, Y, W_r^{[i]}, b) = [X;Y]^T W_r^{[i]} [X;Y] + b_i$$

We use the statistical mean to compose the words connected by ":op" relations (e.g. "*Bill*" and "*Bennet*" in Figure 4).

After composing the vectors of X and Y, we apply an element-wise sigmoid activation function to the composed vector and generate the hidden layer representations Z. One way to optimize Z is to try to reconstruct the vectors X and Y by generating X' and Y' from Z, and minimizing the reconstruction errors between the input $V_I = [X, Y]$ and output layers $V_O = [X', Y']$. The error is computed based on Euclidean distance function:

$$E(V_I, V_O) = \frac{1}{2} ||V_I - V_O||^2$$

For each pair of words X and Y, the reconstruction error back-propagates from its output layer to input layer through parameters $\Theta_r = (W_r', b_r', W_r, b_r)$. Let δ_O be the residual error of the output layer, and δ_H be the error of the hidden layer:

$$\delta_O = -(V_I - V_O) \cdot f'_{\text{sigmoid}}(V_H^O)$$

$$\delta_H = (\sum_{k=1}^{d} \delta_O^k \cdot (W_r'^k + (W_r'^k)^T) \cdot V_H^O) \cdot f'_{\text{sigmoid}}(V_H^I)$$

where V_H^I and V_H^O denote the input and output of the hidden layer, and $V_H^O = Z$. $W_r'^k$ is the k^{th} slice of tensor W_r'.

To minimize the reconstruction errors, we utilize gradient descent to iteratively update parameters Θ_r:

$$\frac{\partial E(\Theta_r)}{\partial W_r'^k} = \delta_O^k \cdot (V_H^O)^T \cdot V_H^O$$

$$\frac{\partial E(\Theta_r)}{\partial b_r'} = -(V_I - V_O) \cdot f'_{\text{sigmoid}}(V_H^O)$$

$$\frac{\partial E(\Theta_r)}{\partial W_r^k} = \delta_H^k \cdot (V_I)^T \cdot V_I$$

$$\frac{\partial E(\Theta_r)}{\partial b_r} = (\sum_{k=1}^{d} \delta_O^k \cdot (W_r'^k + (W_r'^k)^T) \cdot V_H^O) \cdot f'_{\text{sigmoid}}(V_H^I)$$

After computing the composition vector of Z_1 based on X and Y, for the next layer, it composes Z_1 and another new word vector such as

X_{gl}. For each type of relation r, we randomly sample 2,000 pairs to train optimized parameters Θ_r. For each event structure tree, we iteratively repeat the same steps for each layer. For multiple arguments at each layer, we compose them in the order of their distance to the trigger: the closest argument is composed first.

2.5 Joint Trigger and Argument Clustering

Based on the representation vectors generated above, we compute the similarity between each pair of triggers and arguments, and cluster them into types. Recall that a trigger's arguments are identified as in section 2.2. We observe that, for two triggers t_1 and t_2, if their arguments have the same type and role, then they are more likely to belong to the same type, and vice versa. Therefore we introduce a constraint function f, to enforce inter-dependent triggers and arguments to have coherent types:

$$f(\mathcal{P}_1, \mathcal{P}_2) = \log(1 + \frac{|\mathcal{L}_1 \cap \mathcal{L}_2|}{|\mathcal{L}_1 \cup \mathcal{L}_2|})$$

where $\mathcal{P}_1$ and $\mathcal{P}_2$ are triggers. Elements of $\mathcal{L}_i$ are pairs of the form $(r, \text{id}(a))$, where $\text{id}(a)$ is the cluster ID for argument a that stands in relation r to $\mathcal{P}_i$. For example, let $\mathcal{P}_1$ and $\mathcal{P}_2$ be triggers "*capture*" and "*arrested*" (c.f. Figure 5). If *Barzan Ibrahim Hasan al-Tikriti* and *Ayman Sabawi Ibrahim* share the same cluster ID, the pair $(arg1, \text{id(Barzan Ibrahim Hasan al-Tikriti)})$ will be a member of $\mathcal{L}_1 \cap \mathcal{L}_2$. This argument overlap is evidence that "*capture*" and "*arrested*" have the same type. We define f where $\mathcal{P}_i$ are arguments, and elements $\mathcal{L}_i$ are defined analogously to above.

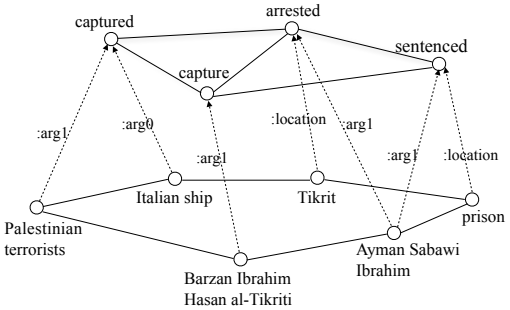


Figure 5: Joint Constraint Clustering for E3,4,5.

Given a trigger set T and their corresponding argument set A, we compute the similarity between two triggers t_1 and t_2 and two arguments a_1 and a_2 by:

$$sim(t_1, t_2) = \lambda \cdot sim_{\cos}(E_g^{t_1}, E_g^{t_2}) + (1-\lambda) \cdot \frac{\Sigma_{r \in R_{t_1} \cap R_{t_2}} sim_{\cos}(E_r^{t_1}, E_r^{t_2})}{|R_{t_1} \cap R_{t_2}|} + f(t_1, t_2)$$

$$sim(a_1, a_2) = sim_{\cos}(E_g^{a_1}, E_g^{a_2}) + f(a_1, a_2)$$

where E_g^t represents the trigger sense vector and E_g^a is the argument vector. R_t is the AMR relation set in the event structure of t, and E_r^t denotes the vector resulting from the last application of the compositional function corresponding to the semantic relation r for trigger t. λ is a regularization parameter that controls the trade-off between these two types of representations. In our experiment $\lambda = 0.6$.

We design a joint constraint clustering approach, which iteratively produces new clustering results based on the above constraints. To find a global optimum, which corresponds to an approximately optimal partition of the trigger set into K clusters $\mathcal{C}^T = \{\mathcal{C}_1^T, \mathcal{C}_2^T, ..., \mathcal{C}_K^T\}$, and a partition of the argument set into M clusters $\mathcal{C}^A = \{\mathcal{C}_1^A, \mathcal{C}_2^A, ..., \mathcal{C}_M^A\}$, we minimize the agreement across clusters and the disagreement within clusters:

$$\arg \min_{K_T, K_A, \lambda} O = (D_{\text{inter}}^T + D_{\text{intra}}^T) + (D_{\text{inter}}^A + D_{\text{intra}}^A)$$

$$D_{\text{inter}}^{\mathcal{P}} = \sum_{i \neq j = 1}^{K} \sum_{u \in \mathcal{C}_i^{\mathcal{P}}, v \in \mathcal{C}_j^{\mathcal{P}}} sim(\mathcal{P}_u, \mathcal{P}_v)$$

$$D_{\text{intra}}^{\mathcal{P}} = \sum_{i=1}^{K} \sum_{u,v \in \mathcal{C}_i^{\mathcal{P}}} (1 - sim(\mathcal{P}_u, \mathcal{P}_v))$$

We incorporate the Spectral Clustering algorithm (Luxburg, 2007) into joint constraint clustering process to get the final optimized clustering results. The detailed algorithm is summarized in Algorithm 1.

2.6 Event Type and Argument Role Naming

For each trigger cluster, we utilize the trigger which is nearest to the centroid of the cluster as the event type name. For a given event trigger, we assign a role name to each of its arguments (identified as in section 2.2). This process depends on which meaning representation was used to select the arguments.

For AMR, we first map the event trigger's OntoNotes sense to PropBank, VerbNet, and FrameNet. We assign each argument a role name as follows. We map AMR core roles (e.g. *":ARG0", "ARG1"*) to FrameNet if possible, otherwise to VerbNet if possible, and finally to PropBank roles if a mapping to VerbNet is not available.[5]. Nearly 5% of AMR core roles can

[5]OntoNotes 5.0 provides a mapping; https://catalog.ldc.upenn.edu/LDC2013T19

Algorithm 1 Joint Constraint Clustering Algorithm

Input: Trigger set T, argument set A, their lexical embedding E_g^T, E_g^A, event structure representation E_R^T, and the minimal (K_T^{min}, K_A^{min}) and maximal (K_T^{max}, K_A^{max}) number of clusters for triggers and arguments;
Output: The optimal clustering results: $\mathcal{C}^T$ and $\mathcal{C}^A$;

- $O_{min} = \infty, \mathcal{C}^T = \varnothing, \mathcal{C}^A = \varnothing$
- For $K_T = K_T^{min}$ to $K_T = K_T^{max}$, $K_A = K_A^{min}$ to $K_A = K_A^{max}$
  - Clustering with Spectral Clustering Algorithm:
  - $\mathcal{C}_{curr}^T = spectral(T, E_g^T, E_R^T, K_T)$
  - $\mathcal{C}_{curr}^A = spectral(A, E_g^A, K_A)$
  - $O_{curr} = O(\mathcal{C}_{curr}^T, \mathcal{C}_{curr}^A)$
  - if $O_{curr} < O_{min}$
    - $O_{min} = O_{curr}, \mathcal{C}^T = \mathcal{C}_{curr}^T, \mathcal{C}^A = \mathcal{C}_{curr}^A$
  - while iterate time ≤ 10
    - $\mathcal{C}_{curr}^T = spectral(T, E_g^T, E_R^T, K_T, \mathcal{C}_{curr}^A)$
    - $\mathcal{C}_{curr}^A = spectral(A, E_g^A, K_A, \mathcal{C}_{curr}^T)$
    - $O_{curr} = O(\mathcal{C}_{curr}^T, \mathcal{C}_{curr}^A)$
    - if $O_{curr} < O_{min}$
      - O_{min}=$O_{curr}, \mathcal{C}^T = \mathcal{C}_{curr}^T, \mathcal{C}^A = \mathcal{C}_{curr}^A$
- return $O_{min}, \mathcal{C}^T, \mathcal{C}^A$;

be mapped to FrameNet roles and 55% can be mapped to VerbNet roles, and the remaining can be mapped to PropBank. Table 3 shows some mapping examples. We map non-core roles from AMR to FrameNet, as shown in Table 4.

When Stanford Typed Dependencies are used for meaning representation we construct a manual mapping AMR relations and use the above procedure. When FrameNet is used for meaning representation we simply keep the FrameNet role name for argument role naming.

Concept	AMR Core Role	FrameNet Role	VerbNet Role	PropBank Description
fire.1	ARG0	Agent	Agent	Shooter
fire.1	ARG1	Projectile	Theme	Gun/projectile
extrude.1	ARG0		Agent	Extruder, agent
extrude.1	ARG1		Theme	Entity extruded
extrude.1	ARG2		Source	Extruded from
blood.1	ARG0			Agent
blood.1	ARG1			Theme, one bled

Table 3: Core Role Mapping Examples.

3 Evaluation

3.1 Data

We used the August 11, 2014 English Wikipedia dump to learn trigger sense and argument embeddings. For evaluation we choose a subset of ERE (Entity Relation Event) corpus (50 documents) which has perfect AMR annotations so we can compare the impact of perfect AMR and system generated AMR. To compare with state-of-the-art event extraction on Automatic Content Extraction (ACE2005) data, we follow the same evaluation setting in previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Hong et al., 2011) and use 40 newswire documents as our test set.

AMR None-Core Role	FrameNet Role
topic	Topic
instrument	Instrument
manner	Manner
poss	Possessor
prep-for, prep-to, prep-on-behalf	Purpose
time, decade, year, weekday, duration	Time
mod, cause, prep-as	Explanation
prep-by, medium, path	Means
location, destination, prep-in	Place

Table 4: None-Core Role Mapping.

3.2 Schema Discovery

Figure 6 shows some examples as part of the event schema discovered from the ERE data set. Each cluster denotes an event type, with a set of event mentions and sentences. Each event mention is also associated with some arguments and their roles. The event and argument role annotations for sample sentences may serve as an example-based corpus-customized "annotation guideline" for event extraction.

Table 5 compares the coverage of event schema discovered by our approach, using AMR as meaning representation, with the predefined ACE and ERE event schemas. Besides the types defined in ACE and ERE, this approach discovers many new event types such as **Build** and **Threaten** as displayed in Figure 6. Our approach can also discover new argument roles for a given event type. For example, for Attack events, besides five types of existing arguments (Attacker, Target, Instrument, Time, and Place) defined in ACE, we also discover a new type of argument **Purpose**. For example, in "*The Dutch government, facing strong public anti-war pressure, said it would not commit fighting forces to the war against Iraq but added it supported the military* ***campaign*** *to* ***disarm Saddam***.", "*disarm Saddam*" is identified as the **Purpose** for the **Attack** event triggered by "*campaign*". Note that while FrameNet specifies Purpose as an argument role for the Attack, such information specific to Attack is not part of AMR.

Event Type: Transport

S1: The court official stated that on 18 March 2008 Luong stated to judges that she was hired by an unidentified *man* to **ship** the *heroin* to *Australia* in exchange for 15000 U.S. dollars.
***Event**: ship,*
***Arguments**: man(Agent), Austrialia(Destination),heroin(Theme)*

S2: State media didn't identify the 2 convicts hanged in Zahedan but stated that *they* had been found guilty of **transporting** 5.25 kilograms of *heroin*.
***Event**: transporting, **Arguments**: they(Agent), heroin(Theme)*

Event Type: Die

S1: Police in the strict communist country discovered his methamphetamine manufacturing plant disguised as a soap factory and sentenced *him* to **death** in *1997*.
***Event**: death, **Arguments**: him(Theme), 1997(Time)*

S2: A newspaper report on January 1, 2008 that *Iran* **hanged** two convicted *drug traffickers* in the *south-eastern city of Zahedan*.
***Event**: hanged, **Arguments**: Iran(Agent), drug traffickers(Theme), southeastern city of Zahedan(Place)*

Event Type: Build

S1: The **construction** of the *facility* started in *790000*, but stopped after the 910000 Soviet collapse when Tajikistan slid into a 5 year civil war that undermined its economy.
***Event**:construction,*
***Arguments**: facility(Product), 790000(Time)*

S2: The closed *Soviet-era military facility* was **fou-nded** in *570000* and collects and analyzes all information gathered from Russia's military spy satellites.
***Event**: founded,*
***Arguments**: Soviet-era facility(Product), 570000(Time)*

Event Type: Threaten

S1: *Colombian Government* was **alarmed** because uranium is the primary basis for generating weapons of mass destruction.
***Event**:alarmed, **Arguments**:Columbian Government(Experiencer)*

S2: Cluster bomblets have been criticized by human rights groups because they kill indiscriminately and because unexploded *ordinance* poses a **threat** to *civilians* similar to that of land mines.
***Event**:threat,*
***Arguments**:ordinance(Cause), civilian(Experiencer)*

Figure 6: Example Output of the Event Schema.

Data	ACE			ERE				
	Human	*SystemAMR*	*Overlap*	*Human*	*PerfectAMR*	*Overlap*	*SystemAMR*	*Overlap*
# of Events	440	2,395	331	580	3,765	517	2,498	477
# of Event Types	33	134	N/A	26	137	N/A	120	N/A
# of Arguments	883	4,361	587	1,231	6,195	919	4,288	801

Table 5: Schema Coverage Comparison on ACE and ERE.

3.3 Event Extraction for All Types

To evaluate the performance of the whole event schema, we randomly sample 100 sentences from ERE data set and ask two linguistic experts to fully annotate the events and arguments. As a starting point, annotators were given output from our Schema Discovery using gold standard AMR. For each sentence, they saw event triggers and corresponding arguments. Their job was to correct this output by marking incorrectly identified events and arguments, and adding missing events and arguments. The inter-annotator agreement is 83% for triggers and 79% for arguments.

To evaluate trigger and argument identification, we automatically compare this gold standard with system output (see Table 6). To evaluate trigger and argument typing, annotators manually checked system output and assessed whether the type name was reasonable (see Table 6). Note that automatic comparison between system and gold standard output is not appropriate for typing; for a given cluster, there is no definitive "best" name.

We found that most event triggers not recovered by our system are multi-word expressions such as "*took office*" or adverbs such as "*previously*" and "*formerly*". For argument identification, our approach fails to identify some arguments that require world knowledge to extract. For example, in "*Anti-corruption judge **Saul Pena** stated Montesinos has admitted to the abuse of authority **charge***", "*Saul Pena*" is not identified as a **Adjudicator** argument of event "*charge*" because it has no direct semantic relations with the event trigger.

3.4 Impact of Semantic Information and Meaning Representations

Table 7 assesses the impact of various types of semantic information, and also compares the effectiveness of each type of meaning representation for the typing task only. We note that F-measure drops 14.4 points if only WSD based embeddings are not used. In addition, AMR relations specifying both core and non-core roles are informative for learning distinct compositional operators. To compare typing results across meaning representations, we use triggers identified by both the AMR and FrameNet parsers. Using Stanford Typed Dependencies, relations are likely too coarse-grained or lack sufficient semantic information. Thus, our approach cannot leverage the inter-dependency between event trigger type and argument role to achieve pure trigger clusters. Compared with dependency relations, the fine-grained AMR semantic relations such as *:location*, *:manner*, *:topic*, *:instrument* appear to be more informative to infer the argument roles. For example, in sentence "*Approximately 25 kilometers southwest of Sringar 2 militants were killed in a second gun battle.*", "*gun*" is identified as an **Instrument** for "*battle*" event based on the AMR relation :instrument. In contrast, dependency parsing identifies "*gun*" as a

Method	Trigger Identification (%)			Trigger Typing (%)			Arg Identification (%)			Arg Typing (%)		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Perfect AMR	87.0	98.7	92.5	70.0	79.5	74.5	94.0	83.7	88.6	72.4	64.4	68.2
System AMR	93.0	67.2	78.0	69.8	50.5	58.6	95.7	59.6	73.4	68.9	42.9	52.9

Table 6: Overall Performance of Liberal Event Extraction on ERE data for All Event Types.

Method	Trigger F_1 (%)			Arg F_1 (%)		
	P	R	F_1	P	R	F_1
Perfect AMR	70	79.5	74.5	72.4	64.4	68.2
w/o Structure Representation	52.8	59.4	55.9	52.1	48.0	50.0
w/o WSD based embeddings	62.8	57.4	60.1	61.9	50.3	55.5
w/o None-Core Roles	61.5	72.2	66.5	61.3	58.0	59.6
w/o Core Roles	57.3	49.7	53.2	63.6	49.5	55.7
System AMR	69.8	50.5	58.6	68.9	42.9	52.9
Replace AMR with Dependency Parsing	45.9	61.9	52.7	63.9	18.2	28.4
Replace AMR with FrameNet Parsing	43.1	57.1	49.2	78.1	7.1	13.0

Table 7: Impact of semantic information and representations on typing for ERE data.

compound modifier of "*battle*". Note that we used a static mapping to map dependency relations to AMR relations (see section 2.6), whereas ideally this mapping would be context-dependent. Creating a context-dependent mapping would constitute significant steps toward building an AMR parser.

Using FrameNet results in low recall for argument typing. SEMAFOR's output often does not identify all the arguments identified by our annotators. Many triggers are associated with zero or one argument, thus there is not enough data to learn the event structure representation. In addition, most of the arguments from identified by SEMAFOR are long phrases. Because no internal structure is assigned, we simply average all single token's vectors to represent the phrase. However, the high precision may be due to the fact that FrameNet relations are designed to specify semantic roles.

3.5 Event Extraction for ACE/ERE Types

We manually select the event triggers in the ACE and ERE evaluation sets discovered by our AMR-based approaches that are ACE/ERE events based on their annotation guidelines. If a trigger doesn't already have a gold standard ACE/ERE annotation we provide one. For each such event we use core roles and Instrument/Possessor/Time/Place relations to detect arguments. Each trigger and argument role type is assessed manually if an ACE/ERE annotation does not exist. We evaluate our approach for trigger and argument typing by comparing system output to manual annotation, considering synonymous labels to be equivalent (e.g., our approach's *kill* type ACE's *die*). We compare our approach with the following state-of-the-art supervised methods which are trained from 529 ACE documents or 336 ERE documents:

- DMCNN: A dynamic multi-pooling convolutional neural network based on distributed word representations (Chen et al., 2015).
- Joint: A structured perceptron model based on symbolic semantic features (Li et al., 2013).
- LSTM: A long short-term memory neural network (Hochreiter and Schmidhuber, 1997) based on distributed semantic features.

Table 8 shows the results. On ACE events, both DMCNN and Joint methods outperform our approach for trigger and argument extraction. However, when moving to ERE event schema, although re-trained based on ERE labeled data, their performance still degrades significantly. These previous methods heavily rely on the quality and quantity of the training data. When the training data is not adequate (the ERE training documents contain 1,068 events and 2,448 arguments, while ACE training documents contain more than 4,700 events and 9,700 arguments), the performance is low. In contrast, our approach is unsupervised and can automatically identify events, arguments and assign types/roles, and is not tied to one event schema.

3.6 Event Extraction for Biomedical Domain

To demonstrate the portability of our approach to a new domain, we conduct our experiment on 14 biomedical articles (755 sentences) with perfect AMR annotations (Garg et al., 2016). We utilize a word2vec model[6] trained from all paper abstracts from PubMed[7] and full-text documents from the PubMed Central Open Access subset. To evaluate the performance, we randomly sample 100 sentences and ask a biomedical scientist to assess the correctness of each event and argument role. Our approach achieves 83.1% precision on trigger labeling (619 events in total) and 78.4% precision on argument labeling (1,124 arguments in total).

[6] http://bio.nlplab.org/

[7] http://www.ncbi.nlm.nih.gov/pubmed

Method	ERE: Trigger F_1 (%)			ERE: Arg F_1(%)			ACE: Trigger F_1 (%)			ACE: Arg F_1 (%)		
	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
LSTM	41.5	46.8	44.1	9.9	11.6	10.7	66.0	60	62.8	29.3	32.6	30.8
Joint	42.3	41.7	42.0	61.8	23.2	33.7	73.7	62.3	67.5	64.7	44.4	52.7
DMCNN	-	-	-	-	-	-	75.6	63.6	**69.1**	68.8	46.9	**53.5**
Liberal$_{PerfectAMR}$	79.8	50.5	**61.8**	48.9	32.9	**39.3**	-	-	-	-	-	-
Liberal$_{SystemAMR}$	88.5	42.6	57.5	47.6	30.0	36.8	80.7	50.1	61.8	51.9	39.4	44.8

Table 8: Performance on ERE and ACE events.

It demonstrates that our approach can be rapidly adapted to a new domain and discover domain-rich event schema. An example schema for an event type "*Dissociate*" is shown in Figure 7.

Event Type: Dissociate

S1: Ras acts as a molecular switch that is activated upon GTP loading and deactivated upon **hydrolysis** of GTP to GDP.

***Event**: hydrolysis **Arguments**:GTP (Patient), (GDP) (Result)*

S2: Activation requires **dissociation** of protein-bound GDP , an intrinsically slow process that is accelerated by guanine nucleotide exchange factors.

***Event**: dissociation **Arguments**: GDP (Patient)*

S3: His - ubiquitinated proteins were purified by Co2+ metal affinity chromatography in 8M urea **denaturing** conditions.

***Event**: denaturing **Arguments**: proteins(Patient)*

Figure 7: Example Output of the Discovered Biomedical Event Schema.

4 Related Work

Most of previous event extraction work focused on learning supervised models based on symbolic features (Ji and Grishman, 2008; Miwa et al., 2009; Liao and Grishman, 2010; Liu et al., 2010; Hong et al., 2011; McClosky et al., 2011; Sebastian and Andrew, 2011; Chen and Ng, 2012; Li et al., 2013) or distributional features through deep learning (Chen et al., 2015; Nguyen and Grishman, 2015). They usually rely on a predefined event schema and a large amount of training data. Compared with other paradigms such as Open Information Extraction (Etzioni et al., 2005; Banko et al., 2007; Banko et al., 2008; Etzioni et al., 2011; Ritter et al., 2012), Preemptive IE (Shinyama and Sekine, 2006), On-demand IE (Sekine, 2006) and semantic frame based event discovery (Kim et al., 2013), our approach can explicitly name each event type and argument role. Some recent work focused on universal schema discovery (Chambers and Jurafsky, 2011; Pantel et al., 2012; Yao et al., 2012; Yao et al., 2013; Chambers, 2013; Nguyen et al., 2015). However, the schemas discovered from these methods are rather static and they are not customized for any specific input corpus.

Our work is also related to efforts at composing word embeddings using syntactic structures (Hermann and Blunsom, 2013; Socher et al., 2013a; Socher et al., 2013b; Bowman et al., 2014; Zhao et al., 2015). Our trigger sense representation is similar to Word Sense Induction (Navigli, 2009; Bordag, 2006; Pinto et al., 2007; Brody and Lapata, 2009; Manandhar et al., 2010; Navigli and Lapata, 2010; Van de Cruys and Apidianaki, 2011; Wang et al., 2015b). Besides word sense, we exploit related concepts to enrich trigger representation.

5 Conclusions and Future Work

We proposed a novel Liberal event extraction framework which combines the merits of symbolic semantics and distributed semantics. Experiments on news and biomedical domain demonstrate that this framework can discover explicitly defined rich event schemas which cover not only most types in existing manually defined schemas, but also new event types and argument roles. The granularity of event types is also customized for specific input corpus. And it can produce high-quality event annotations simultaneously without using annotated training data. In the future, we will extend this framework to other Information Extraction tasks.

Acknowledgements

We would like to thank Kevin Knight and Jonathan May (ISI) for sharing biomedical AMR annotations. This work was supported by the U.S. ARL NS-CTA No. W911NF-09-2-0053 and DARPA DEFT No. FA8750-13-2-0041, and in part by NSF IIS-1523198, IIS-1017362, IIS-1320617 and IIS-1354329, and NIH BD2K grant 1U54GM114838. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

C. F. Baker and H. Sato. 2003. The framenet data and software. In *Proc. ACL2003*.

L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider. 2013. Abstract meaning representation for sembanking. In *Proc. ACL2013 Workshop on Linguistic Annotation and Interoperability with Discourse*.

M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. 2007. Open information extraction for the web. In *Proc. IJCAI2007*.

M. Banko, O. Etzioni, and T. Center. 2008. The tradeoffs between open and traditional relation extraction. In *Proc. ACL-HLT2008*.

S. Bordag. 2006. Word sense induction: Triplet-based clustering and automatic evaluation. In *Proc. EACL2006*.

S. Bowman, C. Potts, and C. Manning. 2014. Recursive neural networks for learning logical semantics. *CoRR*, abs/1406.1827.

S. Brody and M. Lapata. 2009. Bayesian word sense induction. In *Proc. EACL2009*.

N. Chambers and D. Jurafsky. 2011. Template-based information extraction without the templates. In *Proc. ACL-HLT2011*.

N. Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, volume 13, pages 1797–1807.

C. Chen and V. Ng. 2012. Joint modeling for chinese event extraction with rich linguistic features. In *In COLING*. Citeseer.

Y. Chen, L. Xu, K. Liu, D. Zeng, and J. Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proc. ACL2015*.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

O. Etzioni, M. Cafarella, D. Downey, A. Popescu, T. Shaked, S. Soderland, D. Weld, and A. Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165:91–134.

O. Etzioni, A. Fader, J. Christensen, S. Soderland, and M. Mausam. 2011. Open information extraction: The second generation. In *Proc. IJCAI2011*, volume 11, pages 3–10.

S. Garg, A. Galstyan, U. Hermjakob, and D. Marcu. 2016. Extracting biomolecular interactions using semantic parsing of biomedical text. In *Proc. AAAI*.

R. Grishman and B. Sundheim. 1996. Message understanding conference-6: A brief history. In *Proc. COLING1996*.

Z. Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

K. Hermann and P. Blunsom. 2013. The role of syntax in vector space models of compositional semantics. In *Proc. ACL2013*.

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

Y. Hong, J. Zhang, B. Ma, J. Yao, G. Zhou, and Q. Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proc. ACL*, pages 1127–1136. Association for Computational Linguistics.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: the 90% solution. In *Proc. NAACL2006*.

H. Ji and R. Grishman. 2008. Refining event extraction through cross-document inference. In *ACL*.

H. Ji and R. Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proc. ACL2011*.

H. Kim, X. Ren, Y. Sun, C. Wang, and J. Han. 2013. Semantic frame-based document representation for comparable corpora. In *ICDM*.

K. Kipper, A. Korhonen, N. Ryant, and M. Palmer. 2008. A large-scale classification of english verbs. *Language Resources and Evaluation Journal*, 42(1):21–40.

Q. Li, H. Ji, and L. Huang. 2013. Joint event extraction via structured prediction with global features. In *Proc. ACL*, pages 73–82.

S. Liao and R. Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proc. ACL*.

B. Liu, L. Qian, H. Wang, and G. Zhou. 2010. Dependency-driven feature-based learning for extracting protein-protein interactions from biomedical text. In *Proc. COLING*.

U. Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416.

S. Manandhar, I. Klapaftis, D. Dligach, and S. Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proc. ACL2010 international workshop on semantic evaluation*.

Dan Klein Christopher D Manning. 2003. Natural language parsing. In *Advances in Neural Information Processing Systems 15: Proceedings of the 2002 Conference*, volume 15, page 3. MIT Press.

D. M. Marie-Catherine, Bill M., and Christopher D. M. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings LREC*, pages 449,454.

D. McClosky, M. Surdeanu, and C. D. Manning. 2011. Event extraction as dependency parsing. In *ACL*, pages 1626–1635.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

M. Miwa, R. Stre, Y. Miyao, and J. Tsujii. 2009. A rich feature vector for protein-protein interaction extraction from multiple corpora. In *Proc. EMNLP*.

R. Navigli and M. Lapata. 2010. An experimental study of graph connectivity for unsupervised word sense disambiguation. *Pattern Analysis and Machine Intelligence*, 32(4):678–692.

R. Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

T. Nguyen and R. Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. *Volume 2: Short Papers*, page 365.

K. Nguyen, X. Tannier, O. Ferret, and R. Besançon. 2015. Generative event schema induction with entity disambiguation. In *Proc. ACL*.

B. Onyshkevych, M. E. Okurowski, and L. Carlson. 1993. Tasks, domains, and languages for information extraction. In *TIPSTER*.

M. Palmer, D. Gildea, and P. Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

P. Pantel, T. Lin, and M. Gamon. 2012. Mining entity types from query logs via user intent modeling. In *Proc. ACL2012*.

D. Pinto, P. Rosso, and H. Jimenez-Salazar. 2007. Upv-si: Word sense induction using self term expansion. In *Proc. ACL2007 International Workshop on Semantic Evaluations*.

S. Pradhan, L. Ramshaw, M. Marcus, M. Palmer, R. Weischedel, and N. Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proc. CONLL2011*.

A. Ritter, O. Etzioni, and S. Clark. 2012. Open domain event extraction from twitter. In *Proc. SIGKDD2012*, pages 1104–1112. ACM.

R. Sebastian and M. Andrew. 2011. Fast and robust joint models for biomedical event extraction. In *EMNLP*.

S. Sekine. 2006. On-demand information extraction. In *Proc. COLING-ACL2006*.

Y. Shinyama and S. Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proc. HLT-NAACL2006*.

R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proc. EMNLP*, pages 151–161.

R. Socher, A. Karpathy, Q. V. Le, C. Manning, and A. Y. Ng. 2013a. Grounded compositional semantics for finding and describing images with sentences. *TACL2013*.

R. Socher, A. Perelygin, J. Wu, J. Chuang, C. Manning, A. Ng, and C. Potts. 2013b. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. EMNLP2013*.

E. Tjong, K. Sang, and F. Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proc. CONLL2003*.

T. Van de Cruys and M. Apidianaki. 2011. Latent semantic word sense induction and disambiguation. In *Proc. ACL-HLT2011*.

C. Wang, N. Xue, and S. Pradhan. 2015a. Boosting transition-based amr parsing with refined actions and auxiliary analyzers. In *Proc. ACL2015*.

J. Wang, M. Bansal, K. Gimpel, B. Ziebart, and T. Clement. 2015b. A sense-topic model for word sense induction with unsupervised data enrichment. *TACL*, 3:59–71.

L. Yao, S. Riedel, and A. McCallum. 2012. Probabilistic databases of universal schema. In *Proc. NIPS2012 Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

L. Yao, S. Riedel, and A. McCallum. 2013. Universal schema for entity type prediction. In *Proc. NIPS2013 Workshop on Automated Knowledge Base Construction*.

Y. Zhao, Z. Liu, and M. Sun. 2015. Phrase type sensitive tensor indexing model for semantic composition. In *Proc. AAAI2015*.

Z. Zhong and H. T. Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *Proceedings of the ACL 2010 System Demonstrations*, pages 78–83.

Jointly Event Extraction and Visualization on Twitter via Probabilistic Modelling

Deyu Zhou[†‡] **Tianmeng Gao**[†] **Yulan He**[§]
[†] School of Computer Science and Engineering, Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University, China
[‡] State Key Laboratory for Novel Software Technology, Nanjing University, China
[§] School of Engineering and Applied Science, Aston University, UK
d.zhou@seu.edu.cn, gaotianmeng@seu.edu.cn, y.he@cantab.net

Abstract

Event extraction from texts aims to detect structured information such as what has happened, to whom, where and when. Event extraction and visualization are typically considered as two different tasks. In this paper, we propose a novel approach based on probabilistic modelling to jointly extract and visualize events from tweets where both tasks benefit from each other. We model each event as a joint distribution over named entities, a date, a location and event-related keywords. Moreover, both tweets and event instances are associated with coordinates in the visualization space. The manifold assumption that the intrinsic geometry of tweets is a low-rank, non-linear manifold within the high-dimensional space is incorporated into the learning framework using a regularization. Experimental results show that the proposed approach can effectively deal with both event extraction and visualization and performs remarkably better than both the state-of-the-art event extraction method and a pipeline approach for event extraction and visualization.

1 Introduction

Event extraction, one of the important and challenging tasks in information extraction, aims to detect structured information such as what has happened, to whom, where and when. The outputs of event extraction could be beneficial for downstream applications such as summarization and personalized news systems. Data visualization, an important exploratory data analysis task, provides a simple way to reveal the relationships among data (Nakaji and Yanai, 2012).

Although event extraction and visualization are two different tasks and typically studied separately in the literature, these two tasks are highly related. Documents which are close to each other in the low-dimensional visualization space are likely to describe the same event. Events in nearby locations in the visualization space are likely to share similar event elements. Therefore, jointly learning the two tasks could potentially bring benefits to each other. However, it is not straightforward to learn event extraction and visualization jointly since event extraction usually relies on semantic parsing results (McClosky et al., 2011) while visualization is accomplished by dimensionality reduction (Iwata et al., 2007; López-Rubio et al., 2002).

In this paper, we propose a novel probabilistic model, called Latent Event Extraction & Visualization (LEEV) model, for joint event extraction and visualization on Twitter. It is partly inspired by the Latent Event Model (LEM) (Zhou et al., 2015) where each tweet is assigned to one event instance and each event is modeled as a joint distribution over named entities, a date/time, a location and the event-related keywords. Going beyond LEM, we assume that each event is not only modeled as the joint distribution over event elements as in (Zhou et al., 2015), but also associate with coordinates in the visualization space. The Euclidean distance between a tweet and each events determines which event the tweet should be assigned to. Furthermore, the manifold assumption that the intrinsic geometry of tweets is a low-rank, non-linear manifold within the high-dimensional space, is incorporated in the learning framework using a regularization. Experimental results show that the proposed approach can effectively deal with both event extraction and visualization tasks and performs remarkably better than both the state-of-the-art event extraction method

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 269–278,
Berlin, Germany, August 7-12, 2016.

and a pipeline approach for event extraction and visualization.

2 Related Work

Our proposed work is related to two lines of research, event extraction and joint topic modeling and visualization.

2.1 Event Extraction

Research on event extraction of tweets can be categorized into domain-specific and open domain approaches. Domain-specific approaches usually have target events in mind and aim to extract events from a particular location or for emergency response during natural disasters. Anantharam et al. (2015) focused on extracting city events by solving a sequence labeling problem. Evaluation was carried out on a real-world dataset consisting of event reports and tweets collected over four months from San Francisco Bay Area. TSum4act (Nguyen et al., 2015) was designed for emergency response during disasters and was evaluated on a dataset containing 230,535 tweets.

Most of open domain approaches focused on extracting a summary of events discussed in social media. For example Benson et al. (2011) proposed a structured graphical model which simultaneously analyzed individual messages, clustered, and induced a canonical value for each event. Capdevila et al. (2015) proposed a model named Tweet-SCAN based on the hierarchical Dirichlet process to detect events from geo-located tweets. To extract more information, a system called SEEFT (Wang et al., 2015) used links in tweets and combined tweets and linked articles to identify events. Zhou et al. (2014; 2015) proposed an unsupervised Bayesian model called latent event model (LEM) for event extraction from Twitter by assuming that each tweet message is assigned to one event instance and each event is modeled as a joint distribution over named entities, a date/time, a location and the event-related keywords. Our proposed method is partly inspired by (Zhou et al., 2015). However, different from previous methods, our approach not only extracts the structured representation of events, but also learns the coordinates of events and tweets simultaneously.

2.2 Joint Topic Modeling and Visualization

Since our proposed approach can be considered as a variant of topic model, we also review the related work of joint topic modeling and visualization here.

Traditionally, topic modeling and visualization are considered as two disjoint tasks and can be combined for pipeline processing. For example, probabilistic latent semantic analysis (Hofmann, 1999) can be first performed followed by parametric embedding (Iwata et al., 2007). Another pipeline approach (Millar et al., 2009) is based on latent Dirichlet allocation followed by self-organizing maps (López-Rubio et al., 2002).

Jointly modeling topics and visualization is a new problem explored in very few works. The state-of-the-art is a joint approach proposed in (Iwata et al., 2008). In this model, both documents and topics are assumed to have latent coordinates in a visualization space. The topic proportions of a document are determined by the distances between the document and the topics in the visualization space, and each word is drawn from one of the topics according to the document's topic proportions. A visualization was obtained by fitting the model to a given set of documents using the EM algorithm. Following the same line, by considering the local consistency in terms of the intrinsic geometric structure of the document manifold, an unsupervised probabilistic model, called SEMAFORE, was proposed in (Le and Lauw, 2014a) by preserving the manifold in the lower dimensional space. In (Le and Lauw, 2014b), a semantic visualization model is learned by associating each document a coordinate in the visualization space, a multinomial distribution in the topic space, and a directional vector in a high-dimensional unit hypersphere in the word space.

Our work is partly inspired by (Le and Lauw, 2014a). However, our proposed approach differs from (Le and Lauw, 2014a) in that events, instead of topics, are modelled as the joint distribution over event elements. Both tweets and events are associate with coordinates in the visualization space.

3 Methodology

We follow the same pre-processing steps described in (Zhou et al., 2015) to filter out non-event-related tweets and extract dates, locations, and named entities by temporal resolution, part-of-speech (POS) tagging and named entity recognition. The pre-processed tweets are then fed into our proposed model for event extraction and visu-

Table 1: Definition of Notations.

Notation	Definition
e	event index, $e \in \{1..E\}$
$W = \{w_m\}$	tweets, $m \in \{1..M\}$
$Z = \{z_m\}$	event labels for tweets
N_{my}	number of named entities in w_m
N_{md}	number of dates in w_m
N_{ml}	number of locations in w_m
N_{mk}	number of keywords in w_m
θ_{ey}	probability of named entity y in event e
φ_{ed}	probability of date d in event e
ψ_{el}	probability of location l in event e
ω_{ek}	probability of keyword k in event e
$\beta, \gamma, \eta, \lambda$	Dirichlet hyperparameters
χ, δ	Normal hyperparameters
G	dimension of visualization space

alization. We describe our model in more details below.

3.1 Latent Event Extraction & Visualization (LEEV) Model

We propose an unsupervised latent variable model called the Latent Event Extraction & Visualization (LEEV) model which simultaneously extracts events from tweets and generates a visualization of the events. Table 1 lists notations used in this paper.

In LEEV, each tweet message w_m, $m \in \{1...M\}$ is associated with a latent coordinate x_m in the visualization space. Each event $e \in \{1...E\}$ is also associated with a coordinate ϕ_e. Assuming that each tweet message $w_m, m \in \{1...M\}$ is assigned to one event instance $z_m = e$ and e is modeled as a joint distribution over named entities y, the date d when e happened, the location l and the event-related keywords k, the generative process of the model is described as follows:

- For each event $e \in \{1..E\}$, draw multinomial distributions $\theta_e \sim$ Dirichlet(β), $\varphi_e \sim$ Dirichlet(γ), $\psi_e \sim$ Dirichlet(η), $\omega_e \sim$ Dirichlet(λ), draw event coordinate $\phi_e \sim$ Normal$(0, \chi^{-1}I)$;
- For each tweet $w_m, m \in \{1..M\}$
  - * Choose tweet coordinate: $x_m \sim$ Normal$(0, \delta^{-1}I)$;
  - * Choose an event $z_m = e \sim$ Multinomial$(\{P(e|x_m, \Phi)_{e=1}^{E}\})$;
  - * For each named entity in the tweet w_m, choose a named entity $y \sim$ Multinomial(θ_e);

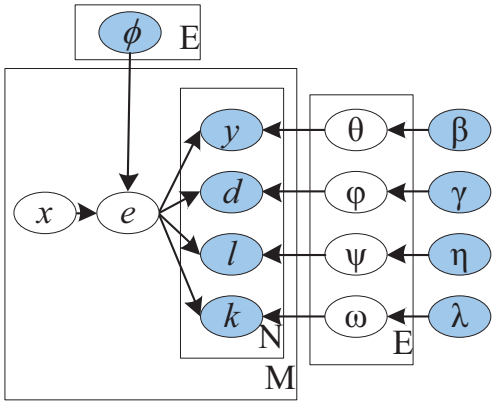


Figure 1: Latent Event Extraction & Visualization (LEEV) Model.

  - * For each date in the tweet w_m, choose a date $d \sim$ Multinomial(φ_e);
  - * For each location in the tweet w_m, choose a location $l \sim$ Multinomial(ψ_e);
  - * For other words in the tweet w_m, choose a word $k \sim$ Multinomial(ω_e).

Here, $\beta, \gamma, \eta, \lambda, \chi, \delta$ are priors, I is an identity matrix, and $P(e|x_m, \Phi)$ is the probability of the tweet w_m with coordinate x_m belonging to the event e. It is defined as,

$$P(e|x_m, \Phi) = \frac{\exp(-\frac{1}{2} \parallel x_m - \phi_e \parallel^2)}{\sum_{e'=1}^{E} \exp(-\frac{1}{2} \parallel x_m - \phi_{e'} \parallel^2)}. \quad (1)$$

It is calculated as the normalized Euclidean distance between a tweet w_m and an event e. Using this equation, when the Euclidean distance between a tweet w_m and and an event e is small, the probability that tweet w_m belongs to event e becomes large. The graphical model of LEEV is shown in Figure 1.

The parameters to be learned are $\Theta = \{\theta_e, \varphi_e, \psi_e, \omega_e\}_{e=1}^{E}$, tweets' coordinates $\mathcal{X} = \{x_m\}_{m=1}^{M}$ and events' coordinates $\Phi = \{\phi_e\}_{e=1}^{E}$, which are collectively denoted as $\mathcal{B} = \langle \Theta, \mathcal{X}, \Phi \rangle$. Let

$$H(w_m, e) = \prod_{n=1}^{N_{my}} P(y_n|\theta_e) \prod_{n=1}^{N_{md}} P(d_n|\varphi_e) \prod_{n=1}^{N_{ml}} P(l_n|\psi_e) \prod_{n=1}^{N_{mk}} P(k_n|\omega_e).$$

The log likelihood of $\mathcal{B}$ given tweets W is,

$$\mathcal{L}(\mathcal{B}|W) = \sum_{m=1}^{M} \log\Big\{\sum_{e=1}^{E} P(e|x_m, \Phi) \times H(w_m, e)\Big\} + \sum_{m=1}^{M} \log(P(x_m)) + \sum_{e=1}^{E} \log(P(\phi_e)) + \sum_{e=1}^{E} \{\log(P(\theta_e) * P(\varphi_e) * P(\psi_e) * P(\omega_e))\}. \quad (2)$$

For the events' coordinate ϕ_e and tweets' coordinate x_m, we use a Gaussian prior with a zero mean and a spherical covariance:

$$p(\phi_e) = (\frac{\chi}{2\pi})^{\frac{G}{2}} \exp(-\frac{\chi}{2} \parallel \phi_e \parallel^2)$$
$$p(x_m) = (\frac{\delta}{2\pi})^{\frac{G}{2}} \exp(-\frac{\delta}{2} \parallel x_m \parallel^2).$$

3.2 LEEV with Manifold Regularization

Recent studies suggest that the intrinsic geometry of textual data is a low-rank, non-linear manifold lying in the high dimensional space (Cai et al., 2008; Zhang et al., 2005). We therefore assume that when two tweets w_i and w_j are close in the intrinsic geometry of the manifold Υ, their low-rank representations should be close as well. To capture this assumption, we consider Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2003) which has been commonly used in manifold learning algorithms (Le and Lauw, 2014a). It constructs a k-nearest neighbors graph to represent data residing on a low-dimensional manifold embedded in a higher-dimensional space. In this paper, we use LE to incorporate neighborhood information of tweets. We construct a manifold graph with edges connecting two data points w_i and w_j. Set the edge weight $v_{ij} = 1$ if w_j is one of the k-nearest neighbors of w_i; Otherwise $v_{ij} = 0$. That makes LEEV an special case when $\xi = 0$. We represent each tweet as a word-count vector, i.e., each element of a vector is weighted by its corresponding term frequency, and use cosine similarity metric to measure the distance between tweets when constructing the manifold graph. We also tried vectors with the TFIDF weighting strategy to represent tweets and found word-count vectors give better results.

We apply a regularization framework to incorporate a manifold structure into a learning model. The new regularized log-likelihood function L is

$$L(\mathcal{B}|W, \Upsilon) = L(\mathcal{B}|W) - \frac{\xi}{2}\mathcal{R}(\mathcal{B}|\Upsilon), \quad (3)$$

where ξ is the regularization parameter. The second component $\mathcal{R}$ is a regularization function, which consists of two parts:

$$\mathcal{R}(\mathcal{B}|\Upsilon) = \mathcal{R}_+(\mathcal{B}|\Upsilon) + \mathcal{R}_-(\mathcal{B}|\Upsilon), \quad (4)$$

$$\mathcal{R}_+(\mathcal{B}|\Upsilon) = \sum_{i,j=1;i\neq j}^{M} v_{ij} \cdot \mathcal{F}(w_i, w_j), \quad (5)$$

$$\mathcal{R}_-(\mathcal{B}|\Upsilon) = \sum_{i,j=1;i\neq j}^{M} \frac{1 - v_{ij}}{\mathcal{F}(w_i, w_j) + 1}, \quad (6)$$

where $\mathcal{F}$ is a distance function that operates on the low rank space. We define $\mathcal{F}$ as the squared Euclidean distance of coordinates in the visualization space. $\mathcal{F}(w_i, w_j)$is computed as follows:

$$\mathcal{F}(w_i, w_j) = \parallel x_i - x_j \parallel^2 . \quad (7)$$

Minimizing $\mathcal{R}_+$ leads to minimizing the distance between neighbors and minimizing $\mathcal{R}_-$ leads to maximizing the distance between non-neighbors. By enforcing manifold learning, we capture the spirit of keeping neighbors close and keeping none-neighbors apart.

3.3 Parameter Estimation

As in Equation 2, the presence of the sum over e prevents the logarithm form directly acting on the joint distribution. Assuming that the corresponding latent event z_m of each tweet w_m is known, $\{W, Z\}$ is called the complete data. Maximizing the log likelihood of the complete data, $\log P(W, Z|\mathcal{B})$, can be easily done. However, in practice we don't observe the latent variables Z and only have the incomplete data W. Therefore, the expectation maximization (EM) algorithm is employed to handle the incomplete data. EM involves an efficient iterative procedure to compute the Maximum Likelihood estimation of probabilistic models with unobserved latent variables involved.

The class posterior probability of the m^{th} tweet under the current parameter values $\hat{\mathcal{B}}$, $P(z_m = e|m, \hat{\mathcal{B}})$, is given as follows:

$$P(z_m = e|m, \hat{\mathcal{B}}) = \frac{P(z_m = e|\hat{x_m}, \hat{\Phi}, \hat{\mathcal{B}}) \times H(w_m, e)}{\sum_{e'=1}^{E} P(z_m = e'|\hat{x_m}, \hat{\Phi}, \hat{\mathcal{B}}) \times H(w_m, e')}, \quad (8)$$

which corresponds to the E-step in EM algorithm.

In M-step, model parameters $\mathcal{B}$ are updated by maximizing the regularized conditional expectation of the complete data log likelihood with priors defined as follows:

$$\begin{aligned}\mathcal{Q}(\mathcal{B}|\hat{\mathcal{B}}) &= \sum_{m=1}^{M}\sum_{e=1}^{E}\{P(z_m = e|m,\hat{\mathcal{B}}) \\ &\times \log[P(e|x_m,\Phi)\times H(w_m,e)]\} \\ &+ \sum_{m=1}^{M}\log(P(x_m)) + \sum_{e=1}^{E}\log(P(\phi_e)) \\ &+ \sum_{e=1}^{E}\{\log(P(\theta_e)*P(\varphi_e)*P(\psi_e)*P(\omega_e))\} \\ &- \frac{\xi}{2}\mathcal{R}(\mathcal{B}|\Upsilon),\end{aligned}$$

where $P(z_m = e|m,\hat{\mathcal{B}})$ is calculated in E-step.

By maximizing $\mathcal{Q}(\mathcal{B}|\hat{\mathcal{B}})$ w.r.t $\theta_{ey}, \varphi_{ed}, \psi_{el}, \omega_{ek}$, the next estimates are given as follows,

$$\theta_{ey} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_{my}} I(y_{mn}=y)P(z_m=e|m,\hat{\mathcal{B}}) + \beta}{\sum_{y=1}^{Y}\sum_{m=1}^{M}\sum_{n=1}^{N_{my}} I(y_{mn}=y)P(z_m=e|m,\hat{\mathcal{B}}) + Y\beta},$$

$$\varphi_{ed} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_{md}} I(d_{mn}=d)P(z_m=e|m,\hat{\mathcal{B}}) + \gamma}{\sum_{d=1}^{D}\sum_{m=1}^{M}\sum_{n=1}^{N_{md}} I(d_{mn}=d)P(z_m=e|m,\hat{\mathcal{B}}) + D\gamma},$$

$$\psi_{el} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_{ml}} I(l_{mn}=l)P(z_m=e|m,\hat{\mathcal{B}}) + \eta}{\sum_{l=1}^{L}\sum_{m=1}^{M}\sum_{n=1}^{N_{ml}} I(l_{mn}=l)P(z_m=e|m,\hat{\mathcal{B}}) + L\eta},$$

$$\omega_{ek} = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N_{mk}} I(k_{mn}=k)P(z_m=e|m,\hat{\mathcal{B}}) + \lambda}{\sum_{k=1}^{K}\sum_{m=1}^{M}\sum_{n=1}^{N_{mk}} I(k_{mn}=k)P(z_m=e|m,\hat{\mathcal{B}}) + K\eta},$$

where Y, D, L, K are the total numbers of distinct named entities, dates, locations, and words appeared in the whole Twitter corpus, respectively.

ϕ_e and x_m cannot be solved in a closed form, and are estimated by maximizing $\mathcal{Q}(\mathcal{B}|\hat{\mathcal{B}})$ using quasi-Newton method. The gradients of $\mathcal{Q}(\mathcal{B}|\hat{\mathcal{B}})$ w.r.t ϕ_e and x_m are as follows:

$$\begin{aligned}\frac{\partial\mathcal{Q}}{\partial\phi_e} &= \sum_{m=1}^{M} p(e|m,\hat{\mathcal{B}})(p(e|x_m,\Phi)-1)(\phi_e - x_m) \\ &- \chi\phi_e,\end{aligned}$$

$$\begin{aligned}\frac{\partial\mathcal{Q}}{\partial x_m} &= \sum_{e=1}^{E} p(e|m,\hat{\mathcal{B}})(p(e|x_m,\Phi)-1)(x_m - \phi_e) \\ &- \delta x_m - \frac{\xi}{2}\frac{\partial\mathcal{R}(\mathcal{B}|\Upsilon)}{\partial x_m},\end{aligned}$$

where the gradient of $\mathcal{R}(\mathcal{B}|\Upsilon)$ w.r.t. x_m is computed as follows:

$$\begin{aligned}\frac{\partial\mathcal{R}(\mathcal{B}|\Upsilon)}{\partial x_m} &= \sum_{j=1,j\neq m}^{M} 2\upsilon_{mj}(x_m - x_j) \\ &- \sum_{j=1,j\neq m}^{M} 2(x_m - x_j)\frac{1-\upsilon_{mj}}{(\mathcal{F}(x_m,x_j)+1)^2}.\end{aligned}$$

We set the parameter $\chi = 0.00005$, $\delta = 0.05$, $\beta = \gamma = \eta = \lambda = 0.1$ and run EM algorithm for 50 iterations. Finally we select an entity y, a date d, a location l and two keywords k with the highest probabilities to form a tuple $\langle y, d, l, k\rangle$ to represent each potential event.

3.4 Post-processing

In order to filter out spurious events, we calculate the correlation coefficient of each event element. Remove the event element if its correlation coefficient is less than a threshold C_e and remove the event if the sum of the correlation coefficients of all its four event elements is less than C_t.

For an event element A, its correlation coefficient is calculated below:

$$C_A = \log\frac{\sum_{B\in\Omega, B\neq A}\#(A,B)}{\#(A)}, \qquad (0)$$

where Ω is the set of the four event elements $\langle y, d, l, k\rangle$ and $\#(x)$ indicates the number of times x appeared in the whole corpus. We empirically set C_e to 0.4 and C_t to 4.

4 Experiments

In this section, we firstly describe the datasets used in our experiments and then present the experimental results.

4.1 Setup

We choose two datasets for model evaluation. The first one is the First Story Detection (FSD) dataset (Petrovic et al., 2013) (Dataset I) which contains 2,499 tweets published between 7th July and 12th September 2011. These tweets have been manually annotated with 27 events, covering a wide range of topics from accidents to science discoveries and from disasters to celebrity news. We filter out events mentioned in less than 15 tweets since events mentioned in very few tweets are less likely to be significant. The final dataset contains 2,453 tweets annotated with 20 events. This dataset has been previously used for evaluating event extraction models and the state-of-the-art results have been achieved using LEM (Zhou et al., 2015). We also create another dataset, called Dataset II, by manually annotating 1,000 tweets published in December 2010. A total of 20 events are annotated.

We compare our model with LEM (Zhou et al., 2015), which also extracts events as 4-tuples $\langle$

y,d,l,k $\rangle$. The main difference between LEM and our model is that LEM directly estimates the event distribution from the sampled latent event labels, while we derive the distribution from coordinates of tweets and events x_m, ϕ_e. We re-implemented the system described in (Zhou et al., 2015) and used the same evaluation metrics such as precision, recall and F-measure. Precision is defined as the proportion of the correctly identified events out of the system returned events. Recall is defined as the proportion of correctly identified true events. For calculating the precision of the 4-tuple $\langle y, d, l, k \rangle$, we use following criteria:

- Do the entity y, location l, date d and keyword k that we have extracted refer to the same event?
- If the extracted representation contains keywords, are they informative enough to tell us what happened?

As mentioned in Section 2, PE (Iwata et al., 2007) is a nonlinear visualization method which takes a set of class posterior vectors as input and embeds samples in a low-dimensional Euclidean space. By minimizing the sum of Kullback-Leibler divergences, PE tries to preserve the posterior structure in the embedding space. In order to evaluate the visualization results, we compare our proposed method with a pipeline approach, event extraction using LEM (Zhou et al., 2015) followed by event visualization using PE (Iwata et al., 2007), named as LEM+PE.

4.2 Event Extraction Results

Table 2 shows the event extraction results on the two datasets. LEEV+R is LEEV with manifold regularization incorporated, in which the model parameters are estimated by the EM algorithm described in Section 3.3. For LEEV and LEEV+R, the number of events, E, is set to 50 for both datasets. For LEEV+R, the number of neighborhood size k is set to 10 and the regularization parameter ξ is set to 1. For LEM, E is set to 25 for both datasets following the suggestion in (Zhou et al., 2015).

We ran our experiments on a server equipped with 3.40 GHz Intel Corel i7 CPU and 8 GB memory. The average running time of LEEV is 2328.1 seconds on Dataset I and 940.7 seconds on Dataset II for one iteration. The average running time of LEEV+R is 2612.7 seconds on Dataset I and 1296.4 seconds on Dataset II for one iteration.

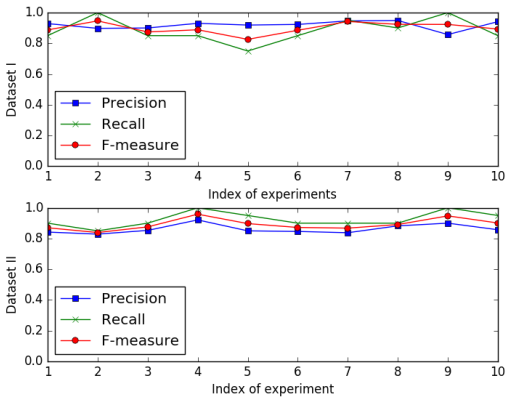


Figure 2: Experimental results of LEEV+R in 10 different runs.

Table 2: Comparison of the event extraction results on the two datasets.

Dataset I			
Method	Prec. (%)	Rec. (%)	F-measure (%)
LEM	84.00	76.19	80.35
LEEV	**92.10**	80.00	85.62
LEEV+R	91.91	**88.50**	**89.88**
Dataset II			
Method	Prec. (%)	Rec. (%)	F-measure (%)
LEM	80.00	90.00	84.70
LEEV	83.33	**95.00**	88.78
LEEV+R	**86.18**	92.50	**89.19**

It can be observed that both LEEV and LEEV+R outperforms the state-of-the-art results achieved by LEM on Dataset I. In particular, LEEV improves upon LEM by over 5% in F-measure and with regularization, LEEV-R further improves upon LEEV by over 4%. A similar trend is observed on Dataset II where both LEEV and LEEV+R outperforms LEM and the best performance is given by LEEV+R. This shows the effectiveness of using regularization in LEEV. We will further demonstrate its importance in visualization results. Overall, we see superior performance of LEEV+R over the other two models, with the F-measure of over 89% being achieved on both datasets.

As described in Section 3.1, the coordinates of tweets and events are randomly initialized. Therefore, we would like to see whether the performance of event extraction is influenced heavily by random initialization. We repeat the experiments

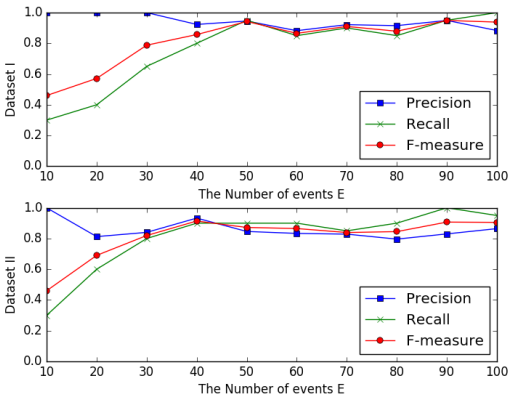


Figure 3: The performance of LEEV+R with different number of events E.

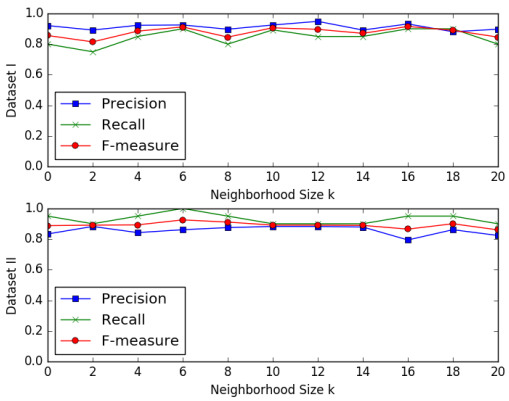


Figure 4: The performance of LEEV+R with different neighborhood size k.

on the two datasets for 10 times using LEEV+R. The experimental results are shown in Figure 2. It can be observed that the performance of LEEV+R is quite stable on both datasets. The standard deviation of F-measure on both Dataset I and I-I is 0.036, which shows that random initialization does not have significant impact on the final performance of the model.

4.3 Impact of Number of Events E

We need to pre-set the number of events E in the proposed approach. Figure 3 shows the performance of event extraction based on LEEV+R versus different values of E on the two datasets. It can be observed that the performance of the proposed approach improves with the increased value of E and when E goes beyond 50, we notice a more balanced precision/recall values and a relatively stable F-measure. This shows that the proposed approach is less sensitive to the number of events E so long as E is set to a relatively larger value.

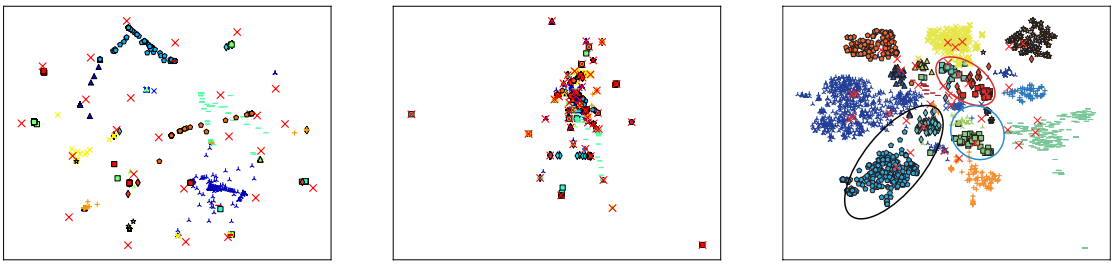
4.4 Impact of Neighborhood Size

As described in Section 3.2, the neighborhood information of tweets is incorporated into the learning framework. A manifold graph with edges connecting two tweets (or data points) w_i and w_j is constructed by setting the edge weight $v_{ij} = 1$ if w_j is among the k-nearest neighbors of w_i and $v_{ij} = 0$ otherwise. Therefore, it is crucial to see whether the performance of LEEV+R heavily depends on the setting of k. Figure 4 shows the performance of our proposed approach with different neighborhood size k. It can be observed that the performance of LEEV+R is quite stable and independent of the k value.

4.5 Visualization Results

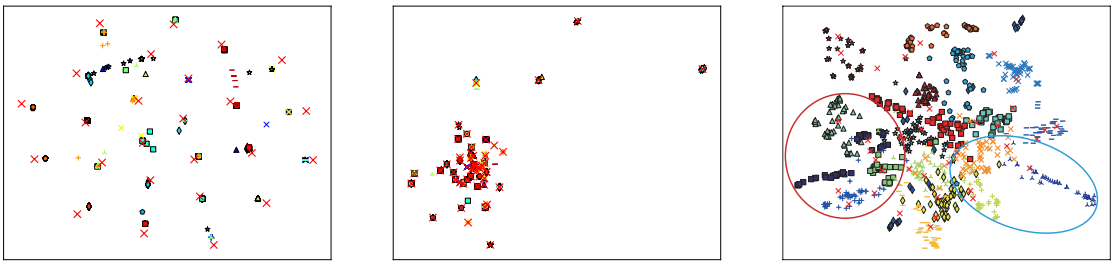
We show the visualization results produced by different approaches on the two datasets in Figure 5 and 6 respectively. We compare LEEV and LEEV+R with the pipeline approach LEM+PE. In the figures, each point represents a tweet and different shapes and colors represent the different events they are associated with. Each red cross represents an extracted event with coordinate ϕ_z.

For Dataset I, it can be observed from Figure 5(a) that the visualization result generated by LEM+PE is not informative. Tweets from different events are mixed together and events are evenly distributed across the whole visualization space. Thus, this visualization does not provide any sensible information about the relationships between tweets and events. The result generated by LEEV without manifold Regularization unit $\mathcal{R}$ seems better than that from LEM+PE, as shown in Figure 5(b). However, a large amount of tweets crowded together at the center, which makes it difficult to reveal the relations between tweets and events. The best visualization result is given by LEEV+R as shown in Figure 5(c) that different events are well separated and related events are located nearby. For example, the three events enclosed by a red circle represent "people died in terrorist attacks in Delhi, Oslo and Norway" respectively, while three events in the blue circle represent "riots in Ealing, Totteham and Croydon", respectively. And two events in the black circle represent "American credit rating" and "House



(a) LEM+PE (b) LEEV (c) LEEV+R

Figure 5: Visualization results on Dataset I.



(a) LEM+PE (b) LEEV (c) LEEV+R

Figure 6: Visualization results on Dataset II.

debt bill", respectively. It shows that LEEV+R with manifold learning incorporated significantly improved upon LEEV without regularization and gives better visualization results. The relationships of events are directly reflected in the distances between their coordinates in the visualization space.

Similar visualization results have been obtained on Dataset II. Figure 6(a) and 6(b) failed to convey the semantic relations between different events. LEEV+R in Figure 6(c) is good at separating tweets from different events. The events in the red circle are the government activities of the United States. The events in the blue circle are categorized as traffic accidents. They are " Transport chaos caused by heavy snow ", "Train to Paris crushed " and "Demonstrators attacked car carrying Prince Charles ". Compared to LEM+PE and LEEV, LEEV+R gives much more informative visualization results.

To further analyze the visualization results in more detail, the 4 representative events and their corresponding tweets in the red circle of Figure 6(c) are visualized in Figure 7. These four events are "Senate vote on repealing gay ban", "US state governor plan to visit North Korea", "Send letter to President Obama to stop tax cut deal" and "Congress passed the Child Nutrition Bill". Their corresponding tweets are denoted as green '△', blue '□', green '□' and blue '+' individually in Figure 7. It can be observed that these four events are all about government activities of the United States, and they are located close to each other in the low-dimensional visualization space. Moreover, the tweets describing the same event are located close to each other and center around their corresponding events, while the tweets describing different events are far away from each other.

5 Conclusions

In this paper, we have proposed an unsupervised Bayesian model, called Latent Event Extraction & Visualization (LEEV) model, to extract the structured representations of events from social media and simultaneously visualize them in a two-dimensional Euclidean space. The proposed approach has been evaluated on two datasets. Experimental results show that the proposed approach outperforms the previously reported best result on

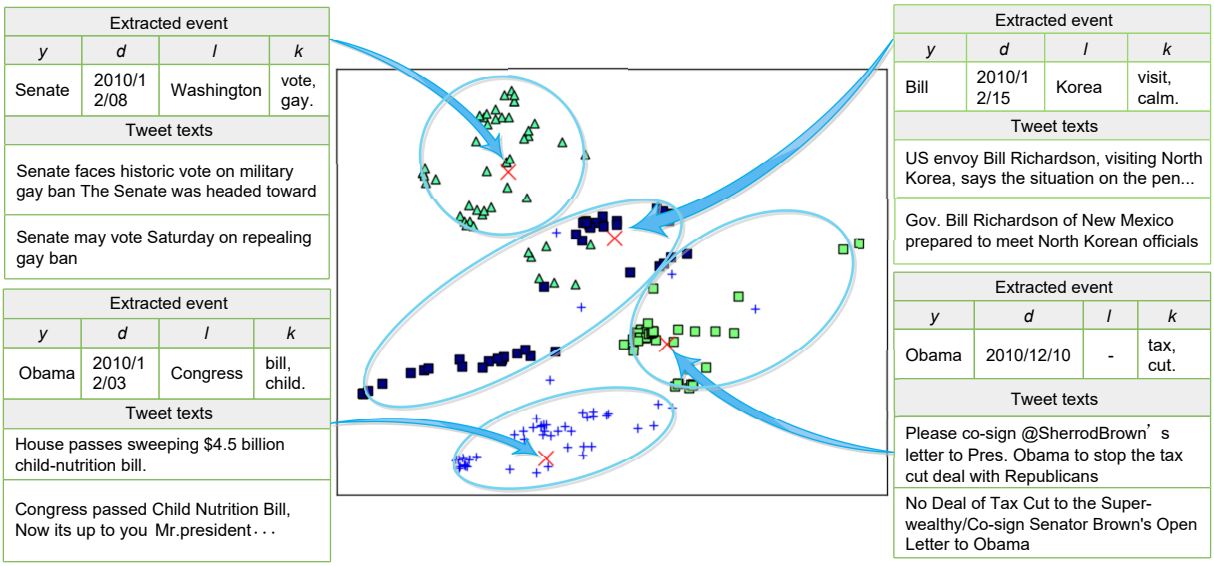


Figure 7: Four representative events and their corresponding tweets in the red circle of Figure 6(c).

Dataset I by nearly 10% in F-measure. Visualization results show that the proposed approach with manifold regularization can significantly improve the quality of event visualization. These results show that by jointly learning event extraction and visualization, our proposed approach is able to give better results on both tasks. In future work, we will investigate scalable and parallel model learning to explore the performance of our model for large-scale real-time event extraction and visualization.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments and suggestions. This work was funded by the National Natural Science Foundation of China (61528302), the Innovate UK under the grant number 101779 and the Collaborative Innovation Center of Wireless Communications Technology.

References

Pramod Anantharam, Payam Barnaghi, Krishnaprasad Thirunarayan, and Amit Sheth. 2015. Extracting city traffic events from social streams. *ACM Transactions on Intelligent Systems and Technology*, 6(4):e110206.

Mikhail Belkin and Partha Niyogi. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, June.

Edward Benson, Aria Haghighi, and Regina Barzilay. 2011. Event discovery in social media feeds. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 389–398, Stroudsburg, PA, USA. Association for Computational Linguistics.

Deng Cai, Qiaozhu Mei, Jiawei Han, and Chengxiang Zhai. 2008. Modeling hidden topics on document manifold. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 911–920, New York, NY, USA. ACM.

Joan Capdevila, Jess Cerquides, Jordi Nin, and Jordi Torres. 2015. Tweet-scan: An event discovery technique for geo-located tweets. In *Artificial Intelligence Research and Development: Proceedings of the 18th International Conference of the Catalan Association for Artificial Intelligence*, volume 277, pages 110–119.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 50–57, New York, NY, USA. ACM.

Tomoharu Iwata, Kazumi Saito, Naonori Ueda, Sean Stromsten, Thomas L. Griffiths, and Joshua B. Tenenbaum. 2007. Parametric embedding for class visualization. *Neural Computation*, 19(9):2536–56.

Tomoharu Iwata, Takeshi Yamada, and Naonori Ueda. 2008. Probabilistic latent semantic visualization: Topic model for visualizing documents. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 363–371, New York, NY, USA. ACM.

Tuan M. V. Le and Hady W. Lauw. 2014a. Manifold learning for jointly modeling topic and visualization. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1960–1967. AAAI Press.

Tuan M.V. Le and Hady W. Lauw. 2014b. Semantic visualization for spherical representation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1007–1016, New York, NY, USA. ACM.

Ezequiel López-Rubio, José Muñoz-Pérez, and José Antonio Gómez-Ruiz. 2002. Self-organizing dynamic graphs. *Neural Processing Letters*, 16(2):93–109(17).

David McClosky, Mihai Surdeanu, and Christopher D. Manning. 2011. Event extraction as dependency parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1626–1635, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jeremy Millar, Gilbert Peterson, and Michael Mendenhall. 2009. Document clustering and visualization with latent dirichlet allocation and self-organizing maps. In *Proceedings of Florida Artificial Intelligence Research Society Conference*.

Yusuke Nakaji and Keiji Yanai. 2012. Visualization of real-world events with geotagged tweet photos. In *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo Workshops*, ICMEW '12, pages 272–277, Washington, DC, USA. IEEE Computer Society.

Minh-Tien Nguyen, Asanobu Kitamoto, and Tri-Thanh Nguyen. 2015. Tsum4act: A framework for retrieving and summarizing actionable tweets during a disaster for reaction. In *Advances in Knowledge Discovery and Data Mining*, pages 64–75. Springer.

Saša Petrovic, Miles Osborne, Richard McCreadie, Craig Macdonald, Iadh Ounis, and Luke Shrimpton. 2013. Can twitter replace newswire for breaking news? In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*.

Yu Wang, David Fink, and Eugene Agichtein. 2015. Seeft: Planned social event discovery and attribute extraction by fusing twitter and web content. In *Proceedings of the Ninth International AAAI Conference on Web and Social Media*, pages 483–492.

Dell Zhang, Xi Chen, and Wee Sun Lee. 2005. Text classification with kernels on the multinomial manifold. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 266–273, New York, NY, USA. ACM.

Deyu Zhou, Liangyu Chen, and Yulan He. 2014. A simple bayesian modelling approach to event extraction from twitter. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 700–705. ACL.

Deyu Zhou, Liangyu Chen, and Yulan He. 2015. An unsupervised framework of exploring events on twitter: Filtering, extraction and categorization. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2468–2474. AAAI Press.

Using Sentence-Level LSTM Language Models for Script Inference

Karl Pichotta
Department of Computer Science
The University of Texas at Austin
pichotta@cs.utexas.edu

Raymond J. Mooney
Department of Computer Science
The University of Texas at Austin
mooney@cs.utexas.edu

Abstract

There is a small but growing body of research on statistical scripts, models of event sequences that allow probabilistic inference of implicit events from documents. These systems operate on structured verb-argument events produced by an NLP pipeline. We compare these systems with recent Recurrent Neural Net models that directly operate on raw tokens to predict sentences, finding the latter to be roughly comparable to the former in terms of predicting missing events in documents.

1 Introduction

Statistical scripts are probabilistic models of event sequences (Chambers and Jurafsky, 2008). A learned script model is capable of processing a document and inferring events that are probable but not explicitly stated. These models operate on automatically extracted structured events (for example, verbs with entity arguments), which are derived from standard NLP tools such as dependency parsers and coreference resolution engines.

Recent work has demonstrated that standard sequence models applied to such extracted event sequences, e.g. discriminative language models (Rudinger et al., 2015) and Long Short Term Memory (LSTM) recurrent neural nets (Pichotta and Mooney, 2016), are able to infer held-out events more accurately than previous approaches. These results call into question the extent to which statistical event inference systems require linguistic preprocessing and syntactic structure. In an attempt to shed light on this issue, we compare existing script models to LSTMs trained as *sentence-level language models* which try to predict the sequence of words in the next sentence from a learned representation of the previous sentences using no linguistic preprocessing.

Some prior statistical script learning systems are focused on knowledge induction. These systems are primarily designed to induce collections of co-occurring event types involving the same entities, and their ability to infer held-out events is not their primary intended purpose (Chambers and Jurafsky, 2008; Ferraro and Van Durme, 2016, *inter alia*). In the present work, we instead investigate the behavior of systems trained to directly optimize performance on the task of predicting subsequent events; in other words, we are investigating statistical models of events in discourse.

Much prior research on statistical script learning has also evaluated on inferring missing events from documents. However, the exact form that this task takes depends on the adopted definition of what constitutes an event: in previous work, events are defined in different ways, with differing degrees of structure. We consider simply using raw text, which requires no explicit syntactic annotation, as our mediating representation, and evaluate how raw text models compare to models of more structured events.

Kiros et al. (2015) introduced *skip-thought vector* models, in which an RNN is trained to encode a sentence within a document into a low-dimensional vector that supports predicting the neighboring sentences in the document. Though the objective function used to train networks maximizes performance on the task of predicting sentences from their neighbors, Kiros et al. (2015) do not evaluate directly on the ability of networks to predict text; they instead demonstrate that the intermediate low-dimensional vector embeddings are useful for other tasks. We directly evaluate the text predictions produced by such sentence-level RNN encoder-decoder models, and measure their utility for the task of predicting subsequent events.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 279–289,
Berlin, Germany, August 7-12, 2016.

We find that, on the task of predicting the text of held-out sentences, the systems we train to operate on the level of raw text generally outperform the systems we train to predict text mediated by automatically extracted event structures. On the other hand, if we run an NLP pipeline on the automatically generated text and extract structured events from these predictions, we achieve prediction performance roughly comparable to that of systems trained to predict events directly. The difference between word-level and event-level models on the task of event prediction is marginal, indicating that the task of predicting the next event, particularly in an encoder-decoder setup, may not necessarily need to be mediated by explicit event structures. To our knowledge, this is the first effort to evaluate sentence-level RNN language models directly on the task of predicting document text. Our results show that such models are useful for predicting missing information in text; and the fact that they require no linguistic preprocessing makes them more applicable to languages where quality parsing and co-reference tools are not available.

2 Background

2.1 Statistical Script Learning

Scripts, structured models of stereotypical sequences of events, date back to AI research from the 1970s, in particular the seminal work of Schank and Abelson (1977). In this conception, scripts are modeled as temporally ordered sequences of symbolic structured events. These models are nonprobabilistic and brittle, and pose serious problems for automated learning.

In recent years, there has been a growing body of research into statistical script learning systems, which enable statistical inference of implicit events from text. Chambers and Jurafsky (2008; 2009) describe a number of simple event co-occurrence based systems which infer (verb, dependency) pairs related to a particular discourse entity. For example, given the text:

> *Andrew Wiles won the 2016 Abel prize for proving Fermat's last theorem*,

such a system will ideally be able to infer novel facts like (*accept*, subject) or (*publish*, subject) for the entity *Andrew Wiles*, and facts like (*accept*, object) for the entity *Abel prize*. A number of other systems inferring the same types of pair events have been shown to provide superior performance in modeling events in documents (Jans et al., 2012; Rudinger et al., 2015).

Pichotta and Mooney (2014) give a co-occurrence based script system that models and infers more complex multi-argument events from text. For example, in the above example, their model would ideally be able to infer a single event like *accept*(*Wiles*, *prize*), as opposed to the two simpler pairs from which it is composed. They provide evidence that modeling and inferring more complex multi-argument events also yields superior performance on the task of inferring simpler (verb, dependency) pair events. These events are constructed using only coreference information; that is, the learned event co-occurrence models do not directly incorporate noun information.

More recently, Pichotta and Mooney (2016) presented an LSTM-based script inference model which models and infers multi-argument events, improving on previous systems on the task of inferring verbs with arguments. This system can incorporate both noun and coreference information about event arguments. We will use this multi-argument event formulation (formalized below) and compare LSTM models using this event formulation to LSTM models using raw text.

2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are neural nets whose computation graphs have cycles. In particular, RNN sequence models are RNNs which map a sequence of inputs $x_1, \ldots, x_T$ to a sequence of outputs $y_1, \ldots, y_T$ via a learned latent vector whose value at timestep t is a function of its value at the previous timestep $t-1$.

The most basic RNN sequence models, so-called "vanilla RNNs" (Elman, 1990), are described by the following equations:

$$z_t = f(W_{i,z}x_t + W_{z,z}z_{t-1})$$
$$o_t = g(W_{z,o}z_t)$$

where x_t is the vector describing the input at time t; z_t is the vector giving the hidden state at time t; o_t is the vector giving the predicted output at time t; f and g are element-wise nonlinear functions (typically sigmoids, hyperbolic tangent, or rectified linear units); and $W_{i,z}$, $W_{z,z}$, and $W_{z,o}$ are learned matrices describing linear transformations. The recurrency in the computation graph arises from the fact that z_t is a function of z_{t-1}.

The more complex Long Short-Term Memory (LSTM) RNNs (Hochreiter and Schmidhuber,

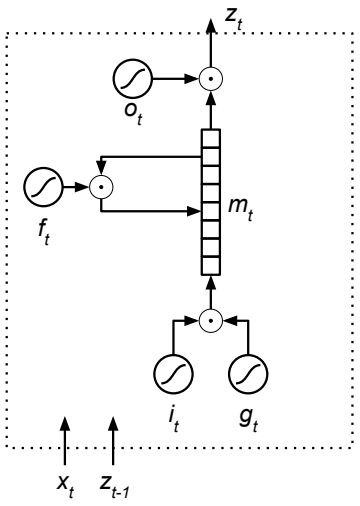


Figure 1: Long Short-Term Memory unit at timestep t. The four nonlinearity nodes (i_t, g_t, f_t, and o_t) all have, as inputs, x_t and z_{t-1}. Small circles with dots are elementwise vector multiplications.

1997) have been shown to perform well on a wide variety of NLP tasks (Sutskever et al., 2014; Hermann et al., 2015; Vinyals et al., 2015, *inter alia*). The LSTM we use is described by:

$$
\begin{aligned}
i_t &= \sigma\left(W_{x,i}x_t + W_{z,i}z_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{x,f}x_t + W_{z,f}z_{t-1} + b_f\right) \\
o_t &= \sigma\left(W_{x,o}x_t + W_{h,i}z_{t-1} + b_o\right) \\
g_t &= \tanh\left(W_{x,m}x_t + W_{z,m}z_{t-1} + b_g\right) \\
m_t &= f_t \circ m_{t-1} + i_t \circ g_t \\
z_t &= o_t \circ \tanh m_t.
\end{aligned}
$$

The model is depicted graphically in Figure 1. The memory vector m_t is a function of both its previous value m_{t-1} and the input x_t; the vector z_t is output both to any layers above the unit (which are trained to predict the output values y_t), and is additionally given as input to the LSTM unit at the next timestep $t+1$. The $W_{*,*}$ matrices and b_* vectors are learned model parameters, and $u \circ v$ signifies element-wise multiplication.

2.3 Sentence-Level RNN Language Models

RNN sequence models have recently been shown to be extremely effective for word-level and character-level language models (Mikolov et al., 2011; Jozefowicz et al., 2016). At each timestep, these models take a word or character as input, update a hidden state vector, and predict the next timestep's word or character. There is also a growing body of work on training RNN encoder-decoder models for NLP problems. These systems first encode the entire input into the network's hidden state vector and then, in a second step, decode the entire output from this vector (Sutskever et al., 2014; Vinyals et al., 2015; Serban et al., 2016).

Sentence-level RNN language models, for example the skip-thought vector system of Kiros et al. (2015), conceptually bridge these two approaches. Whereas standard language models are trained to predict the next token in the sequence of tokens, these systems are explicitly trained to predict the next *sentence* in the sequence of sentences. Kiros et al. (2015) train an encoder-decoder model to encode a sentence into a fixed-length vector and subsequently decode both the following and preceding sentence, using Gated Recurrent Units (Chung et al., 2014). In the present work, we train an LSTM model to predict a sentence's successor, which is essentially the forward component of the skip-thought system. Kiros et al. (2015) use the skip-thought system as a means of projecting sentences into low-dimensional vector embeddings, demonstrating the utility of these embeddings on a number of other tasks; in contrast, we will use our trained sentence-level RNN language model directly on the task its objective function optimizes: predicting a sentence's successor.

3 Methodology

3.1 Narrative Cloze Evaluation

The evaluation of inference-focused statistical script systems is not straightforward. Chambers and Jurafsky (2008) introduced the *Narrative Cloze* evaluation, in which a single event is held out from a document and systems are judged by the ability to infer this held-out event given the remaining events. This evaluation has been used by a number of published script systems (Chambers and Jurafsky, 2009; Jans et al., 2012; Pichotta and Mooney, 2014; Rudinger et al., 2015). This automated evaluation measures systems' ability to model and predict events as they co-occur in text.

The exact definition of the Narrative Cloze evaluation depends on the formulation of events used in a script system. For example, Chambers and Jurafsky (2008), Jans et al. (2012), and Rudinger et al. (2015) evaluate inference of held-out (verb, dependency) pairs from documents; Pichotta and Mooney (2014) evaluate inference of

verbs with coreference information about multiple arguments; and Pichotta and Mooney (2016) evaluate inference of verbs with noun information about multiple arguments. In order to gather human judgments of inference quality, the latter also learn an encoder-decoder LSTM network for transforming verbs and noun arguments into English text to present to annotators for evaluation.

We evaluate instead on the task of directly inferring sequences of words. That is, instead of defining the Narrative Cloze to be the evaluation of predictions of held-out events, we define the task to be the evaluation of predictions of held-out text; in this setup, predictions need not be mediated by noisy, automatically-extracted events. To evaluate inferred text against gold standard text, we argue that the BLEU metric (Papineni et al., 2002), commonly used to evaluate Statistical Machine Translation systems, is a natural evaluation metric. It is an n-gram-level analog to the event-level Narrative Cloze evaluation: whereas the Narrative Cloze evaluates a system on its ability to reconstruct events as they occur in documents, BLEU evaluates a system on how well it reconstructs the n-grams.

This evaluation takes some inspiration from the evaluation of neural encoder-decoder translation models (Sutskever et al., 2014; Bahdanau et al., 2015), which use similar architectures for the task of Machine Translation. That is, the task we present can be thought of as "translating" a sentence into its successor. While we do not claim that BLEU is necessarily the optimal way of evaluating text-level inferences, but we do claim that it is a natural ngram-level analog to the Narrative Cloze task on events.

If a model infers text, we may also evaluate it on the task of inferring events by automatically extracting structured events from its output text (in the same way as events are extracted from natural text). This allows us to compare directly to previous event-based models on the task they are optimized for, namely, predicting structured events.

3.2 Models

Statistical script systems take a sequence of events from a document and infer additional events that are statistically probable. Exactly what constitutes an *event* varies: it may be a (verb, dependency) pair inferred as relating to a particular discourse entity (Chambers and Jurafsky, 2008; Rudinger et al., 2015), a simplex verb (Chambers and Jurafsky, 2009; Orr et al., 2014), or a verb with multiple arguments (Pichotta and Mooney, 2014). In the present work, we adopt a representation of events as verbs with multiple arguments (Balasubramanian et al., 2013; Pichotta and Mooney, 2014; Modi and Titov, 2014). Formally, we define an event to be a variadic tuple (v, s, o, p^*), where v is a verb, s is a noun standing in subject relation to v, o is a noun standing as a direct object to v, and p^* denotes an arbitrary number of (*pobj*, *prep*) pairs, with *prep* a preposition and *pobj* a noun related to the verb v via the preposition *prep*.[1] Any argument except v may be *null*, indicating no noun fills that slot. For example, the text

Napoleon sent the letter to Josephine

would be represented by the event (*sent, Napoleon, letter, (Josephine, to)*). We represent arguments by their grammatical head words.

We evaluate on a number of different neural models which differ in their input and output. All models are LSTM-based encoder-decoder models. These models encode a sentence (either its events or text) into a learned hidden vector state and then, subsequently, decode that vector into its successor sentence (either its events or its text).

Our general system architecture is as follows. At each timestep t, the input token is represented as a learned 100-dimensional embedding vector (learned jointly with the other parameters of the model), such that predictively similar words should get similar embeddings. This embedding is fed as input to the LSTM unit (that is, it will be the vector x_t in Section 2.2, the input to the LSTM). The output of the LSTM unit (called z_t in Section 2.2) is then fed to a softmax layer via a learned linear transformation.

During the encoding phase the network is not trained to produce any output. During the decoding phase the output is a one-hot representation of the subsequent timestep's input token (that is, with a V-word vocabulary, the output will be a V-dimensional vector with one 1 and $V - 1$ zeros). In this way, the network is trained to consume an entire input sequence and, as a second step, iteratively output the subsequent timestep's

[1] This is essentially the event representation of Pichotta and Mooney (2016), but whereas they limited events to having a single prepositional phrase, we allow an arbitrary number, and we do not lemmatize words.

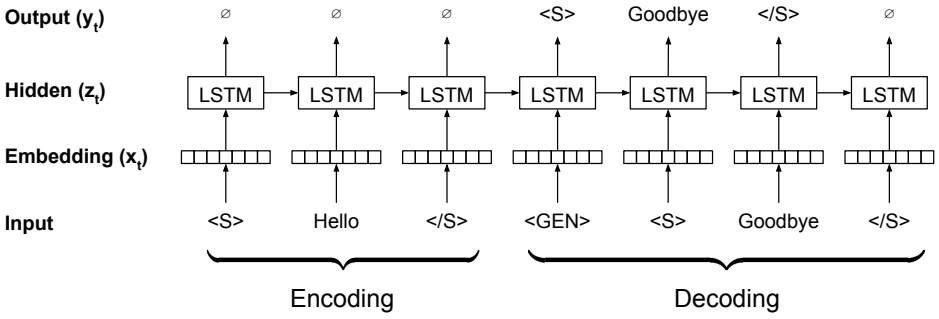


Figure 2: Encoder-Decoder setup predicting the text "Goodbye" from "Hello"

input, which allows the prediction of full output sequences. This setup is pictured diagrammatically in Figure 2, which gives an example of input and output sequence for a token-level encoder-decoder model, encoding the sentence "Hello ." and decoding the successor sentence "Goodbye ." Note that we add beginning-of-sequence and end-of-sequence pseudo-tokens to sentences. This formulation allows a system to be trained which can encode a sentence and then infer a successor sentence by iteratively outputting next-input predictions until the `</S>` end-of-sentence pseudo-token is predicted. We use different LSTMs for encoding and decoding, as the dynamics of the two stages need not be identical.

We notate the different systems as follows. Let s_1 be the input sentence and s_2 its successor sentence. Let t_1 denote the sequence of raw tokens in s_1, and t_2 the tokens of s_2. Further, let e_1 and e_2 be the sequence of structured events occurring in s_1 and s_2, respectively (described in more detail in Section 4.1), and let $e_2[0]$ denote the first event of e_2. The different systems we compare are named systematically as follows:

- The system $t_1 \rightarrow t_2$ is trained to encode a sentence's tokens and decode its successor's tokens.
- The system $e_1 \rightarrow e_2$ is trained to encode a sentence's events and decode its successor's events.
- The system $e_1 \rightarrow e_2 \rightarrow t_2$ is trained to encode a sentence's events, decode its successor's events, and then encode the latter and subsequently decode the successor's text.

We will not explicitly enumerate all systems, but other systems are defined analogously, with the schema $X \rightarrow Y$ describing a system which is trained to encode X and subsequently decode Y, and $X \rightarrow Y \rightarrow Z$ indicating a system which is trained to encode X, decode Y, and subsequently encode Y and decode Z. Note that in a system $X \rightarrow Y \rightarrow Z$, only X is provided as input.

We also present results for systems of the form $X \overset{a}{\rightarrow} Y$, which signifies that the system is trained to decode Y from X with the addition of an attention mechanism. We use the attention mechanism of Vinyals et al. (2015). In short, these models have additional parameters which can learn soft alignments between positions of encoded inputs and positions in decoded outputs. Attention mechanisms have recently been shown to be quite empirically valuable in many complex sequence prediction tasks. For more details on the model, see Vinyals et al. (2015).

Figure 3 gives a diagrammatic representation of the different system setups. Text systems infer successor text and, optionally, parse that text and extract events from it; event sequences infer successor events and, optionally, expand inferred events into text.

Note that the system $t_1 \rightarrow t_2$, in which both the encoding and decoding steps operate on raw text, is essentially a one-directional version of the skip-thought system of Kiros et al. (2015).[2] Further, the system $e_1 \rightarrow e_2 \rightarrow t_2$, which is trained to take a sentence's event sequence as input, predict its successor's events, and then predict its successor's words, is comparable to the event inference system of Pichotta and Mooney (2016). They use an LSTM sequence model of events in sequence

[2]The system of Kiros et al. (2015), in addition to being trained to predict the next sentence, also contains a backward-directional RNN trained to predict a sentence's predecessor; we condition only on previous text. Kiros et al. (2015) also use Gated Recurrent Units instead of LSTM.

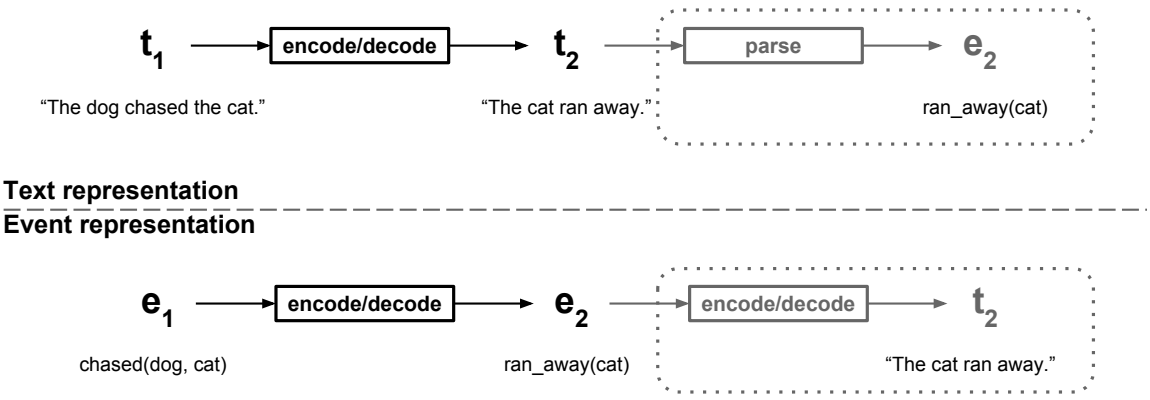


Figure 3: Different system setups for modeling the two-sentence sequence "The dog chased the cat." followed by "The cat ran away." The gray components inside dotted boxes are only present in some systems.

for event inference, and optionally transform inferred events to text using another LSTM; we, on the other hand, use an encoder/decoder setup to infer text directly.

4 Evaluation

4.1 Experimental Details

We train a number of LSTM encoder-decoder networks which vary in their input and output. Models are trained on English Language Wikipedia, with 1% of the documents held out as a validation set. Our test set consists of 10,000 unseen sentences (from articles in neither the training nor validation set). We train models with batch stochastic gradient descent with momentum, minimizing the cross-entropy error of output predictions. All models are implemented in TensorFlow (Abadi et al., 2015). We use a vocabulary of the 50,000 most frequent tokens, replacing all other tokens with an out-of-vocabulary pseudo-token. Learned word embeddings are 100-dimensional, and the latent LSTM vector is 500-dimensional. To extract events from text, we use the Stanford Dependency Parser (De Marneffe et al., 2006; Socher et al., 2013). We use the Moses toolkit (Koehn et al., 2007) to calculate BLEU.[3]

We evaluate the task of predicting held-out text with three metrics. The first metric is **BLEU**, which is standard BLEU (the geometric mean of modified 1-, 2-, 3-, and 4-gram precision against a gold standard, multiplied by a brevity penalty which penalizes short candidates). The second metric we present, **BLEU-BP,** is BLEU without the brevity penalty: in the task of predicting successor sentences, depending on predictions' end use, on-topic brevity is not necessarily undesirable. Evaluations are over top system inferences (that is, decoding is done by taking the argmax). Finally, we also present values for unigram precision (**1G P**), one of the components of BLEU.

We also evaluate on the task of predicting held-out verb-argument events, either directly or via inferred text. We use two evaluation metrics for this task. First, the **Accuracy** metric measures the percentage of a system's most confident guesses that are totally correct. That is, for each held-out event, a system makes its single most confident guess for that event, and we calculate the total percentage of such guesses which are totally correct. Some authors (e.g. Jans et al. (2012), Pichotta and Mooney (2016)) present results on the "Recall at k" metric, judging gold-standard recall against a list of top k event inferences; this metric is equivalent to "Recall at 1." This is quite a stringent metric, as an inference is only counted correct if the verb and all arguments are correct. To relax this requirement, we also present results on what we call the **Partial Credit** metric, which is the percentage of held-out event components identical to the respective components in a system's top inference.[4]

4.2 Experimental Evaluation

Table 1 gives the results of evaluating predicted successor sentence text against the gold standard using BLEU. The baseline system $t_1 \rightarrow t_1$ sim-

[3]Via the script `multi-bleu.pl`.

[4]This metric was used in Pichotta and Mooney (2014), but there it was called *Accuracy*. In the present work, we use "accuracy" only to mean Recall at 1.

System	BLEU	BLEU-BP	1G P
$t_1 \rightarrow t_1$	1.88	1.88	22.6
$e_1 \rightarrow e_2 \rightarrow t_2$	0.34	0.66	19.9
$e_1 \overset{a}{\rightarrow} e_2 \rightarrow t_2$	0.30	0.39	15.8
$t_1 \rightarrow t_2$	**5.20**	7.84	30.9
$t_1 \overset{a}{\rightarrow} t_2$	4.68	**8.09**	**32.2**

Table 1: Successor text predictions evaluated with BLEU.

System	Accuracy	Partial Credit
Most common	0.2	26.5
$e_1 \rightarrow e_2[0]$	**2.3**	26.7
$e_1 \overset{a}{\rightarrow} e_2[0]$	2.2	25.6
$t_1 \rightarrow t_2 \rightarrow e_2[0]$	2.0	**30.3**
$t_1 \overset{a}{\rightarrow} t_2 \rightarrow e_2[0]$	2.0	27.7

Table 2: Next event prediction accuracy (numbers are percentages: maximum value is 100).

ply reproduces the input sentence as its own successor.[5] Below this are systems which make predictions from event information, with systems which make predictions from raw text underneath. Transformations written $X \overset{a}{\rightarrow} Y$ are, recall, encoder-decoder LSTMs with attention.

Note, first, that the text-level models outperform other models on BLEU. In particular, the two-step model $e_1 \rightarrow e_2 \rightarrow t_2$ (and comparable model with attention) which first predicts successor events and then, as a separate step, expands these events into text, performs quite poorly. This is perhaps due to the fact that the translation from text to events is lossy, so reconstructing raw sentence tokens is not straightforward.

The BLEU-BP scores, which are BLEU without the brevity penalty, are noticeably higher in the text-level models than the raw BLEU scores. This is in part because these models seem to produce shorter sentences, as illustrated below in section 4.4.

The attention mechanism does not obviously benefit either text or event level prediction encoder-decoder models. This could be because there is not an obvious alignment structure between contiguous spans of raw text (or events) in natural documents.

These results provide evidence that, if the Narrative Cloze task is defined to evaluate prediction of held-out text from a document, then sentence-level RNN language models provide superior performance to RNN models operating at the event level. In other words, linguistic pre-processing does not obviously benefit encoder-decoder models trained to predict succeeding text.

Table 2 gives results on the task of predicting the next verb with its nominal arguments; that is, whereas Table 1 gave results on a text analog to the Narrative Cloze evaluation (BLEU), Table 2 gives results on the verb-with-arguments prediction version. In the $t_1 \rightarrow t_2 \rightarrow e_2[0]$ system (and the comparable system with attention), events are extracted from automatically generated text by parsing output text and applying the same event extractor to this parse used to extract events from raw text.[6] The row labeled **Most common** in Table 2 gives performance for the baseline system which always guesses the most common event in the training set.

The LSTM models trained to directly predict events are roughly comparable to systems which operate on raw text, performing slightly worse on accuracy and slightly better when taking partial credit into account. As with the previous comparisons with BLEU, the attention mechanism does not provide an obvious improvement when decoding inferences, perhaps, again, because the event inference problem lacks a clear alignment structure.

These systems infer their most probable guesses of $e_2[0]$, the first event in the succeeding sentence. In order for a system prediction to be counted as correct, it must have the correct strings for grammatical head words of all components of the correct event. Note also that we judge only against a system's single most confident prediction (as opposed to some prior work (Jans et al., 2012; Pichotta and Mooney, 2014) which takes the top k predictions—the numbers presented here are therefore noticeably lower). We do this mainly for computational reasons: namely, a beam search over a full sentence's text would be quite computationally expensive.

4.3 Adding Additional Context

The results given above are for systems which encode information about one sentence and decode

[5] "$t_1 \rightarrow t_1$" is minor abuse of notation, as the system is not an encoder/decoder but a simple identity function.

[6] This is also a minor abuse of notation, as the second transformation uses a statistical parser rather than an encoder/decoder.

information about its successor. This is within the spirit of the skip-gram system of Kiros et al. (2015), but we may wish to condition on more of the document. To investigate this, we perform an experiment varying the number of previous sentences input during the encoding step of $t_1 \rightarrow t_2$ text-level models without attention. We train three different models, which take either one, three, or five sentences as input, respectively, and are trained to output the successor sentence.

Num Prev Sents	BLEU	BLEU-BP	1G P
1	5.80	8.59	29.4
3	5.82	**9.35**	**31.2**
5	**6.83**	6.83	21.4

Table 3: Varying the amount of context in text-level models. "Num Prev Sents" is the number of previous sentences supplied during encoding.

Table 3 gives the results of running these models on 10,000 sentences from the validation set. As can be seen, in the training setup we investigate, more additional context sentences have a mixed effect, depending on the metric. This is perhaps due in part to the fact that we kept hyperparameters fixed between experiments, and a different hyperparameter regime would benefit predictions from longer input sequences. More investigation could prove fruitful.

4.4 Qualitative Analysis

Figure 4 gives some example automatic next-sentence text predictions, along with the input sentence and the gold-standard next sentence. Note that gold-standard successor sentences frequently introduce new details not obviously inferrable from previous text. Top system predictions, on the other hand, are frequently fairly short. This is likely due part to the fact that the cross-entropy loss does not directly penalize short sentences and part to the fact that many details in gold-standard successor text are inherently difficult to predict.

4.5 Discussion

The general low magnitude of the BLEU scores presented in Table 1, especially in comparison to the scores typically reported in Machine Translation results, indicates the difficulty of the task. In open-domain text, a sentence is typically not straightforwardly predictable from preceding text; if it were, it would likely not be stated.

On the task of verb-argument prediction in Table 2, the difference between $t_1 \rightarrow t_2$ and $e_1 \rightarrow e_2[0]$ is fairly marginal. This raises the general question of how much explicit syntactic analysis is required for the task of event inference, particularly in the encoder/decoder setup. These results provide evidence that a sentence-level RNN language model which operates on raw tokens can predict what comes next in a document as well or nearly as well as an event-mediated script model.

5 Future Work

There are a number of further extensions to this work. First, in this work (and, more generally, Neural Machine Translation research), though generated text is evaluated using BLEU, systems are optimized for per-token cross-entropy error, which is a different objective (Luong et al. (2016) give an example of a system which improves cross-entropy error but reduces BLEU score in the Neural Machine Translation context). Finding differentiable objective functions that more directly target more complex evaluation metrics like BLEU is an interesting future research direction.

Relatedly, though we argue that BLEU is a natural token-sequence-level analog to the verb-argument formulation of the Narrative Cloze task, it is not obviously the best metric for evaluating inferences of text, and comparing these automated metrics with human judgments is an important direction of future work. Pichotta and Mooney (2016) present results on crowdsourced human evaluation of script inferences that could be repeated for our RNN models.

Though we focus here on forward-direction models predicting successor sentences, bidirectional encoder-decoder models, which predict sentences from both previous and subsequent text, are another interesting future research direction.

6 Related Work

The use of scripts in AI dates back to the 1970s (Minsky, 1974; Schank and Abelson, 1977); in this conception, scripts were composed of complex events with no probabilistic semantics, which were difficult to learn automatically. In recent years, a growing body of research has investigated learning probabilistic co-occurrence models with simpler events. Chambers and Jurafsky (2008) propose a model of co-occurrence of (verb, dependency) pairs, which can be used to infer such

Input:	As of October 1 , 2008 , ⟨OOV⟩ changed its company name to Panasonic Corporation.
Gold:	⟨OOV⟩ products that were branded "National" in Japan are currently marketed under the "Panasonic" brand.
Predicted:	The company's name is now ⟨OOV⟩.
Input:	White died two days after Curly Bill shot him.
Gold:	Before dying, White testified that he thought the pistol had accidentally discharged and that he did not believe that Curly Bill shot him on purpose.
Predicted:	He was buried at ⟨OOV⟩ Cemetery.
Input:	The foundation stone was laid in 1867.
Gold:	The members of the predominantly Irish working class parish managed to save £700 towards construction, a large sum at the time.
Predicted:	The ⟨OOV⟩ was founded in the early 20th century.
Input:	Soldiers arrive to tell him that ⟨OOV⟩ has been seen in camp and they call for his capture and death.
Gold:	⟨OOV⟩ agrees .
Predicted:	⟨OOV⟩ is killed by the ⟨OOV⟩.

Figure 4: Sample next-sentence text predictions. ⟨OOV⟩ is the out-of-vocabulary pseudo-token, which frequently replaces proper names.

pairs from documents; Jans et al. (2012) give a superior model in the same general framework. Chambers and Jurafsky (2009) give a method of generalizing from single sequences of pair events to collections of such sequences. Rudinger et al. (2015) apply a discriminative language model to the (verb, dependency) sequence modeling task, raising the question of to what extent event inference can be performed with standard language models applied to event sequences. Pichotta and Mooney (2014) describe a method of learning a co-occurrence based model of verbs with multiple coreference-based entity arguments.

There is a body of related work focused on learning models of co-occurring events to automatically induce templates of complex events comprising multiple verbs and arguments, aimed ultimately at maximizing coherency of templates (Chambers, 2013; Cheung et al., 2013; Balasubramanian et al., 2013). Ferraro and Van Durme (2016) give a model integrating various levels of event information of increasing abstraction, evaluating both on coherence of induced templates and log-likelihood of predictions of held-out events. McIntyre and Lapata (2010) describe a system that learns a model of co-occurring events and uses this model to automatically generate stories via a Genetic Algorithm.

There have been a number of recent published neural models for various event- and discourse-related tasks. Pichotta and Mooney (2016) show that an LSTM event sequence model outperforms previous co-occurrence methods for predicting verbs with arguments. Granroth-Wilding and Clark (2016) describe a feedforward neural network which composes verbs and arguments into low-dimensional vectors, evaluating on a multiple-choice version of the Narrative Cloze task. Modi and Titov (2014) describe a feedforward network which is trained to predict event orderings. Kiros et al. (2015) give a method of embedding sentences in low-dimensional space such that embeddings are predictive of neighboring sentences. Li et al. (2014) and Ji and Eisenstein (2015), use RNNs for discourse parsing; Liu et al. (2016) use a Convolutional Neural Network for implicit discourse relation classification.

7 Conclusion

We have given what we believe to be the first systematic evaluation of sentence-level RNN language models on the task of predicting held-out document text. We have found that models operating on raw text perform roughly comparably to identical models operating on predicate-argument event structures when predicting the latter, and that text models provide superior predictions of raw text. This provides evidence that, for the task of held-out event prediction, encoder/decoder models mediated by automatically extracted events may not be learning appreciably more structure than systems trained on raw tokens alone.

Acknowledgments

Thanks to Stephen Roller, Amelia Harrison, and the UT NLP group for their help and feedback. Thanks also to the anonymous reviewers for their very helpful suggestions. This research was supported in part by the DARPA DEFT program under AFRL grant FA8750-13-2-0026.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR 2015)*.

Niranjan Balasubramanian, Stephen Soderland, Mausam, and Oren Etzioni. 2013. Generating coherent event schemas at scale. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*.

Nathanael Chambers and Daniel Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 789–797.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-09)*, pages 602–610.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP-2013)*.

Jackie Chi Kit Cheung, Hoifung Poon, and Lucy Vanderwende. 2013. Probabilistic frame induction. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-13)*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Deep Learning Workshop*.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources & Evaluation (LREC-2006)*, volume 6, pages 449–454.

Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14:179–211.

Francis Ferraro and Benjamin Van Durme. 2016. A unified Bayesian model of scripts, frames and language. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Mark Granroth-Wilding and Stephen Clark. 2016. What happens next? Event prediction using a compositional neural network model. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*.

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

Bram Jans, Steven Bethard, Ivan Vulić, and Marie Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL-12)*, pages 336–344.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributional semantics for discourse relations. *Transactions of the Association for Computational Linguistics (TACL)*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07) Companion Volume: Proceedings of the Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic.

Jiwei Li, Rumeng Li, and Eduard Hovy. 2014. Recursive deep models for discourse parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2061–2069, October.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Minh-Thang Luong, Quoc V. Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2016. Multi-task sequence to sequence learning. In *Proceedings of the 4th International Conference on Learning Representations (ICLR-16)*.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1562–1572.

Tomas Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Cernockỳ. 2011. Empirical evaluation and combination of advanced language modeling techniques. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association 2011 (INTERSPEECH 2011)*, pages 605–608.

Marvin Minsky. 1974. A framework for representing knowledge. Technical report, MIT-AI Laboratory.

Ashutosh Modi and Ivan Titov. 2014. Inducing neural models of script knowledge. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning (CoNLL-2014)*, Baltimore, MD, USA.

J Walker Orr, Prasad Tadepalli, Janardhan Rao Doppa, Xiaoli Fern, and Thomas G Dietterich. 2014. Learning scripts as Hidden Markov Models. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 311–318.

Karl Pichotta and Raymond J. Mooney. 2014. Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 220–229.

Karl Pichotta and Raymond J. Mooney. 2016. Learning statistical scripts with LSTM recurrent neural networks. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Rachel Rudinger, Pushpendre Rastogi, Francis Ferraro, and Benjamin Van Durme. 2015. Script induction as language modeling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP-15)*.

Roger C. Schank and Robert P. Abelson. 1977. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum and Associates.

Iulian V. Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*.

Richard Socher, John Bauer, Christopher D. Manning, and Andrew Y. Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL-13)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NIPS-14)*, pages 3104–3112.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NIPS-15)*, pages 2755–2763.

A Supplemental Material

Our Wikipedia dump from which the training, development, and test sets are constructed is from Jan 2, 2014. We parse text using version 3.3.1 of the Stanford CoreNLP system. We use a vocab consisting of the 50,000 most common tokens, replacing all others with an Out-of-vocabulary pseudotoken. We train using batch stochastic gradient descent with momentum with a batch size of 10 sequences, using an initial learning rate of 0.1, damping the learning rate by 0.99 any time the previous hundred updates' average test error is greater than any of the average losses in the previous ten groups of hundred updates. Our momentum parameter is 0.95. Our embedding vectors are 100-dimensional, and our LSTM hidden state is 500-dimensional. We train all models for 300k batch updates (with the exception of the models compared in §4.3, all of which we train for 150k batch updates, as training is appreciably slower with longer input sequences). Training takes approximately 36 hours on an NVIDIA Titan Black GPU.

Two Discourse Driven Language Models for Semantics

Haoruo Peng and **Dan Roth**
University of Illinois, Urbana-Champaign
Urbana, IL, 61801
{hpeng7,danr}@illinois.edu

Abstract

Natural language understanding often requires deep semantic knowledge. Expanding on previous proposals, we suggest that some important aspects of semantic knowledge can be modeled as a language model if done at an appropriate level of abstraction. We develop two distinct models that capture semantic frame chains and discourse information while abstracting over the specific mentions of predicates and entities. For each model, we investigate four implementations: a "standard" N-gram language model and three discriminatively trained "neural" language models that generate embeddings for semantic frames. The quality of the semantic language models (SemLM) is evaluated both intrinsically, using perplexity and a narrative cloze test and extrinsically – we show that our SemLM helps improve performance on semantic natural language processing tasks such as co-reference resolution and discourse parsing.

1 Introduction

Natural language understanding often necessitates deep semantic knowledge. This knowledge needs to be captured at multiple levels, from words to phrases, to sentences, to larger units of discourse. At each level, capturing meaning frequently requires context sensitive abstraction and disambiguation, as shown in the following example (Winograd, 1972):

Ex.1 *[Kevin] was **robbed** by [Robert]. [He] was **arrested** by the police.*

Ex.2 *[Kevin] was **robbed** by [Robert]. [He] was **rescued** by the police.*

In both cases, one needs to resolve the pronoun "he" to either "Robert" or "Kevin". To make the correct decisions, one needs to know that the subject of "rob" is more likely than the object of "rob" to be the object of "arrest" while the object of "rob" is more likely to be the object of "rescue". Thus, beyond understanding individual predicates (e.g., at the semantic role labeling level), there is a need to place them and their arguments in a global context.

However, just modeling semantic frames is not sufficient; consider a variation of Ex.1:

Ex.3 *Kevin was **robbed** by Robert, but the police mistakenly **arrested** him.*

In this case, "him" should refer to "Kevin" as the discourse marker "but" reverses the meaning, illustrating that it is necessary to take discourse markers into account when modeling semantics.

In this paper we propose that these aspects of semantic knowledge can be modeled as a *Semantic Language Model* (SemLM). Just like the "standard" syntactic language models (LM), we define a basic vocabulary, a finite representation language, and a prediction task, which allows us to model the distribution over the occurrence of elements in the vocabulary as a function of their (well-defined) context. In difference from syntactic LMs, we represent natural language at a higher level of semantic abstraction, thus facilitating modeling deep semantic knowledge.

We propose two distinct discourse driven language models to capture semantics. In our first semantic language model, the *Frame-Chain SemLM*, we model all semantic frames and discourse markers in the text. Each document is viewed as a single chain of semantic frames and discourse markers. Moreover, while the vocabulary of discourse markers is rather small, the number of different surface form semantic frames that could appear in the text is very large. To achieve a better level of abstraction, we disambiguate semantic frames and map them to their PropBank/FrameNet represen-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 290–300,
Berlin, Germany, August 7-12, 2016.

tation. Thus, in Ex.3, the resulting frame chain is "rob.01 — but — arrest.01" ("01" indicates the predicate sense).

Our second semantic language model is called *Entity-Centered SemLM*. Here, we model a sequence of semantic frames and discourse markers involved in a specific co-reference chain. For each co-reference chain in a document, we first extract semantic frames corresponding to each co-referent mention, disambiguate them as before, and then determine the discourse markers between these frames. Thus, each unique frame contains both the disambiguated predicate and the argument label of the mention. In Ex.3, the resulting sequence is "rob.01#obj — but — arrest.01#obj" (here "obj" indicates the argument label for "Kevin" and "him" respectively). While these two models capture somewhat different semantic knowledge, we argue later in the paper that both models can be induced at high quality, and that they are suitable for different NLP tasks.

For both models of SemLM, we study four language model implementations: N-gram, skip-gram (Mikolov et al., 2013b), continuous bag-of-words (Mikolov et al., 2013a) and log-bilinear language model (Mnih and Hinton, 2007). Each model defines its own prediction task. In total, we produce eight different SemLMs. Except for N-gram model, others yield embeddings for semantic frames as they are neural language models.

In our empirical study, we evaluate both the quality of all SemLMs and their application to co-reference resolution and shallow discourse parsing tasks. Following the traditional evaluation standard of language models, we first use perplexity as our metric. We also follow the script learning literature (Chambers and Jurafsky, 2008b; Chambers and Jurafsky, 2009; Rudinger et al., 2015) and evaluate on the narrative cloze test, i.e. randomly removing a token from a sequence and test the system's ability to recover it. We conduct both evaluations on two test sets: a hold-out dataset from the New York Times Corpus and gold sequence data (for frame-chain SemLMs, we use PropBank (Kingsbury and Palmer, 2002); for entity-centered SemLMs, we use Ontonotes (Hovy et al., 2006)). By comparing the results on these test sets, we show that we do not incur noticeable degradation when building SemLMs using preprocessing tools. Moreover, we show that SemLMs improves the performance of co-reference resolution, as well as that of predicting the sense of discourse connectives for both explicit and implicit ones.

The main contributions of our work can be summarized as follows: 1) The design of two novel discourse driven Semantic Language models, building on text abstraction and neural embeddings; 2) The implementation of high quality SemLMs that are shown to improve state-of-the-art NLP systems.

2 Related Work

Our work is related to script learning. Early works (Schank and Abelson, 1977; Mooney and DeJong, 1985) tried to construct knowledge bases from documents to learn scripts. Recent work focused on utilizing statistical models to extract high-quality scripts from large amounts of data (Chambers and Jurafsky, 2008a; Bejan, 2008; Jans et al., 2012; Pichotta and Mooney, 2014; Granroth-Wilding et al., 2015; Pichotta and Mooney, 2016). Other works aimed at learning a collection of structured events (Chambers, 2013; Cheung et al., 2013; Cheung et al., 2013; Balasubramanian et al., 2013; Bamman and Smith, 2014; Nguyen et al., 2015), and several works have employed neural embeddings (Modi and Titov, 2014b; Modi and Titov, 2014a; Frermann et al., 2014; Titov and Khoddam, 2015).

In our work, the semantic sequences in the entity-centered SemLMs are similar to narrative schemas (Chambers and Jurafsky, 2009). However, we differ from them in the following aspects: 1) script learning does not generate a probabilistic model on semantic frames[1]; 2) script learning models semantic frame sequences incompletely as they do not consider discourse information; 3) works in script learning rarely show applications to real NLP tasks.

Some prior works have used scripts-related ideas to help improve NLP tasks (Irwin et al., 2011; Rahman and Ng, 2011; Peng et al., 2015b). However, since they use explicit script schemas either as features or constraints, these works suffer from data sparsity problems. In our work, the SemLM abstract vocabulary ensures a good coverage of frame semantics.

[1]Some works may utilize a certain probabilistic framework, but they mainly focus on generating high-quality frames by filtering.

Table 1: **Comparison of vocabularies between frame-chain (FC) and entity-centered (EC) SemLMs.** "F-Sen" stands for frames with predicate sense information while "F-Arg" stands for frames with argument role label information; "Conn" means discourse marker and "Per" means period. "Seq/Doc" represents the number of sequence per document.

	F-Sen	F-Arg	Conn	Per	Seq/Doc
FC	YES	NO	YES	YES	Single
EC	YES	YES	YES	NO	Multiple

3 Two Models for SemLM

In this section, we describe how we capture sequential semantic information consisted of semantic frames and discourse markers as semantic units (i.e. the vocabulary).

3.1 Semantic Frames and Discourse Markers

Semantic Frames A semantic frame is composed of a predicate and its corresponding argument participants. Here we require the predicate to be disambiguated to a specific sense, and we need a certain level of abstraction of arguments so that we can assign abstract labels. The design of PropBank frames (Kingsbury and Palmer, 2002) and FrameNet frames (Baker et al., 1998) perfectly fits our needs. They both have a limited set of frames (in the scale of thousands) and each frame can be uniquely represented by its predicate sense. These frames provide a good level of generalization as each frame can be instantiated into various surface forms in natural texts. We use these frames as part of our vocabulary for SemLMs. Formally, we use the notation f to represent a frame. Also, we denote fa $\triangleq$ f#Arg when referring to an argument role label (Arg) inside a frame (f).

Discourse Markers We use discourse markers (connectives) to model discourse relationships between frames. There is only a limited number of unique discourse markers, such as *and, but, however*, etc. We get the full list from the Penn Discourse Treebank (Prasad et al., 2008) and include them as part of our vocabulary for SemLMs. Formally, we use *dis* to denote the discourse marker. Note that discourse relationships can exist without an explicit discourse marker, which is also a challenge for discourse parsing. Since we cannot reliably identify implicit discourse relationships, we only consider explicit ones here. More importantly, discourse markers are associated with arguments (Wellner and Pustejovsky, 2007) in text (usually two sentences/clauses, sometimes one). We only add a discourse marker in the semantic sequence when its corresponding arguments contain semantic frames which belong to the same semantic sequence. We call them *frame-related discourse markers*. Details on generating semantic frames and discourse markers to form semantic sequences are discussed in Sec. 5.

3.2 Frame-Chain SemLM

For frame-chain SemLM, we model all semantic frames and discourse markers in a document. We form the semantic sequence by first including all semantic frames in the order they appear in the text: $[\mathrm{f}_1, \mathrm{f}_2, \mathrm{f}_3, \ldots]$. Then we add *frame-related discourse markers* into the sequence by placing them in their order of appearance. Thus we get a sequence like $[\mathrm{f}_1, \mathrm{dis}_1, \mathrm{f}_2, \mathrm{f}_3, \mathrm{dis}_2, \ldots]$. Note that discourse markers do not necessarily exist between all semantic frames. Additionally, we treat the *period* symbol as a special discourse marker, denoted by "o". As some sentences contain more than one semantic frame (situations like clauses), we get the final semantic sequence like this:

$$[\mathrm{f}_1, \mathrm{dis}_1, \mathrm{f}_2, \mathrm{o}, \mathrm{f}_3, \mathrm{o}, \mathrm{dis}_2, \ldots, \mathrm{o}]$$

3.3 Entity-Centered SemLM

We generate semantic sequences according to co-reference chains for entity-centered SemLM. From co-reference resolution, we can get a sequence like $[\mathrm{m}_1, \mathrm{m}_2, \mathrm{m}_3, \ldots]$, where mentions appear in the order they occur in the text. Each mention can be matched to an argument inside a semantic frame. Thus, we replace each mention with its argument label inside a semantic frame, and get $[\mathrm{fa}_1, \mathrm{fa}_2, \mathrm{fa}_3, \ldots]$. We then add discourse markers exactly in they way we do for frame-chain SemLM, and get the following sequence:

$$[\mathrm{fa}_1, \mathrm{dis}_1, \mathrm{fa}_2, \mathrm{fa}_3, \mathrm{dis}_2, \ldots]$$

The comparison of vocabularies between frame-chain and entity-centered SemLMs is summarized in Table 1.

4 Implementations of SemLM

In this work, we experiment with four language model implementations: N-gram (NG), Skip-Gram (SG), Continuous Bag-of-Words (CBOW) and Log-bilinear (LB) language model. For ease

of explanation, we assume that a semantic unit sequence is $s = [w_1, w_2, w_3, \ldots, w_k]$.

4.1 N-gram Model

For an n-gram model, we predict each token based on its $n-1$ previous tokens, i.e. we directly model the following conditional probability (in practice, we choose $n = 3$, Tri-gram (TRI)):

$$p(w_{t+2}|w_t, w_{t+1}).$$

Then, the probability of the sequence is

$$p(s) = p(w_1)p(w_2|w_1)\prod_{t=1}^{k-2} p(w_{t+2}|w_t, w_{t+1}).$$

To compute $p(w_2|w_1)$ and $p(w_1)$, we need to back off from Tri-gram to Bi-gram and Uni-gram.

4.2 Skip-Gram Model

The SG model was proposed in Mikolov et al. (2013b). It uses a token to predict its context, i.e. we model the following conditional probability:

$$p(c \in c(w_t)|w_t, \theta).$$

Here, $c(w_t)$ is the context for w_t and θ denotes the learned parameters which include neural network states and embeddings. Then the probability of the sequence is computed as

$$\prod_{t=1}^{k} \prod_{c \in c(w_t)} p(c|w_t, \theta).$$

4.3 Continuous Bag-of-Words Model

In contrast to skip-gram, CBOW (Mikolov et al., 2013a) uses context to predict each token, i.e. we model the following conditional probability:

$$p(w_t|c(w_t), \theta).$$

In this case, the probability of the sequence is

$$\prod_{t=1}^{k} p(w_t|c(w_t), \theta).$$

4.4 Log-bilinear Model

LB was introduced in Mnih and Hinton (2007). Similar to CBOW, it also uses context to predict each token. However, LB associates a token with three components instead of just one vector: a target vector v(w), a context vector v'(w) and a bias b(w). So, the conditional probability becomes:

$$p(w_t|c(w_t)) = \frac{\exp(v(w_t)^\intercal u(c(w_t)) + b(w_t))}{\sum_{w \in \mathcal{V}} \exp(v(w)^\intercal u(c(w_t)) + b(w))}.$$

Here, $\mathcal{V}$ denotes the vocabulary and we define $u(c(w_t)) = \sum_{c_i \in c(w_t)} q_i \odot v'(c_i)$. Note that $\odot$ represents element-wise multiplication and q_i is a vector that depends only on the position of a token in the context, which is a also a model parameter.

So, the overall sequence probability is

$$\prod_{t=1}^{k} p(w_t|c(w_t)).$$

5 Building SemLMs from Scratch

In this section, we explain how we build SemLMs from un-annotated plain text.

5.1 Dataset and Preprocessing

Dataset We use the New York Times Corpus[2] (from year 1987 to 2007) for training. It contains a bit more than 1.8M documents in total.

Preprocessing We pre-process all documents with semantic role labeling (Punyakanok et al., 2004) and part-of-speech tagger (Roth and Zelenko, 1998). We also implement the explicit discourse connective identification module in shallow discourse parsing (Song et al., 2015). Additionally, we utilize within document entity coreference (Peng et al., 2015a) to produce coreference chains. To obtain all annotations, we employ the Illinois NLP tools[3].

5.2 Semantic Unit Generation

FrameNet Mapping We first directly derive semantic frames from semantic role labeling annotations. As the Illinois SRL package is built upon PropBank frames, we do a mapping to FrameNet frames via VerbNet senses (Schuler, 2005), thus achieving a higher level of abstraction. The mapping file[4] defines deterministic mappings. However, the mapping is not complete and there are remaining PropBank frames. Thus, the generated vocabulary for SemLMs contains both PropBank and FrameNet frames. For example, "place" and

[2]https://catalog.ldc.upenn.edu/LDC2008T19

[3]http://cogcomp.cs.illinois.edu/page/software/

[4]http://verbs.colorado.edu/verb-index/fn/vn-fn.xml

"put" with the VerbNet sense id "9.1-2" are converted to the same FrameNet frame "Placing".

Augmenting to Verb Phrases We apply three heuristic modifications to augment semantic frames defined in Sec. 3.1: 1) if a preposition immediately follows a predicate, we append the preposition to the predicate e.g. "take over"; 2) if we encounter the semantic role label AM-PRD which indicates a secondary predicate, we also append this secondary predicate to the main predicate e.g. "be happy"; 3) if we see the semantic role label AM-NEG which indicates negation, we append "not" to the predicate e.g. "not like". These three augmentations can co-exist and they allow us to model more fine-grained semantic frames.

Verb Compounds We have observed that if two predicates appear very close to each other, e.g. "eat and drink", "decide to buy", they actually represent a unified semantic meaning. Thus, we construct compound verbs to connect them together. We apply the rule that if the gap between two predicates is less than two tokens, we treat them as a unified semantic frame defined by the conjunction of the two (augmented) semantic frames, e.g. "eat.01-drink.01" and "decide.01-buy.01".

Argument Labels for Co-referent Mentions To get the argument role label information for co-referent mentions, we need to match each mention to its corresponding semantic role labeling argument. If a mention head is inside an argument, we regard it as a match. We do not consider singleton mentions.

Vocabulary Construction After generating all semantic units for (augmented and compounded) semantic frames and discourse markers, we merge them together as a tentative vocabulary. In order to generate a sensible SemLM, we filter out rare tokens which appear less than 20 times in the data. We add the Unknown token (UNK) and End-of-Sequence token (EOS) to the eventual vocabulary.

Statistics on the eventual SemLM vocabularies and semantic sequences are shown in Table 2. We also compare frame-chain and entity-centered SemLMs to the usual syntactic language model setting. The statistics in Table 2 shows that they are comparable both in vocabulary size and in the total number of tokens for training. Moreover, entity-centered SemLMs have shorter sequences then frame-chain SemLMs. We also provide several examples of high-frequency augmented compound semantic frames in our generated SemLM vocabularies. All are very intuitive:

want.01-know.01, agree.01-pay.01, try.01-get.01, decline.02-comment.01, wait.01-see.01, make.02-feel.01, want.01(not)-give.08(up)

Table 2: **Statistics on SemLM vocabularies and sequences.** "F-s" stands for single frame while "F-c" stands for compound frame; "Conn" means discourse marker. "#seq" is the number of sequences, and "#token" is the total number of tokens (semantic units). We also compute the average token in a sequence i.e. "#t/s". We compare frame-chain (FC) and entity-centered (EC) SemLMs to the usual syntactic language model setting i.e. "LM".

	Vocabulary Size			Sequence Size		
	F-s	F-c	Conn	#seq	#token	#t/s
FC	14857	7269	44	1.2M	25.4M	21
EC	8758	2896	44	3.4M	18.6M	5
LM	∼20k			∼3M	∼38M	10-15

5.3 Language Model Training

NG We implement the N-gram model using the SRILM toolkit (Stolcke, 2002). We also employ the well-known KneserNey Smoothing (Kneser and Ney, 1995) technique.

SG & CBOW We utilize the word2vec package to implement both SG and CBOW. In practice, we set the context window size to be 10 for SG while set the number as 5 for CBOW (both are usual settings for syntactic language models). We generate 300-dimension embeddings for both models.

LB We use the OxLM toolkit (Paul et al., 2014) with Noise-Constrastive Estimation (Gutmann and Hyvarinen, 2010) for the LB model. We set the context window size to 5 and produce 150-dimension embeddings.

6 Evaluation

In this section, we first evaluate the quality of SemLMs through perplexity and a narrative cloze test. More importantly, we show that the proposed SemLMs can help improve the performance of co-reference resolution and shallow discourse parsing. This further proves that we successfully capture semantic sequence information which can potentially benefit a wide range of semantic related NLP tasks.

We have designed two models for SemLM: *frame-chain* (**FC**) and *entity-centered* (**EC**). By training on both types of sequences respectively, we implement four different language models:

TRI, SG, CBOW, LB. We focus the evaluation efforts on these eight SemLMs.

6.1 Quality Evaluation of SemLMs

Datasets We use three datasets. We first randomly sample 10% of the New York Times Corpus documents (roughly two years of data), denoted the *NYT Hold-out Data*. All our SemLMs are trained on the remaining NYT data and tested on this hold-out data. We generate semantic sequences for the training and test data using the methodology described in Sec. 5.

We use PropBank data with gold frame annotations as another test set. In this case, we only generate frame-chain SemLM sequences by applying semantic unit generation techniques on gold frames, as described in Sec 5.2. When we test on *Gold PropBank Data with Frame Chains*, we use frame-chain SemLMs trained from all NYT data.

Similarly, we use Ontonotes data (Hovy et al., 2006) with gold frame and co-reference annotations as the third test set, *Gold Ontonotes Data with Coref Chains*. We only generate entity-centered SemLMs by applying semantic unit generation techniques on gold frames and gold co-reference chains, as described in Sec 5.2.

Baselines We use Uni-gram (**UNI**) and Bi-gram (**BG**) as two language model baselines. In addition, we use the point-wise mutual information (PMI) for token prediction. Essentially, PMI scores each pair of tokens according to their co-occurrences. It predicts a token in the sequence by choosing the one with the highest total PMI with all other tokens in the sequence. We use the ordered PMI (**OP**) as our baseline, which is a variation of PMI by considering asymmetric counting (Jans et al., 2012).

6.1.1 Perplexity

As SemLMs are language models, it is natural to evaluate the perplexity, which is a measurement of how well a language model can predict sequences.

Results for SemLM perplexities are presented in Table 3. They are computed without considering end token (EOS). We apply tri-gram Kneser-Ney Smoothing to CBOW, SG and LB. LB consistently shows the lowest perplexities for both frame-chain and entity-centered SemLMs across all test sets. Similar to syntactic language models, perplexities are fast decreasing from UNI, BI to TRI. Also, CBOW and SG have very close perplexity results which indicate that their language modeling abilities are at the same level.

Table 3: **Perplexities for SemLMs.** UNI, BG, TRI, CBOW, SG, LB are different language model implementations while "FC" and "EC" stand for the two SemLM models studied, respectively. "FC-FM" and "EC-FM" indicate that we removed the "FrameNet Mapping" step (Sec. 5.2). LB consistently produces the lowest perplexities for both frame-chain and entity-centered SemLMs.

	Baselines		SemLMs			
	UNI	BG	TRI	CBOW	SG	LB
NYT Hold-out Data						
FC	952.1	178.3	119.2	115.4	114.1	**108.5**
EC	914.7	154.4	114.9	111.8	113.8	**109.7**
Gold PropBank Data with Frame Chains						
FC-FM	992.9	213.7	139.1	135.6	128.4	121.8
FC	970.0	191.2	132.7	126.4	123.5	**115.4**
Gold Ontonotes Data with Coref Chains						
EC-FM	956.4	187.7	121.1	115.6	117.2	113.7
EC	923.8	163.2	120.5	113.7	115.0	**109.3**

We can compare the results of our frame-chain SemLM on *NYT Hold-out Data* and *Gold PropBank Data with Frame Chains*, and our entity-centered SemLM on *NYT Hold-out Data* and *Gold Ontonotes Data with Coref Chains*. While we see differences in the results, the gap is narrow and the relative ranking of different SemLMs does not change. This indicates that the automatic SRL and Co-reference annotations added some noise but, more importantly, that the resulting SemLMs are robust to this noise as we still retain the language modeling ability for all methods.

Additionally, our ablation study removes the "FrameNet Mapping" step in Sec. 5.2 ("FC-FM" and "EC-FM" rows), resulting in only using PropBank frames in the vocabulary. The increase in perplexities shows that "FrameNet Mapping" does produce a higher level of abstraction, which is useful for language modeling.

6.1.2 Narrative Cloze Test

We follow the Narrative Cloze Test idea used in script learning (Chambers and Jurafsky, 2008b; Chambers and Jurafsky, 2009). As Rudinger et al. (2015) points out, the narrative cloze test can be regarded as a language modeling evaluation. In the narrative cloze test, we randomly choose and remove one token from each semantic sequence in the test set. We then use language models to predict the missing token and evaluate the correctness. For all SemLMs, we use the conditional probabilities defined in Sec. 4 to get token predictions. We also use ordered PMI as an additional baseline. The narrative cloze test is conducted on

Table 4: **Narrative cloze test results for SemLMs.** UNI, BG, TRI, CBOW, SG, LB are different language model implementations while "FC" and "EC" stand for our two SemLM models, respectively. "FC-FM" and "EC-FM" mean that we remove the FrameNet mappings. "w/o DIS" indicates the removal of discourse makers in SemLMs. "Rel-Impr" indicates the relative improvement of the best performing SemLM over the strongest baseline. We evaluate on two metrics: mean reciprocal rank (MRR)/recall at 30 (Recall@30). LB outperforms other methods for both frame-chain and entity-centered SemLMs.

	Baselines			SemLMs				Rel-Impr
	OP	UNI	BG	TRI	CBOW	SG	LB	
MRR								
NYT Hold-out Data								
FC	0.121	0.236	0.225	0.249	0.242	0.247	**0.276**	8.5%
EC	0.126	0.235	0.210	0.242	0.249	0.249	**0.261**	5.9%
EC w/o DIS	0.092	0.191	0.188	0.212	0.215	0.216	**0.227**	18.8%
Rudinger et al. (2015)*	0.083	0.186	0.181	—	—	—	**0.223**	19.9%
Gold PropBank Data with Frame Chains								
FC	0.106	0.215	0.212	0.232	0.228	0.229	**0.254**	18.1%
FC-FM	0.098	0.201	0.204	0.223	0.218	0.220	0.243	———
Gold Ontonotes Data with Coref Chains								
EC	0.122	0.228	0.213	0.239	0.247	0.246	**0.257**	12.7%
EC-FM	0.109	0.215	0.208	0.230	0.237	0.239	0.254	———
Recall@30								
NYT Hold-out Data								
FC	33.2	46.8	45.3	47.3	46.6	47.5	**55.4**	18.4%
EC	29.4	43.7	41.6	44.8	46.5	46.6	**52.0**	19.0%
Gold PropBank Data with Frame Chains								
FC	26.3	39.5	38.1	45.5	43.6	43.8	**53.9**	36.5%
FC-FM	24.4	37.3	37.3	42.8	41.9	42.1	48.2	———
Gold Ontonotes Data with Coref Chains								
EC	30.6	42.1	39.7	46.4	48.3	48.1	**51.5**	22.3%
EC-FM	26.6	39.9	37.6	45.4	46.7	46.2	49.8	———

the same test sets as the perplexity evaluation. We use mean reciprocal rank (MRR) and recall at 30 (Recall@30) to evaluate.

Results are provided in Table 4. Consistent with the results in the perplexity evaluation, LB outperforms other methods for both frame-chain and entity-centered SemLMs across all test sets. It is interesting to see that UNI performs better than BG in this prediction task. This finding is also reflected in the results reported in Rudinger et al. (2015). Though CBOW and SG have similar perplexity results, SG appears to be stronger in the narrative cloze test. With respect to the strongest baseline (UNI), LB achieves close to 20% relative improvement for Recall@30 metric on NYT hold-out data. On gold data, the frame-chain SemLMs get a relative improvement of 36.5% for Recall@30 while entity-centered SemLMs get 22.3%. For MRR metric, the relative improvement is around half that of the Recall@30 metric.

In the narrative cloze test, we also carry out an ablation study to remove the "FrameNet Mapping" step in Sec. 5.2 ("FC-FM" and "EC-FM" rows). The decrease in MRR and Recall@30 metrics further strengthens the argument that "FrameNet Mapping" is important for language modeling as it improves the generalization on frames.

We cannot directly compare with other related works (Rudinger et al., 2015; Pichotta and Mooney, 2016) because of the differences in data and evaluation metrics. Rudinger et al. (2015) also use the NYT portion of the Gigaword corpus, but with Concrete annotations; Pichotta and Mooney (2016) use the English Wikipedia as their data, and Stanford NLP tools for pre-processing while we use the Illinois NLP tools. Consequently, the eventual chain statistics are different, which leads to different test instances.[5] We counter this difficulty

[5]Rudinger et al. (2015) is similar to our entity-centered SemLM without discourse information. So, in Table 4, we

Table 5: **Co-reference resolution results with entity-centered SemLM features.** "EC" stands for the entity-centered SemLM. "TRI" is the tri-gram model while "LB" is the log-bilinear model. "p_c" means conditional probability features and "em" represents frame embedding features. "w/o DIS" indicates the ablation study by removing all discourse makers for SemLMs. We conduct the experiments by adding SemLM features into the base system. We outperform the state-of-art system (Wiseman et al., 2015), which reports the best results on CoNLL12 dataset. The improvement achieved by "EC_LB ($p_c + em$)" over the base system is statistically significant.

	ACE04	CoNLL12
Wiseman et al. (2015)	—	63.39
Base (Peng et al., 2015a)	71.20	63.03
Base+EC-TRI (p_c)	71.31	63.14
Base+EC-TRI w/o DIS	71.08	62.99
Base+EC-LB (p_c)	71.71	63.42
Base+EC-LB ($p_c + em$)	**71.79**	**63.46**
Base+EC-LB w/o DIS	71.12	63.00

by reporting results on "Gold PropBank Data" and "Gold Ontonotes Data". We hope that these two gold annotation datasets can become standard test sets. Rudinger et al. (2015) does share a common evaluation metric with us: MRR. If we ignore the data difference and make a rough comparison, we find that the absolute values of our results are better while Rudinger et al. (2015) have higher relative improvement ("Rel-Impr" in Table 4). This means that 1) the discourse information is very likely to help better model semantics 2) the discourse information may boost the baseline (UNI) more than it does for the LB model.

6.2 Evaluation of SemLM Applications

6.2.1 Co-reference Resolution

Co-reference resolution is the task of identifying mentions that refer to the same entity. To help improve its performance, we incorporate SemLM information as features into an existing co-reference resolution system. We choose the state-of-art Illinois Co-reference Resolution system (Peng et al., 2015a) as our base system. It employs a supervised joint mention detection and co-reference framework. We add additional features into the mention-pair feature set.

Given a pair of mentions (m_1, m_2) where m_1 appears before m_2, we first extract the corresponding semantic frame and the argument role label of each mention. We do this by following the procedures in Sec. 5. Thus, we can get a pair of semantic frames with argument information $(\text{fa}_1, \text{fa}_2)$. We may also get an additional discourse marker between these two frames, e.g. $(\text{fa}_1, \text{dis}, \text{fa}_2)$. Now, we add the following conditional probability as the feature from SemLMs:

$$p_c = p(\text{fa}_2|\text{fa}_1, \text{dis}).$$

We also add p_c^2, $\sqrt{p_c}$ and $1/p_c$ as features. To get the value of p_c, we follow the definitions in Sec. 4, and we only use the entity-centered SemLM here as its vocabulary covers frames with argument labels. For the neural language model implementations (CBOW, SG and LB), we also include frame embeddings as additional features.

We evaluate the effect of the added SemLM features on two co-reference benchmark datasets: ACE04 (NIST, 2004) and CoNLL12 (Pradhan et al., 2012). We use the standard split of 268 training documents, 68 development documents, and 106 testing documents for ACE04 data (Culotta et al., 2007; Bengtson and Roth, 2008). For CoNLL12 data, we follow the train and test document split from CoNLL-2012 Shared Task. We report CoNLL AVG for results (average of MUC, B^3, and $CEAF_e$ metrics), using the v7.0 scorer provided by the CoNLL-2012 Shared Task.

Co-reference resolution results with entity-centered SemLM features are shown in Table 5. Tri-grams with conditional probability features improve the performance by a small margin, while the log-bilinear model achieves a 0.4-0.5 F1 points improvement. By employing log-bilinear model embeddings, we further improve the numbers and we outperform the best reported results on the CoNLL12 dataset (Wiseman et al., 2015).

In addition, we carry out ablation studies to remove all discourse makers during the language modeling process. We re-train our models and study their effects on the generated features. Table 5 ("w/o DIS" rows) shows that without discourse information, the SemLM features would hurt the overall performance, thus proving the necessity of considering discourse for semantic language models.

6.2.2 Shallow Discourse Parsing

Shallow discourse parsing is the task of identifying explicit and implicit discourse connectives,

make a rough comparison between them.

Table 6: **Shallow discourse parsing results with frame-chain SemLM features.** "FC" stands for the frame-chain SemLM. "TRI" is the tri-gram model while "LB" is the log-bilinear model. "p_c", "em" are conditional probability and frame embedding features, resp. "w/o DIS" indicates the case where we remove all discourse makers for SemLMs. We do the experiments by adding SemLM features to the base system. The improvement achieved by "FC-LB ($p_c + em$)" over the baseline is statistically significant.

	CoNLL16 Test			CoNLL16 Blind		
	Explicit	Implicit	Overall	Explicit	Implicit	Overall
Base (Song et al., 2015)	89.8	35.6	60.4	75.8	31.9	52.3
Base + FC-TRI (q_c)	90.3	35.8	60.7	76.4	32.5	52.9
Base + FC-TRI w/o DIS	89.2	35.3	60.0	75.5	31.6	52.0
Base + FC-LB (q_c)	90.9	36.2	61.3	76.8	32.9	53.4
Base + FC-LB ($q_c + em$)	**91.1**	**36.3**	**61.4**	**77.3**	**33.2**	**53.8**
Base + FC-LB w/o DIS	90.1	35.7	60.6	76.9	33.0	53.5

determine their senses and their discourse arguments. In order to show that SemLM can help improve shallow discourse parsing, we evaluate on identifying the correct sense of discourse connectives (both explicit and implicit ones).

We choose Song et al. (2015), which uses a supervised pipeline approach, as our base system. The system extracts context features for potential discourse connectives and applies the discourse connective sense classifier. Consider an explicit connective "dis"; we extract the semantic frames that are closest to it (left and right), resulting in the sequence [f_1, dis, f_2] by following the procedures described in Sec. 5. We then add the following conditional probabilities as features. Compute

$$q_c = p(\mathrm{dis}|\mathrm{f}_1, \mathrm{f}_2).$$

and, similar to what we do for co-reference resolution, we add $q_c, q_c^2, \sqrt{q_c}, 1/q_c$ as conditional probability features, which can be computed following the definitions in Sec. 4. We also include frame embeddings as additional features. We only use frame-chain SemLMs here.

We evaluate on CoNLL16 (Xue et al., 2015) test and blind sets, following the train and development document split from the Shared Task, and report F1 using the official shared task scorer.

Table 6 shows the results for shallow discourse parsing with SemLM features. Tri-gram with conditional probability features improve the performance for both explicit and implicit connective sense classifiers. Log-bilinear model with conditional probability features achieves even better results, and frame embeddings further improve the numbers. SemLMs improve relatively more on explicit connectives than on implicit ones.

We also show an ablation study in the same setting as we did for co-reference, i.e. removing discourse information ("w/o DIS" rows). While our LB model can still exhibit improvement over the base system, its performance is lower than the proposed discourse driven version, which means that discourse information improves the expressiveness of semantic language models.

7 Conclusion

The paper builds two types of discourse driven semantic language models with four different language model implementations that make use of neural embeddings for semantic frames. We use perplexity and a narrative cloze test to prove that the proposed SemLMs have a good level of abstraction and are of high quality, and then apply them successfully to the two challenging tasks of co-reference resolution and shallow discourse parsing, exhibiting improvements over state-of-the-art systems. In future work, we plan to apply SemLMs to other semantic related NLP tasks e.g. machine translation and question answering.

Acknowledgments

The authors would like to thank Christos Christodoulopoulos and Eric Horn for comments that helped to improve this work. This work is supported by Contract HR0011-15-2-0025 with the US Defense Advanced Research Projects Agency (DARPA). Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government. This material is also based upon work supported by the U.S. Department of Homeland Security under Award Number 2009-ST-061-CCI002-07.

References

C. F. Baker, C. J. Fillmore, and J. B. Lowe. 1998. The berkeley framenet project. In *COLING/ACL*, pages 86–90.

N. Balasubramanian, S. Soderland, Mausam, and O. Etzioni. 2013. Generating coherent event schemas at scale. In *EMNLP*, pages 1721–1731.

D. Bamman and N. A. Smith. 2014. Unsupervised discovery of biographical structure from text. *TACL*, 2:363–376.

C. A. Bejan. 2008. Unsupervised discovery of event scenarios from texts. In *FLAIRS Conference*, pages 124–129.

E. Bengtson and D. Roth. 2008. Understanding the value of features for coreference resolution. In *EMNLP*.

N. Chambers and D. Jurafsky. 2008a. Jointly combining implicit constraints improves temporal ordering. In *EMNLP*.

N. Chambers and D. Jurafsky. 2008b. Unsupervised learning of narrative event chains. In *ACL*, volume 94305, pages 789–797.

N. Chambers and D. Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *ACL*, volume 2, pages 602–610.

N. Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *EMNLP*, volume 13, pages 1797–1807.

J. C. K. Cheung, H. Poon, and L. Vanderwende. 2013. Probabilistic frame induction. *arXiv:1302.4813*.

A. Culotta, M. Wick, R. Hall, and A. McCallum. 2007. First-order probabilistic models for coreference resolution. In *NAACL*.

L. Frermann, I. Titov, and Pinkal. M. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *EACL*.

M. Granroth-Wilding, S. Clark, M. T. Llano, R. Hepworth, S. Colton, J. Gow, J. Charnley, N. Lavrač, M. Žnidaršič, and M. Perovšek. 2015. What happens next? event prediction using a compositional neural network model.

M. Gutmann and A. Hyvarinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*.

E. Hovy, M. Marcus, M. Palmer, L. Ramshaw, and R. Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of HLT/NAACL*.

J. Irwin, M. Komachi, and Y. Matsumoto. 2011. Narrative schema as world knowledge for coreference resolution. In *CoNLL Shared Task*, pages 86–92.

B. Jans, S. Bethard, I. Vulić, and M. F. Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *EACL*, pages 336–344.

P. Kingsbury and M. Palmer. 2002. From Treebank to PropBank. In *Proceedings of LREC-2002*.

R. Kneser and H. Ney. 1995. Improved backing-off for m-gram language modeling. In *ICASSP*.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv:1301.3781*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *NAACL*.

A. Mnih and G. Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*, pages 641–648.

A. Modi and I. Titov. 2014a. Inducing neural models of script knowledge. In *CoNLL*.

A. Modi and I. Titov. 2014b. Learning semantic script knowledge with event embeddings. In *ICLR Workshop*.

R. Mooney and G. DeJong. 1985. Learning schemata for natural language processing.

K.-H. Nguyen, X. Tannier, O. Ferret, and R. Besançon. 2015. Generative event schema induction with entity disambiguation. In *ACL*.

US NIST. 2004. The ace evaluation plan. *US National Institute for Standards and Technology (NIST)*.

B. Paul, B. Phil, and H. Hieu. 2014. Oxlm: A neural language modelling framework for machine translation. *The Prague Bulletin of Mathematical Linguistics*, 102(1):81–92.

H. Peng, K. Chang, and D. Roth. 2015a. A joint framework for coreference resolution and mention head detection. In *CoNLL*.

H. Peng, D. Khashabi, and D. Roth. 2015b. Solving hard coreference problems. In *NAACL*.

K. Pichotta and R. J. Mooney. 2014. Statistical script learning with multi-argument events. In *EACL*, volume 14, pages 220–229.

K. Pichotta and R. J. Mooney. 2016. Learning statistical scripts with lstm recurrent neural networks. In *AAAI*.

S. Pradhan, A. Moschitti, N. Xue, O. Uryupina, and Y. Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *CoNLL*.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The penn discourse treebank 2.0. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC 2008)*.

V. Punyakanok, D. Roth, W. Yih, and D. Zimak. 2004. Semantic role labeling via integer linear programming inference. In *COLING*.

A. Rahman and V. Ng. 2011. Coreference resolution with world knowledge. In *ACL*.

D. Roth and D. Zelenko. 1998. Part of speech tagging using a network of linear separators. In *COLING-ACL*.

R. Rudinger, P. Rastogi, F. Ferraro, and B. Van Durme. 2015. Script induction as language modeling. In *EMNLP*.

R. C. Schank and R. P. Abelson. 1977. Scripts, plans, goals, and understanding: An inquiry into human knowledge structures. In *JMZ*.

K. K. Schuler. 2005. Verbnet: A broad-coverage, comprehensive verb lexicon.

Y. Song, H. Peng, P. Kordjamshidi, M. Sammons, and D. Roth. 2015. Improving a pipeline architecture for shallow discourse parsing. In *CoNLL Shared Task*.

A. Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*, volume 2002, page 2002.

I. Titov and E. Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *NAACL*.

Ben Wellner and James Pustejovsky. 2007. Automatically identifying the arguments of discourse connectives. In *Proceedings of the 2007 Joint Conference of EMNLP-CoNLL*.

T. Winograd. 1972. Understanding natural language. *Cognitive psychology*, 3(1):1–191.

S. Wiseman, A. M. Rush, S. M. Shieber, and J. Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *ACL*.

N. Xue, H. T. Ng, S. Pradhan, R. P. C. Bryant, and A. T. Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *CoNLL*.

Sentiment Domain Adaptation with Multiple Sources

Fangzhao Wu and Yongfeng Huang*
Tsinghua National Laboratory for Information Science and Technology
Department of Electronic Engineering
Tsinghua University, Beijing, China
wufangzhao@gmail.com, yfhuang@tsinghua.edu.cn

Abstract

Domain adaptation is an important research topic in sentiment analysis area. Existing domain adaptation methods usually transfer sentiment knowledge from only one source domain to target domain. In this paper, we propose a new domain adaptation approach which can exploit sentiment knowledge from multiple source domains. We first extract both global and domain-specific sentiment knowledge from the data of multiple source domains using multi-task learning. Then we transfer them to target domain with the help of words' sentiment polarity relations extracted from the unlabeled target domain data. The similarities between target domain and different source domains are also incorporated into the adaptation process. Experimental results on benchmark dataset show the effectiveness of our approach in improving cross-domain sentiment classification performance.

1 Introduction

Sentiment classification is a hot research topic in natural language processing field, and has many applications in both academic and industrial areas (Pang and Lee, 2008; Liu, 2012; Wu et al., 2015; Wu and Huang, 2016). Sentiment classification is widely known as a domain-dependent task (Blitzer et al., 2007; Glorot et al., 2011). The sentiment classifier trained in one domain may not perform well in another domain. This is because sentiment expressions used in different domains are usually different. For example, "boring" and "lengthy" are frequently used to express negative sentiments in *Book* domain. However, they rarely appear in *Electronics* domain (Bollegala et al., 2011). Thus a sentiment classifier trained in *Electronics* domain cannot accurately predict their sentiments in *Book* domain. In addition, the same word may convey different sentiments in different domains. For example, in *Electronics* domain "easy" is usually used in positive reviews, e.g., "this digital camera is easy to use." However, it is frequently used as a negative word in *Movie* domain. For instance, "the ending of this movie is easy to guess." Thus, the sentiment classifier trained in one domain usually cannot be applied to another domain directly (Pang and Lee, 2008).

In order to tackle this problem, sentiment domain adaptation has been widely studied (Liu, 2012). For example, Blitzer et al. (2007) proposed to compute the correspondence among features from different domains using their associations with pivot features based on structural correspondence learning (*SCL*). Pan et al. (2010) proposed a spectral feature alignment (*SFA*) algorithm to align the domain-specific words from different domains in order to reduce the gap between source and target domains. However, all of these methods transfer sentiment information from only one source domain. When the source and target domains have significant difference in feature distributions, the adaptation performance will heavily decline. In some cases, the performance of sentiment domain adaptation is even worse than that without adaptation, which is usually known as *negative transfer* (Pan and Yang, 2010).

In this paper we propose a new domain adaptation approach for cross-domain sentiment classification. Our approach can exploit the sentiment information in multiple source domains to reduce the risk of negative transfer effectively. Our approach consists of two steps, i.e., *training* and

*Corresponding author.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 301–310,
Berlin, Germany, August 7-12, 2016.

adaptation. At the *training* stage, we extract two kinds of sentiment models, i.e., the global model and the domain-specific models, from the data of multiple source domains using multi-task learning. The global sentiment model can capture the common sentiment knowledge shared by various domains, and has better generalization performance than the sentiment model trained in a single source domain. The domain-specific sentiment model can capture the specific sentiment knowledge in each source domain. At the *adaptation* stage, we transfer both kinds of sentiment knowledge to target domain with the help of the words' sentiment graph of target domain. The sentiment graph contains words' domain-specific sentiment polarity relations extracted from the syntactic parsing results of the unlabeled data in target domain. Since sentiment transfer between similar domains is more effective than dissimilar domains, we incorporate the similarities between target domain and different source domains into the adaptation process. In order to estimate the similarity between two domains, we propose a novel domain similarity measure based on their sentiment graphs. Extensive experiments were conducted on the benchmark Amazon product review dataset. The experimental results show that our approach can improve the performance of cross-domain sentiment classification effectively.

2 Related work

Sentiment classification is widely known as a domain-dependent task, since different expressions are used to express sentiments in different domains (Blitzer et al., 2007). The sentiment classifier trained in one domain may not perform well in another domain. Since there are massive domains, it is impractical to annotate enough data for each new domain. Thus, *domain adaptation*, or so called *cross-domain sentiment classification*, which transfers the sentiment knowledge from domains with sufficient labeled data (i.e., source domain) to a new domain with no or scarce labeled data (i.e., target domain), has been widely studied. Existing domain adaptation methods mainly transfer sentiment information from only one source domain. For example, Blitzer et al. (2007) proposed a domain adaptation method based on structural correspondence learning (*SCL*). In their method, a set of pivot features are first selected according to their associations with source domain labels. Then the correspondence among features from source and target domains is computed using their associations with pivot features. In order to reduce the gap between source and target domains, Pan et al. (2010) proposed a spectral feature alignment (*SFA*) algorithm to align the domain-specific sentiment words from different domains into clusters. He et al. (2011) proposed to extract polarity-bearing topics using joint sentiment-topic (*JST*) model to expand the feature representations of texts from both source and target domains. Li et al. (2009) proposed to transfer sentiment knowledge from source domain to target domain using nonnegative matrix factorization. A common shortcoming of above methods is that if the source and target domains have significantly different distributions of sentiment expressions, then the domain adaptation performance will heavily decline (Li et al., 2013).

Using multiple source domains in cross-domain sentiment classification has also been explored. Glorot et al. (2011) proposed a sentiment domain adaptation method based on a deep learning technique, i.e., Stacked Denoising Auto-encoders. The core idea of their method is to learn a high-level representation that can capture generic concepts using the unlabeled data from multiple domains. Yoshida et al. (2011) proposed a probabilistic generative model for cross-domain sentiment classification with multiple source and target domains. In their method, each word is assigned three attributes, i.e., the domain label, the domain dependence/independence label, and sentiment polarity. Bollegala et al. (2011) proposed to construct a sentiment sensitive thesaurus for cross-domain sentiment classification using data from multiple source domains. This thesaurus is used to expand the feature vectors for both training and classification. However, the similarities between target domain and different source domains are not considered in these methods. In addition, although unlabeled data is utilized in these methods, the useful word-level sentiment knowledge in the unlabeled target domain data is not exploited.

General-purpose multiple source domain adaptation methods have also been studied. For example, Mansour et al. (2009) proposed a distribution weighted hypothesis combination approach, and gave theoretical guarantees for it. However, this method is based on the assumption that target distribution is some mixture of source distri-

butions, which may not hold in sentiment domain adaptation scenario. Duan et al. (2009) proposed a Domain Adaptation Machine (*DAM*) method to learn a Least-Squares SVM classifier for target domain by leveraging the classifiers independently trained in multiple source domains. Chattopadhyay et al. (2011) explored to assign psuedo labels to unlabeled samples in the target domain using the classifiers from multiple source domains. Then target domain classifier is trained on these psuedo labeled samples. Compared with these general-purpose domain adaptation methods with multiple source domains, our approach is more suitable for sentiment domain adaptation because our approach exploits more sentiment-related characteristics and knowledge, such as the general sentiment knowledge shared by different domains and the word-level sentiment polarity relations, which is validated by experiments.

3 Sentiment Graph Extraction and Domain Similarity Measure

In this section we introduce two important components used in our sentiment domain adaptation approach, i.e., the words' sentiment graph and domain similarity.

3.1 Sentiment Graph Extraction

Compared with labeled data, unlabeled data is usually much easier and cheaper to collect on a large scale. Although unlabeled samples are not associated with sentiment labels, they can still provide a lot of useful sentiment information for domain adaptation. For example, if "great" and "quick" are frequently used to describe the same target in the same review of *Kitchen* domain, then they probably convey the same sentiment polarity in this domain. Since "great" is a general positive word in both *Book* and *Kitchen* domains, we can infer that "quick" is also a positive word in *Kitchen* domain when transferring from *Book* domain to *Kitchen* domain.

Motivated by above observations, in this paper we propose to extract sentiment polarity relations among words from massive unlabeled data for sentiment domain adaptation. Two kinds of polarity relations are explored, i.e., sentiment coherent relation and sentiment opposite relation. The former means that two words convey the same sentiment polarity while the latter indicates opposite sentiment polarities. These polarity relations are extracted from the syntactic parsing results according to manually selected rules. Two rules are used to extract sentiment coherent relations. The first one is that two words are connected by coordinating conjunctions such as "and" and "as well as". For example, a review in *Kitchen* domain may be "it is so high-quality and professional." Since "high-quality" and "professional" are connected by the coordinating conjunction "and", we infer that they probably convey the same sentiment polarity. The second rule is that two words are not directly connected but are used to describe the same target in the same sentence. For example, a review in *Electronics* domain may be "It is a beautiful, durable, easy-to-use camera." Since "beautiful", "durable", and "easy-to-use" are all used to describe the same camera in the same review, they tend to convey the same sentiment polarity. We also propose two rules for extracting sentiment opposite relations. The first rule is that two words are connected by adversative conjunctions such as "but" and "however". The second rule is that two words are connected by coordinating conjunctions but there is a negation symbol before one of them. For example, a review may be "The battery of this camera is small and not durable." We can infer that "small" and "durable" may convey opposite sentiments when they are used to describe camera battery. An illustrative example of extracting sentiment polarity relations from syntactic parsing results is shown in Fig. 1.

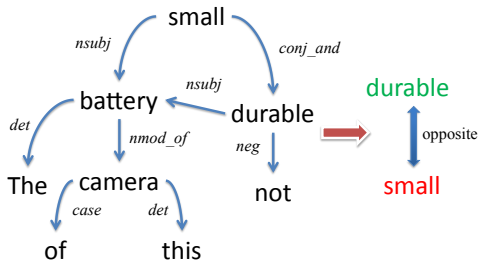


Figure 1: An illustrative example of extracting sentiment polarity relations from syntactic parsing results.

Based on the sentiment polarity relations among words extracted from the unlabeled data, we can build a words' sentiment graph for each domain. The nodes of the sentiment graph represent words and the edges stand for sentiment polarity relations. We denote $\mathbf{R} \in \mathbb{R}^{D \times D}$ as the words' sentiment graph of a specific domain. $R_{i,j}$ represents the sentiment polarity relation score between words i

and j. In this paper we define $R_{i,j}$ as $\frac{n^C_{i,j}-n^O_{i,j}}{n^C_{i,j}+n^O_{i,j}}$, where $n^C_{i,j}$ and $n^O_{i,j}$ represent the frequencies of words i and j sharing coherent and opposite sentiment polarity relations respectively in all the unlabeled samples. Thus, $R_{i,j} \in [-1, 1]$. If $R_{i,j} > 0$, then words i and j tend to convey the same sentiment polarity. Similarly, if $R_{i,j} < 0$, then these two words are more likely to convey opposite sentiments. The absolute value of $R_{i,j}$ represents the confidence of this sentiment polarity relation.

3.2 Domain Similarity

Different pairs of domains have different sentiment relatedness (Remus, 2012; Wu and Huang, 2015). Researchers have found that sentiment domain adaptation between similar domains, such as *Kitchen* and *Electronics*, is much more effective than that between dissimilar domains, such as *Kitchen* and *Book* (Blitzer et al., 2007; Pan et al., 2010). Thus, it is beneficial if we take the similarity between source and target domains into consideration when transferring sentiment knowledge.

In this paper we explore two methods to measure domain similarity. The first one is based on term distribution. The assumption behind this method is that similar domains usually share more common terms than dissimilar domains. For example, *Smart Phone* and *Digital Camera* domains share many common terms such as "screen", "battery", "light", and "durable", while the term distributions of *Digital Camera* and *Book* domains may have significant difference. Term distribution based domain similarity measures, such as $\mathcal{A}$-distance, have been explored in previous works (Blitzer et al., 2007). Inspired by (Remus, 2012), here we apply Jensen-Shannon divergence to measure domain similarity based on term distributions, which is more easy to compute than $\mathcal{A}$-distance. Denote $\mathbf{t}^m \in \mathbb{R}^{D\times 1}$ as the term distribution of domain m, where t^m_w is the probability of term w appearing in domain m. Then the similarity between domains m and n is formulated as:

$$\begin{aligned} TermSim(m,n) &= 1 - D_{JS}(\mathbf{t}^m, \mathbf{t}^n) \\ &= 1 - \frac{1}{2}(D_{KL}(\mathbf{t}^m, \bar{\mathbf{t}}) + D_{KL}(\mathbf{t}^n, \bar{\mathbf{t}})), \end{aligned} \tag{1}$$

where $\bar{\mathbf{t}} = \frac{1}{2}(\mathbf{t}^m + \mathbf{t}^n)$ is the average distribution, and $D_{KL}(\cdot,\cdot)$ is the Kullback-Leibler divergence:

$$D_{KL}(\mathbf{p}, \mathbf{q}) = \sum_{i=1}^{D} p_i \log_2 \left(\frac{p_i}{q_i}\right). \tag{2}$$

We can verify that $D_{JS}(\mathbf{t}^m, \mathbf{t}^n) \in [0, 1]$. Thus, the range of $TermSim(m, n)$ is also $[0, 1]$.

The term distribution based domain similarity can measure whether similar words are used in two domains. However, sharing similar terms does not necessarily mean that sentiment expressions are used similarly in these domains. For example, *CPU* and *Battery* are both related to electronics. The word "fast" is positive when used to describe *CPU*. However, it is frequently used as a negative word in *Battery* domain. For example, "this battery runs out fast." Thus, it is more useful to measure domain similarity based on sentiment word distributions. However, although we can infer the sentiment word distributions of source domains according to labeled samples, it is difficult to compute the sentiment word distribution of target domain, since the labeled data does not exist or is very scarce in target domain.

In order to tackle this problem, in this paper we propose to estimate the similarity between two domains based on their sentiment graphs. Similar domains usually share more common sentiment words and sentiment word pairs than dissimilar domains. In addition, the polarity relation scores of a pair of words in the sentiment graphs of similar domains are also more similar. In other words, they tend to be both positive or negative in these two domains. Motivated by above observations, the domain similarity based on sentiment graph is formulated as follows:

$$SentiSim(m,n) = \frac{\sum_{w=1}^{D}\sum_{v\neq w} |R^m_{w,v} + R^n_{w,v}| \cdot N^{m\cap n}_{w,v}}{\sum_{w=1}^{D}\sum_{v\neq w} (|R^m_{w,v}| \cdot N^m_{w,v} + |R^n_{w,v}| \cdot N^n_{w,v})}, \tag{3}$$

where $R^m_{w,v}$ is the sentiment polarity relation score between words w and v in domain m, and $N^m_{w,v}$ is its frequency in this domain. $N^{m\cap n}_{w,v} = \min\{N^m_{w,v}, N^n_{w,v}\}$. We can verify that $SentiSim(m, n) \in [0, 1]$. If two domains have more common sentiment word pairs and the polarity relation scores of these word pairs are more similar, then these two domains share higher domain similarity according to Eq. (3).

4 Sentiment Domain Adaptation with Multiple Sources

In this section we introduce our sentiment domain adaptation approach in detail. First we introduce several notations that will be used in following discussions. Assume there are M source domains.

Denote $\{\mathbf{X}^m \in \mathbb{R}^{N_m \times D}, \mathbf{y}^m \in \mathbb{R}^{N_m \times 1}\}$ as the labeled data in source domain m, where N_m is the number of labeled samples and D is the size of feature vector. $\mathbf{x}_i^m \in \mathbb{R}^{D \times 1}$ is the feature vector of the i_{th} sample in domain m, and its sentiment label is y_i^m. In this paper we focus on sentiment polarity classification, and $y_i^m \in \{+1, -1\}$. Denote $\mathbf{w} \in \mathbb{R}^{D \times 1}$ as the global sentiment model extracted from multiple source domains and $\mathbf{w}_m \in \mathbb{R}^{D \times 1}$ as the domain-specific sentiment model of source domain m. Denote $\mathbf{w}_t \in \mathbb{R}^{D \times 1}$ as the domain-specific sentiment model of target domain. Denote $f(\mathbf{x}, y, \mathbf{w})$ as the loss of classifying sample $\mathbf{x}$ into label y under linear classification model $\mathbf{w}$. Our approach is flexible to the selection of loss function f, which can be square loss, logistic loss, and hinge loss. Denote $\mathbf{R}^m \in \mathbb{R}^{D \times D}$ as the sentiment graph knowledge of domain m, and $S_{m,t} \in [0, 1]$ as the similarity between source domain m and target domain.

Our sentiment domain adaptation with multiple sources approach (*SDAMS*) consists of two steps, i.e., *training* and *adaptation*. At the *training* stage, the global and domain-specific sentiment knowledge are extracted from the data of multiple source domains. And at the *adaptation* stage, these two kinds of sentiment knowledge are transferred to target domain by incorporating the sentiment graph knowledge of target domain and the similarities between target and source domains.

4.1 Training

Given the labeled data and the sentiment graph knowledge of multiple source domains, at the *training* stage, our goal is to train a robust global sentiment model to capture the general sentiment knowledge shared by various domains and a domain-specific sentiment model for each source domain. The model of the *training* process is motivated by multi-task learning (Evgeniou and Pontil, 2004; Liu et al., 2009) and is formulated as:

$$\begin{aligned}\arg\min_{\mathbf{w},\mathbf{w}_m} \mathcal{L}(\mathbf{w}, \mathbf{w}_m) &= \sum_{m=1}^{M} \frac{1}{N_m} \sum_{i=1}^{N_m} f(\mathbf{x}_i^m, y_i^m, \mathbf{w} + \mathbf{w}_m) \\ &+ \alpha \sum_{m=1}^{M} \sum_{i=1}^{D} \sum_{j \neq i} R_{i,j}^m |(w_i + w_{m,i}) - (w_j + w_{m,j})| \\ &+ \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_1 \sum_{m=1}^{M} \|\mathbf{w}_m\|_2^2 + \lambda_2 \|\mathbf{w}\|_1 + \lambda_2 \sum_{m=1}^{M} \|\mathbf{w}_m\|_1, \end{aligned} \tag{4}$$

where α, λ_1, and λ_2 are nonnegative regularization coefficients. The sentiment classification model of each source domain is decomposed into two components, i.e., a global one and a domain-specific one. The global sentiment model is shared by all source domains and is trained in these domains simultaneously. It is used to capture the general sentiment knowledge, such as the general sentiment words "great", "worst", "perfect" and so on. The domain-specific sentiment model is trained on the labeled data within one source domain and is used to capture the specific sentiment knowledge of this domain. For example, the domain-specific sentiment word "easy" is a positive word in *Electronics* domain but is used as a negative word in *Movie* domain.

In Eq. (4), the first term means minimizing the empirical classification loss on the labeled data of multiple source domains. In this way we incorporate the sentiment information in labeled samples into sentiment classifier learning. In the second term we incorporate the sentiment graph knowledge of each source domain. It is motivated by graph-guided fused Lasso (Chen et al., 2012). If two words have strong coherent (or opposite) sentiment polarity relations, then we constrain that their sentiment scores are more similar (or dissimilar) with each other in the final classification model. The L_1-norm regularization terms are motivated by Lasso (Tibshirani, 1996). It can set many minor sentiment scores in the models to exact zeros. Since not all the words convey sentiments, these terms can help conduct sentiment word selection. We also incorporate the L_2-norm regularization terms in order to improve model stability in high-dimensional problems, which is inspired by elastic net regularization (Zou and Hastie, 2003).

4.2 Adaptation

At the *adaptation* stage, we incorporate the global sentiment knowledge, the domain-specific sentiment knowledge of multiple source domains, the sentiment graph knowledge of target domain, and the domain similarities between target and source domains into a unified framework to learn an accurate sentiment classifier for target domain. The model of our *adaptation* framework is formulated as follows:

$$\begin{aligned}\arg\min_{\mathbf{w}_t} \mathcal{L}(\mathbf{w}_t) &= \sum_{m=1}^{M} S_{m,t} \|\mathbf{w}_m - \mathbf{w}_t\|_2^2 + \lambda_1 \|\mathbf{w}_t\|_2^2 \\ &+ \lambda_2 \|\mathbf{w}_t\|_1 + \beta \sum_{i=1}^{D} \sum_{j \neq i} R_{i,j}^t |(w_i + w_{t,i}) - (w_j + w_{t,j})|, \end{aligned} \tag{5}$$

where β, λ_1, and λ_2 are nonnegative regularization coefficients. The final sentiment classifier of the target domain is a linear combination of $\mathbf{w}$ and $\mathbf{w}_t$, i.e., $\mathbf{w}+\mathbf{w}_t$, where $\mathbf{w}$ is the global sentiment model extracted from multiple source domains at the *training* stage, and $\mathbf{w}_t$ is the domain-specific sentiment model of target domain learned at the *adaptation* stage. In the first term of Eq. (5), we transfer the knowledge in domain-specific sentiment models from multiple source domains to $\mathbf{w}_t$. Since the sentiment knowledge transfer between similar domains is more effective, the transfer of domain-specific sentiment knowledge is weighted by the similarities between target domain and different source domains. If target domain is more similar with source domain m than source domain n (i.e., $S_{m,t} > S_{n,t}$), then more domain-specific sentiment knowledge will be transferred to $\mathbf{w}_t$ from $\mathbf{w}_m$ than $\mathbf{w}_n$. Through the last term we incorporate the sentiment graph knowledge extracted from massive unlabeled data of target domain into the adaptation process. If two words share strong coherent (or opposite) sentiment polarity relations in the target domain, then we constrain that their sentiment scores in the sentiment classification model of target domain are more similar (or dissimilar). This term can help transfer the sentiment knowledge from source domains to target domain more effectively. For example, if we know that "great" is a positive word in the global sentiment model and there is a strong coherent polarity relation between "easy" and "great" in *Electronics* domain, then we can infer that "easy" is also a positive word in this domain.

5 Experiments

5.1 Dataset and Experimental Settings

The dataset used in our experiments is the famous Amazon product review dataset collected by Blitzer et al. (2007). It is widely used as a benchmark dataset for cross-domain sentiment classification. Four domains, i.e., *Book*, *DVD*, *Electronics* and *Kitchen*, are included in this dataset. Each domain contains 1,000 positive and 1,000 negative reviews. Besides, a large number of unlabeled reviews are provided. The detailed statistics of this dataset are shown in Table 1.

In our experiments, each domain was selected in turn as target domain, and remaining domains as source domains. In each experiment, we randomly selected N labeled samples from the source domains to train sentiment models at the *training* stage. These samples were balanced among different source domains. In order to perform fair comparisons with baseline methods, following (Bollegala et al., 2011), we limited the total number of training samples, i.e., N, to 1,600. The target domain sentiment classifier was tested on all the labeled samples of target domain. Following (Blitzer et al., 2007), unigrams and bigrams were used as features. The sentiment polarity relations of bigrams were extracted by expanding the polarity relations between unigrams using modifying relations. For example, from the review "this phone is very beautiful and not expensive," we extract not only sentiment polarity relation between "beautiful" and "expensive", but also polarity relation between "beautiful" and "not_expensive" (coherent sentiment), and that between "very_beautiful" and "expensive" (opposite sentiment), since "very" and "not" are used to modify "beautiful" and "expensive" respectively. Classification accuracy was selected as the evaluation metric. We manually set β in Eq. (5) to 0.01. The values of α, λ_1, and λ_2 in Eq. (4) were selected using cross validation. The optimization problems in Eq. (4) and Eq. (5) were solved using alternating direction method of multipliers (ADMM) (Boyd et al., 2011). Each experiment was repeated 10 times independently and average results were reported.

Table 1: The statistics of the dataset.

Domain	Book	DVD	Electronics	Kitchen
positive	1,000	1,000	1,000	1,000
negative	1,000	1,000	1,000	1,000
unlabeled	973,194	122,438	21,009	17,856

5.2 Comparison of Domain Similarity Measures

In this section, we conducted experiments to compare the effectiveness of the two kinds of domain similarity measures introduced in Section 3.2 in sentiment domain adaptation task. The experimental results are summarized in Fig. 2. The classification loss function used in our approach is hinge loss. The results of other loss functions show similar patterns.

From Fig. 2 we can see that the domain similarity measure based on sentiment graph performs consistently better than that based on term distribution in our approach. This result validates our

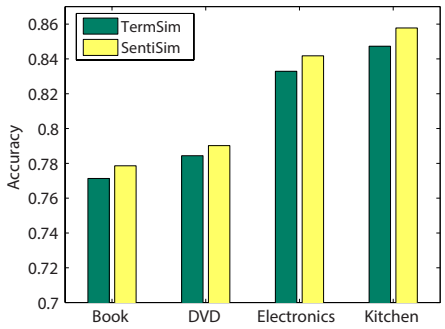


Figure 2: The performance of our approach with different kinds of domain similarity measure.

assumption in Section 3.2 that the sentiment graph based domain similarity can better model the sentiment relatedness between different domains than that based on term distribution in sentiment domain adaptation task. In all the following experiments, the sentiment graph based domain similarities were used in our approach.

5.3 Performance Evaluation

In this section we conducted experiments to evaluate the performance of our approach by comparing it with a series of baseline methods. The methods to be compared are: 1) *SCL*, domain adaptation based on structural correspondence learning (Blitzer et al., 2007); 2) *SFA*, domain adaptation based on spectral feature alignment (Pan et al., 2010); 3) *SCL-com* and *SFA-com*, adapting *SCL* and *SFA* to multiple source domain scenario by first training a cross-domain sentiment classifier in each source domain and then combining their classification results using majority voting; 4) *SST*, cross-domain sentiment classification by using multiple source domains to construct a sentiment sensitive thesaurus for feature expansion (Bollegala et al., 2011); 5) *IDDIWP*, multiple-domain sentiment analysis by identifying domain dependent/independent word polarity (Yoshida et al., 2011); 6) *DWHC*, *DAM* and *CP-MDA*, three general-purpose multiple source domain adaptation methods proposed in (Mansour et al., 2009), (Duan et al., 2009) and (Chattopadhyay et al., 2011) respectively; 7) *SDAMS-LS*, *SDAMS-SVM*, and *SDAMS-Log*, our proposed sentiment domain adaptation approaches with square loss, hinge loss, and logistic loss respectively; 8) *All-Training*, all the domains were involved in the training phase of our approach and there is no adaptation phase. This method is introduced to provide an upper bound for the performance of our approach. The experimental results of these methods are summarized in Table 2.

Table 2: The performance of different methods.

	Book	DVD	Electronics	Kitchen
SCL	0.7457	0.7630	0.7893	0.8207
SFA	0.7598	0.7848	0.7808	0.8210
SCL-com	0.7523	0.7675	0.7918	0.8247
SFA-com	0.7629	0.7869	0.7864	0.8258
SST	0.7632	0.7877	0.8363	0.8518
IDDIWP	0.7524	0.7732	0.8167	0.8383
DWHC	0.7611	0.7821	0.8312	0.8478
DAM	0.7563	0.7756	0.8284	0.8419
CP-MDA	0.7597	0.7792	0.8331	0.8465
SDAMS-LS	0.7795	0.7880	0.8398	0.8596
SDAMS-SVM	0.7786	0.7902	**0.8418**	0.8578
SDAMS-Log	**0.7829**	**0.7913**	0.8406	**0.8629**
All-Training	0.7983	0.8104	0.8463	0.8683

From Table 2 we can see that our approach achieves the best performance among all the methods compared here. *SCL* and *SFA* are famous cross-domain sentiment classification methods. In these methods, the sentiment knowledge is transferred from one source domain to target domain. According to Table 2, our approach performs significantly better than them. This result indicates that the sentiment knowledge extracted from one source domain may contain heavy domain-specific bias and may be inappropriate for the target domain. Our approach can tackle this problem by extracting the global sentiment model from multiple source domains. This global model can capture the general sentiment knowledge shared by various domains and has better generalization performance. It can reduce the risk of negative transfer effectively. Our approach also outperforms *SCL-com* and *SFA-com*. In *SCL-com* and *SFA-com*, the sentiment information in different source domains is combined at the classification stage, while in our approach it is combined at the learning stage. The superior performance of our approach compared with *SCL-com* and *SFA-com* shows that our approach is a more appropriate way to exploit the sentiment knowledge in different source domains. *SST* and *IDDIWP* also utilize data from multiple source domains as our approach. But our approach can still outperform them. This is because in these methods, the similarities between target domain and different source domains are not considered. Since different domains usually

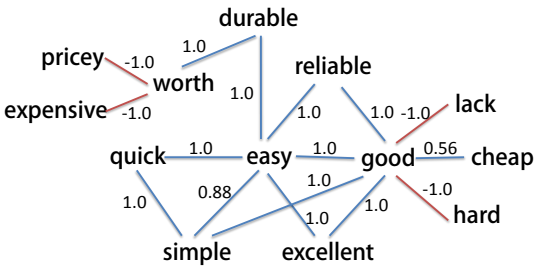


Figure 3: An illustrative example of the sentiment graph of *Electronics* domain. The value on the line represents the sentiment polarity relation score.

have different sentiment relatedness, our approach can exploit the sentiment information in multiple source domains more accurately by incorporating the similarities between target domain and each source domain into the *adaptation* process. Our approach also outperforms the state-of-the-art general-purpose multiple source domain adaptation methods, such as *DWHC*, *DAM*, and *CP-MDA*. This is because our approach can exploit more sentiment-related characteristics and knowledge for sentiment domain adaptation, such as the general sentiment knowledge shared by various domains, the sentiment graph based domain similarities, and the word-level sentiment polarity relations. Thus, our approach is more suitable for sentiment domain adaptation than these general-purpose multiple source domain adaptation methods. Another observation from Table 2 is that the performance of our approach is quite close to the upper bound, i.e., *All-Training*, especially in *Electronics* and *Kitchen* domains. This result validates the effectiveness of our approach in sentiment domain adaptation.

5.4 Case Study

In this section we conducted several case studies to further explore how our sentiment domain adaptation approach works. As an illustrative example, we selected *Electronics* domain as the target domain and remaining domains as source domains. The top sentiment words in the global and domain-specific sentiment models learned from the data of multiple source domains are shown in Table 3. A subgraph of the sentiment graph extracted from the unlabeled data of target domain (*Electronics*) is shown in Fig. 3. The top words in the final domain-specific sentiment model of target domain returned by our approach are shown in Table 3.

From Table 3 we have following observations. First, the global sentiment model extracted from multiple source domains can capture the general sentiment knowledge quite well. It contains many general sentiment words, such as "excellent", "great", "waste" and so on. These general sentiment words convey strong sentiment orientations. In addition, their sentiment polarities are consistent in different domains. Thus, the global sentiment model extracted from multiple source domains has good generalization ability and is more suitable for domain adaptation than the sentiment model trained in a single source domain, which may contain heavy domain-specific sentiment bias. Second, the domain-specific sentiment models can capture rich specific sentiment expressions in each source domain. For example, "easy" is a positive word in *Kitchen* domain while "return" is a negative word in this domain. Third, different domains have different domain-specific sentiment expressions. For example, "read" is frequently used as a positive word in *Book* domain, while it is a negative word in *DVD* domain. Thus, it is important to separate the global and the domain-specific sentiment knowledge. In addition, although different sentiment expressions are used in different domains, similar domains may share many common domain-specific sentiment expressions. For example, "easy" and "works" are positive words in both *Electronics* and *Kitchen* domains, and "return" and "broken" are both negative words in them. Thus, transferring the domain-specific sentiment models from similar source domains to target domain is helpful. From Fig. 3 we can see that the sentiment polarity relations in the sentiment graph extracted from massive unlabeled data are reasonable. Words with positive relation scores tend to convey similar sentiments, and words with negative relation scores usually convey opposite sentiments. In addition, this sentiment graph contains rich domain-specific sentiment information in target domain, which is useful to transfer the sentiment knowledge from multiple source domains to target domain. For example, "excellent", "easy", "simple", and "quick" share the same sentiment polarity in *Electronics* domain according to Fig. 3. We can infer that "easy" is positive in this domain using the sentiment of "excellent" in the global sentiment model and the sentiment relation between "easy" and "excellent". Then we can further infer the sentiments of the domain-specific sentiment words

Table 3: The top words in the global and domain-specific sentiment models.

Global	Positive	excellent, great, best, perfect, love, wonderful, the_best, loved, well, fantastic, enjoy, favorite
	Negative	bad, waste, boring, disappointed, worst, poor, disappointing, disappointment, terrible, poorly
Book	Positive	excellent, wonderful, easy, loved, enjoyable, life, fun, favorite, a_must, read, important, novel
	Negative	no, boring, disappointing, bad, instead, waste, little, writing, poorly, pages, unfortunately
DVD	Positive	enjoy, hope, loved, season, better_than, best, a_must, first, superman, classic, times, back
	Negative	worst, boring, bad, the_worst, terrible, waste, awful, book, horrible, dull, lame, read, hard
Kitchen	Positive	easy, great, perfect, love, works, easy_to, best, little, well, good, nice, long, durable, clean
	Negative	disappointed, back, poor, broken, too, return, off, returned, broke, waste, tried, times, doesn't
Electronics	Positive	excellent, great, perfect, best, love, easy_to, easy, little, the_best, works, good, nice, wonderful
	Negative	disappointed, poor, waste, too, bad, worst, back, broken, return, horrible, off, tried, poorly

"simple" and "quick" in target domain using the polarity of "easy" and their sentiment relations with it, even if they may be covered by no source domain.

6 Conclusion

This paper presents a sentiment domain adaptation approach which transfers the sentiment knowledge from multiple source domains to target domain. Our approach consists of two steps. First, we extract both global and domain-specific sentiment knowledge from the data of multiple source domains. Second, we transfer these two kinds of sentiment knowledge to target domain with the help of the words' sentiment graph. We proposed to build words' sentiment graph for target domain by extracting their sentiment polarity relations from massive unlabeled data. Besides, we proposed a novel domain similarity measure based on sentiment graphs, and incorporated the domain similarities between target and different source domains into the domain adaptation process. The experimental results on a benchmark dataset show that our approach can effectively improve the performance of cross-domain sentiment classification.

Acknowledgements

This research is supported by the Key Program of National Natural Science Foundation of China (Grant nos. U1536201 and U1405254), the National Natural Science Foundation of China (Grant no. 61472092), the National High Technology Research and Development Program of China (863 Program) (Grant no. 2015AA020101), the National Science and Technology Support Program of China (Grant no. 2014BAH41B00), and the Initiative Scientific Research Program of Tsinghua University.

References

John Blitzer, Mark Dredze, Fernando Pereira, et al. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *ACL:HLT*, pages 132–141.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Rita Chattopadhyay, Jieping Ye, Sethuraman Panchanathan, Wei Fan, and Ian Davidson. 2011. Multi-source domain adaptation and its application to early detection of fatigue. In *KDD*, pages 717–725. ACM.

Xi Chen, Qihang Lin, Seyoung Kim, Jaime G Carbonell, Eric P Xing, et al. 2012. Smoothing proximal gradient method for general structured sparse regression. *The Annals of Applied Statistics*, 6(2):719–752.

Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, pages 289–296. ACM.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *KDD*, pages 109–117. ACM.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *ACL:HLT*, pages 123–131.

Tao Li, Vikas Sindhwani, Chris Ding, and Yi Zhang. 2009. Knowledge transformation for cross-domain sentiment classification. In *SIGIR*, pages 716–717. ACM.

Shoushan Li, Yunxia Xue, Zhongqing Wang, and Guodong Zhou. 2013. Active learning for cross-domain sentiment classification. In *IJCAI*, pages 2127–2133.

Jun Liu, Shuiwang Ji, and Jieping Ye. 2009. Multi-task feature learning via efficient l2,1-norm minimization. In *UAI*, pages 339–348.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. In *NIPS*, pages 1041–1048.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *TKDE*, 22(10):1345–1359.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *WWW*, pages 751–760. ACM.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Robert Remus. 2012. Domain adaptation using domain similarity-and domain complexity-based instance selection for cross-domain sentiment analysis. In *2012 IEEE 12th International Conference on Data Mining Workshops*, pages 717–723. IEEE.

Robert Tibshirani. 1996. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Fangzhao Wu and Yongfeng Huang. 2015. Collaborative multi-domain sentiment classification. In *ICDM*, pages 459–468. IEEE.

Fangzhao Wu and Yongfeng Huang. 2016. Personalized microblog sentiment classification via multi-task learning. In *AAAI*, pages 3059–3065.

Fangzhao Wu, Yangqiu Song, and Yongfeng Huang. 2015. Microblog sentiment classification with contextual knowledge regularization. In *AAAI*, pages 2332–2338.

Yasuhisa Yoshida, Tsutomu Hirao, Tomoharu Iwata, Masaaki Nagata, and Yuji Matsumoto. 2011. Transfer learning for multiple-domain sentiment analysisłidentifying domain dependent/independent word polarity. In *AAAI*, pages 1286–1291.

Hui Zou and Trevor Hastie. 2003. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320.

Connotation Frames: A Data-Driven Investigation

Hannah Rashkin **Sameer Singh** **Yejin Choi**
Computer Science & Engineering
University of Washington
{hrashkin, sameer, yejin}@cs.washington.edu

Abstract

Through a particular choice of a predicate (e.g., "x violated y"), a writer can subtly connote a range of implied sentiment and presupposed facts about the entities x and y: (1) *writer's perspective*: projecting x as an "antagonist" and y as a "victim", (2) *entities' perspective*: y probably dislikes x, (3) *effect*: something bad happened to y, (4) *value*: y is something valuable, and (5) *mental state*: y is distressed by the event.

We introduce *connotation frames* as a representation formalism to organize these rich dimensions of connotation using typed relations. First, we investigate the feasibility of obtaining connotative labels through crowdsourcing experiments. We then present models for predicting the connotation frames of verb predicates based on their distributional word representations and the interplay between different types of connotative relations. Empirical results confirm that connotation frames can be induced from various data sources that reflect how language is used in context. We conclude with analytical results that show the potential use of connotation frames for analyzing subtle biases in online news media.

1 Introduction

People commonly express their opinions through subtle and nuanced language (Thomas et al., 2006; Somasundaran and Wiebe, 2010). Often, through seemingly objective statements, the writer can influence the readers' judgments toward an event and their participants. Even by choosing a particular predicate, the writer can indicate rich connotative information about the entities that interact through the predicate. More specifically, through a simple statement such as "x violated y", the writer can convey:

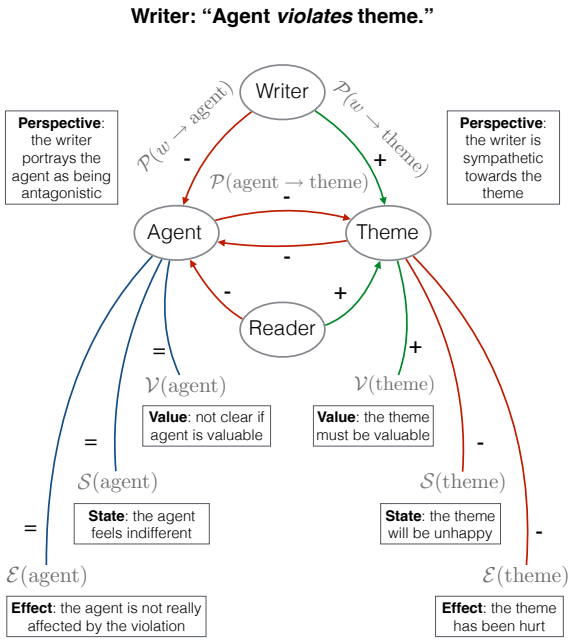


Figure 1: An example connotation frame of "violate" as a set of typed relations: perspective $\mathcal{P}(x \rightarrow y)$, effect $\mathcal{E}(x)$, value $\mathcal{V}(x)$, and mental state $\mathcal{S}(x)$.

(1) writer's perspective: the writer is projecting x as an "antagonist" and y as a "victim", eliciting negative perspective from readers toward x (i.e., blaming x) and positive perspective toward y (i.e., sympathetic or supportive toward y).

(2) entities' perspective: y most likely feels negatively toward x as a result of being violated.

(3) effect: something bad happened to y.

(4) value: y is something valuable, since it does not make sense to violate something worthless. In other words, the writer is presupposing a positive value of y as a fact.

(5) mental state: y is most likely unhappy about the outcome.[1]

[1]To be more precise, y is most likely in a negative state

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 311–321,
Berlin, Germany, August 7-12, 2016.

Verb	Subset of Typed Relations		Example Sentences	L/R
suffer	$\mathcal{P}(w \rightarrow \text{agent}) = +$ $\mathcal{P}(w \rightarrow \text{theme}) = -$ $\mathcal{P}(\text{agent} \rightarrow \text{theme}) = -$	$\mathcal{E}(\text{agent}) = -$ $\mathcal{V}(\text{agent}) = +$ $\mathcal{S}(\text{agent}) = -$	The story begins in Illinois in 1987, when *a 17-year-old girl* **suffered** *a botched abortion.*	R
guard	$\mathcal{P}(w \rightarrow \text{agent}) = +$ $\mathcal{P}(w \rightarrow \text{theme}) = +$ $\mathcal{P}(\text{agent} \rightarrow \text{theme}) = +$	$\mathcal{E}(\text{theme}) = +$ $\mathcal{V}(\text{theme}) = +$ $\mathcal{S}(\text{theme}) = +$	In August, *marshals* **guarded** *25 clinics* in 18 cities.	L
uphold	$\mathcal{P}(w \rightarrow \text{theme}) = +$ $\mathcal{P}(\text{agent} \rightarrow \text{theme}) = +$	$\mathcal{E}(\text{theme}) = +$ $\mathcal{V}(\text{theme}) = +$	A hearing is scheduled to make a decision on whether to **uphold** *the clinic's suspension.*	R

Table 1: Example typed relations (perspective $\mathcal{P}(x \rightarrow y)$, effect $\mathcal{E}(x)$, value $\mathcal{V}(x)$, and mental state $\mathcal{S}(x)$). Not all typed relations are shown due to space constraints. The example sentences demonstrate the usage of the predicates in left [L] or right [R] leaning news sources.

Even though the writer might not explicitly state any of the interpretation [1-5] above, the readers will be able interpret these intentions as a part of their comprehension. In this paper, we present an empirical study of how to represent and induce the connotative interpretations that can be drawn from a verb predicate, as illustrated above.

We introduce *connotation frames* as a representation framework to organize the rich dimensions of the implied sentiment and presupposed facts. Figure 1 shows an example of a connotation frame for the predicate *violate*. We define four different typed relations: $\mathcal{P}(x \rightarrow y)$ for perspective of x towards y, $\mathcal{E}(x)$ for effect on x, $\mathcal{V}(x)$ for value of x, and $\mathcal{S}(x)$ for mental state of x. These relationships can all be either positive (+), neutral (=), or negative (-).

Our work is the first study to investigate frames as a representation formalism for connotative meanings. This contrasts with previous computational studies and resource development for frame semantics, where the primary focus was almost exclusively on denotational meanings of language (Baker et al., 1998; Palmer et al., 2005). Our formalism draws inspirations from the earlier work of frame semantics, however, in that we investigate the connection between a word and the related world knowledge associated with the word (Fillmore, 1976), which is essential for the readers to interpret many layers of the implied sentiment and presupposed value judgments.

We also build upon the extensive amount of literature in sentiment analysis (Pang and Lee, 2008; Liu and Zhang, 2012), especially the recent emerging efforts on implied sentiment analysis (Feng et al., 2013; Greene and Resnik, 2009), entity-entity sentiment inference (Wiebe and Deng, 2014), opinion role induction (Wiegand and Ruppenhofer, 2015) and effect analysis (Choi and Wiebe, 2014). However, our work is the first to organize various aspects of the connotative information into coherent frames.

More concretely, our contributions are threefold: (1) a new formalism, model, and annotated dataset for studying connotation frames from large-scale natural language data and statistics, (2) new data-driven insights into the dynamics among different typed relations within each frame, and (3) an analytic study showing the potential use of connotation frames for analyzing subtle biases in journalism.

The rest of the paper is organized as follows: in §2, we provide the definitions and data-driven insights for connotation frames. In §3, we introduce models for inducing the connotation frames, followed by empirical results, annotation studies, and analysis on news media in §4. We discuss related work in §5 and conclude in §6.

2 Connotation Frame

Given a predicate v, we define a connotation frame $\mathcal{F}(v)$ as a collection of typed relations and their polarity assignments: (i) **perspective** $\mathcal{P}^v(a_i \rightarrow a_j)$: a directed sentiment from the entity a_i to the entity a_j, (ii) **value** $\mathcal{V}^v(a_i)$: whether a_i is presupposed to be valuable, (iii) **effect** $\mathcal{E}^v(a_i)$: whether the event denoted by the predicate v is good or bad for the entity a_i, and (iv) **mental state** $\mathcal{S}^v(a_i)$: the likely mental state of the entity a_i as a result of the event. We assume that each typed relation can have one of the three connotative polarities $\in \{+, -, =\}$, i.e., positive, negative, or neutral. Our goal in this paper is to focus on the general connotation of the predicate considered out of context. We leave contextual interpretation of connotation as future work.

Table 1 shows examples of connotation frame

assuming it is an entity that can have a mental state.

Verb	x's role	$\mathcal{P}(\mathbf{w} \rightarrow \cdot)$	Left-leaning Sources	Right-leaning Sources
accuse	agent	-	Putin, Progressives, Limbaugh, Gingrich	activist, U.S., protestor, Chavez
	theme	+	official, rival, administration, leader	Romney, Iran, Gingrich, regime
attack	agent	-	McCain, Trump, Limbaugh	**Obama**, campaign, Biden, Israel
	theme	+	Gingrich, **Obama**, policy	citizen, Zimmerman
criticize	agent	-	Ugandans, rival, Romney, Tyson	Britain, passage, **Obama**, Maddow
	theme	+	**Obama**, Allen, Cameron, Congress	Pelosi, Romey, GOP, Republicans

Table 2: Media Bias in Connotation Frames: Obama, for example, is portrayed as someone who *attacks* or *criticizes* others by the right-leaning sources, whereas the left-leaning sources portray Obama as the victim of harsh acts like "attack" and "criticize".

relations for the verbs *suffer*, *guard*, and *uphold*, along with example sentences. For instance, for the verb *suffer*, the writer is likely to have a positive perspective towards the agent (e.g., being supportive or sympathetic toward the "17-year-old girl" in the example shown on the right) and a negative perspective towards the theme (e.g., being negative towards 'botched abortion").

2.1 Data-driven Motivation

Since the meaning of language is ultimately contextual, the exact connotation will vary depending on the context of each utterance. Nonetheless, there still are common shifts or biases in the connotative polarities, as we found from two data-driven analyses.

First, we looked at words from the Subjectivity Lexicon (Wilson et al., 2005) that are used in the argument positions of a small selection of predicates in Google Syntactic N-grams (Goldberg and Orwant, 2013). For this analysis, we assumed that the word in the subject position is the agent while the object is the theme. We found 64% of the words in the agent position of *suffer* are positive, and 94% of the words in the theme position are negative, which is consistent with the polarities of the writer's perspective towards these arguments, as shown in Table 1. For *guard*, 57% of the subjects and 76% of the objects are positive, and in the case of *uphold*, 56% of the subjects and 72% of the objects are positive.

We also investigated how media bias can potentially be analyzed through connotation frames. From the Stream Corpus 2014 dataset (KBA, 2014), we selected all articles from news outlets with known political biases,[2] and compared how they use polarised words such as "accuse", "attack", and "criticize" differently in light of $\mathcal{P}(w \rightarrow agent)$ and $\mathcal{P}(w \rightarrow theme)$ relations of the connotation frames. Table 2 shows interesting contrasts. Obama, for example, is portrayed as someone who *attacks* or *criticizes* others according to the right-leaning sources, whereas the left-leaning sources portray Obama as the victim of harsh acts like "attack" or "criticize".[3] Furthermore, by knowing the perspective relationships $\mathcal{P}(w \rightarrow a_i)$ associated with a predicate, we can make predictions about how the left-leaning and right-leaning sources feel about specific people or issues. For example, because left-leaning sources frequently use McCain, Trump, and Limbaugh in the subject position of attack, we might predict that these sources have a negative sentiment towards these entities.

2.2 Dynamics between Typed Relations

Given a predicate, the polarity assignments of typed relations are interdependent. For example, if the writer feels positively towards the agent but negatively towards the theme, then it is likely that the agent and the theme do not feel positively towards each other. This insight is related to that of Wiebe and Deng (2014), but differs in that the polarities are predicate-specific and do not rely on knowledge of prior sentiment towards the arguments. This and other possible interdependencies are summarized in Table 3. These interdependencies serve as general guidelines of what properties we expect to depend on one another, especially in the case where the polarities are non-neutral. We will promote these internal consistencies in our factor graph model (§3) as soft constraints.

There also exist other interdependencies that we will use to simplify our task. First, the directed

[2]The articles come from 30 news sources indicated by others as exhibiting liberal or conservative leanings (Mitchell et al., 2014; Center for Media and Democracy, 2013; Center for Media and Democracy, 2012; HWC Library, 2011)

[3]That is, even if someone truly deserves criticism from Obama, left-learning sources would choose slightly different wordings to avoid a potentially harsh cast on Obama.

Perspective Triad: If A is positive towards B, and B is positive towards C, then we expect A is also positive towards C. Similar dynamics hold for the negative case. $\mathcal{P}_{w\rightarrow a_1} = \neg\left(\mathcal{P}_{w\rightarrow a_2} \oplus \mathcal{P}_{a_1\rightarrow a_2}\right)$
Perspective – Effect: If a predicate has a positive effect on the Subject, then we expect that the interaction between the Subject and Object was positive. Similar dynamics hold for the negative case and for other perspective relations. $\mathcal{E}_{a_1} = \mathcal{P}_{a_2\rightarrow a_1}$
Perspective – Value: If A is presupposed as valuable, then we expect that the writer also views A positively. Similar dynamics hold for the negative case. $\mathcal{V}_{a_1} = \mathcal{P}_{w\rightarrow a_1}$
Effect – Mental State: If the predicate has a positive effect on A, then we expect that A will gain a positive mental state. Similar dynamics hold for the negative case. $\mathcal{S}_{a_1} = \mathcal{E}_{a_1}$

Table 3: Potential Dynamics among Typed Relations: we propose models that parameterize these dynamics using log-linear models (frame-level model in §3).

sentiments between the agent and the theme are likely to be reciprocal, or at least do not directly conflict with + and − simultaneously. Therefore, we assume that $\mathcal{P}(a_1 \rightarrow a_2) = \mathcal{P}(a_2 \rightarrow a_1) = \mathcal{P}(a_1 \leftrightarrow a_2)$, and we only measure for these binary relationships going in one direction. In addition, we assume the *predicted*[4] perspective from the reader r to an argument $\mathcal{P}(r \rightarrow a)$ is likely to be the same as the *implied* perspective from the writer w to the same argument $\mathcal{P}(w \rightarrow a)$. So, we only try to learn the perspective of the writer. Lifting these assumptions will be future work.

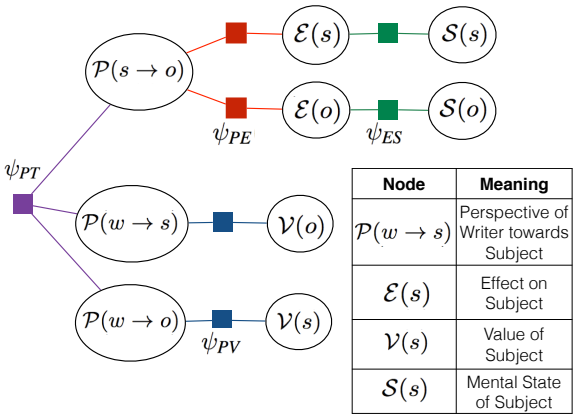
For simplicity, our model only explores the polarities involving the agent and the theme roles. We will assume that these roles are correlated to the subject and object positions, and henceforth refer to them as the "Subject" and "Object" of the event.

3 Modeling Connotation Frames

Our task is essentially that of lexicon induction (Akkaya et al., 2009; Feng et al., 2013) in that we want to induce the connotation frames of previously unseen verbs. For each predicate, we infer a connotation frame composed of 9 relationship aspects that represent: *perspective* $\{\mathcal{P}(w \rightarrow o), \mathcal{P}(w \rightarrow s), \mathcal{P}(s \rightarrow o)\}$, *effect* $\{\mathcal{E}(o), \mathcal{E}(s)\}$, *value* $\{\mathcal{V}(o), \mathcal{V}(s)\}$, and *mental state* $\{\mathcal{S}(o), \mathcal{S}(s)\}$ polarities.

We propose two models: an aspect-level model that makes the prediction for each typed relation independently based on the distributional representation of the context in which the predicate appears (§3.1), and a frame-level model that makes the prediction over the connotation frame collectively in consideration the dynamics between typed relations (§3.2).

[4]Surely different readers can and will form varying opinions after reading the same text. Here we concern with the most likely perspective of the general audience, as a result of reading the text.



Node	Meaning
$\mathcal{P}(w \rightarrow s)$	Perspective of Writer towards Subject
$\mathcal{E}(s)$	Effect on Subject
$\mathcal{V}(s)$	Value of Subject
$\mathcal{S}(s)$	Mental State of Subject

Figure 2: A factor graph for predicting the polarities of the typed relations that define a connotation frame for a given verb predicate. The factor graph also includes unary factors (ψ_{emb}), which we left out for brevity.

3.1 Aspect-Level

Our aspect-level model predicts labels for each of these typed relations separately. As input, we use the 300-dimensional dependency-based word embeddings from Levy and Goldberg (2014). For each aspect, there is a separate MaxEnt (maximum entropy) classifier used to predict the label of that aspect on a given word-embedding, which is treated as a 300 dimensional input vector to the classifier. The MaxEnt classifiers learn their weights using LBFGS on the training data examples with re-weighting of samples to maximize for the best average F1 score.

3.2 Frame-Level

Next we present a factor graph model (Figure 2) of the connotation frames that parameterize the dynamics between typed relations. Specifically, for each verb predicate,[5] the factor graph contains 9 nodes representing the different aspects of the connotation frame. All these variables take polarity values from the set $\{-, =, +\}$.

We define $\mathbf{Y}_i := \{\mathcal{P}_{wo}, \mathcal{P}_{ws}, \mathcal{P}_{so}, \mathcal{E}_o, \mathcal{E}_s, \mathcal{V}_o, \mathcal{V}_s, \mathcal{S}_o, \mathcal{S}_s\}$ as the set of relational aspects for the i^{th} verb. The factor graph for $\mathbf{Y}_i$, is illustrated in Figure 2, and we will describe the factor potentials in more detail in the rest of this section. The probability of an assignment of polarities to the nodes in $\mathbf{Y}_i$ is:

$$\begin{aligned} P(\mathbf{Y}_i) \propto\ & \psi_{\text{PV}}(\mathcal{P}_{ws}, \mathcal{V}_s)\ \psi_{\text{PV}}(\mathcal{P}_{wo}, \mathcal{V}_o) \\ & \psi_{\text{PE}}(\mathcal{P}_{so}, \mathcal{E}_s)\ \psi_{\text{PE}}(\mathcal{P}_{so}, \mathcal{E}_o) \\ & \psi_{\text{ES}}(\mathcal{E}_s, \mathcal{S}_s)\ \psi_{\text{ES}}(\mathcal{E}_o, \mathcal{S}_o) \\ & \psi_{\text{PT}}(\mathcal{P}_{wo}, \mathcal{P}_{ws}, \mathcal{P}_{so}) \prod_{y \in \mathbf{Y}_i} \psi_{emb}(y) \end{aligned}$$

Embedding Factors We include unary factors on all nodes to represent the results of the aspect-level classifier. Incorporating this knowledge as factors, as opposed to *fixing* the variables as observed, affords us the flexibility of representing noise in the labels as soft evidence. The potential function ψ_{emb} is a log-linear function of a feature vector f, which is a one-hot feature vector representing the polarity of a node (+,−,or =). For example, with the node representing the value of the object ($\mathcal{V}_o$):

$$\psi_{emb}(\mathcal{V}_o) = e^{w_{\mathcal{V}_o} \cdot f(\mathcal{V}_o)}$$

The potential ψ_{emb} is defined similarly for the other 8 remaining nodes. All weights were learned using stochastic gradient descent (SGD) over training data.

Interdependency Factors We include interdependency factors to promote the properties defined by the dynamics between relations (§2.2). The potentials for Perspective Triad, Perspective-Value, Perspective-Effect, and Effect-State Relationships (ψ_{PT}, ψ_{PV}, ψ_{PE}, ψ_{ES} respectively) are all defined using log-linear functions of one-hot feature vectors that encode the combination of polarities of the neighboring nodes. The potential for ψ_{PT} is therefore:

$$\psi_{PT}(\mathcal{P}_{wo}, \mathcal{P}_{ws}, \mathcal{P}_{so}) = e^{w_{PT} \cdot f(\mathcal{P}_{wo}, \mathcal{P}_{ws}, \mathcal{P}_{so})}$$

[5] We consider only verb predicates here.

And we define the potentials for ψ_{PV}, ψ_{PE}, and ψ_{ES} for subject nodes as:

$$\psi_{PV}(\mathcal{P}_{ws}, \mathcal{V}_s) = e^{w_{PV,s} \cdot f(\mathcal{P}_{ws}, \mathcal{V}_s)}$$

$$\psi_{PE}(\mathcal{P}_{so}, \mathcal{E}_s) = e^{w_{PE,s} \cdot f(\mathcal{P}_{so}, \mathcal{E}_s)}$$

$$\psi_{ES}(\mathcal{E}_s, \mathcal{S}_s) = e^{w_{ES,s} \cdot f(\mathcal{E}_s, \mathcal{S}_s)}$$

and we define the potentials for the object nodes similarly. As with the unary seed factors, weights were learned using SGD over training data.

Belief Propagation We use belief propagation to induce the connotation frames of previously unseen verbs. In the belief propagation algorithm, messages are iteratively passed between the nodes to their neighboring factors and vice versa. Each message μ, containing a scalar for each value $x \in \{-, 0, +\}$, is defined from each node v to a neighboring factor a as follows:

$$\mu_{v \to a}(x) \propto \prod_{a^* \in N(v)\ a} \mu_{a^* \to v}(x)$$

and from each factor a to a neighboring node v as:

$$\mu_{a \to v} \propto \sum_{x', x'_v = x} \psi(x') \prod_{v^* \in N(a)\ v} \mu_{v^* \to a}(x'_{v^*})$$

At the conclusion of message passing, the probability of a specific polarity associated with node v being equal to x is proportional to $\prod_{a \in N(v)} \mu_{a \to v}(x)$. Our factor graph does not contain any loops, so we are able to perform exact inference.

4 Experiments

We first describe crowd-sourced annotations (§4.1), then present the empirical results of predicting connotation frames (§4.2), and conclude with qualitative analysis of a large corpus (§4.3).

4.1 Data and Crowdsourcing

In order to understand how humans interpret connotation frames, we designed an Amazon Mechanical Turk (AMT) annotation study. We gathered a set of transitive verbs commonly used in the New York Times corpus (Sandhaus, 2008), selecting the 2400 verbs that are used more than 200 times in the corpus. Of these, AMT workers annotated the 1000 most frequently used verbs.

Annotation Design In a pilot annotation experiment, we found that annotators have difficulty thinking about subtle connotative polarities when shown predicates without any context. Therefore,

we designed the AMT task to provide a generic context as follows. We first split each verb predicate into 5 separate tasks that each gave workers a different generic sentence using the verb. To create generic sentences, we used Google Syntactic N-grams (Goldberg and Orwant, 2013) to come up with a frequently seen Subject-Verb-Object tuple which served as a simple three-word sentence with generic arguments. For each of the 5 sentences, we asked 3 annotators to answer questions like "How do you think the Subject feels about the event described in this sentence?" In total, each verb has 15 annotations aggregated over 5 different generic sentences containing the verb.

In order to help the annotators, some of the questions also allowed annotators to choose sentiment using additional classes for "positive or neutral" or "negative or neutral" for when they were less confident but still felt like a sentiment might exist. When taking inter-annotator agreement, we count "positive or neutral" as agreeing with either "positive" or "neutral" classes.

Annotator agreement Table 4 shows agreements and data statistics. The non-conflicting (NC) agreement only counts opposite polarities as disagreement.[6] From this study, we can see that non-expert annotators are able to see these sort of relationships based on their understanding of how language is used. From the NC agreement, we see that annotators do not frequently choose completely opposite polarities, indicating that even when they disagree, their disagreements are based on the degree of connotations rather than the polarity itself. The average Krippendorff alpha for all of the questions posed to the workers is 0.25, indicating stronger than random agreement. Considering the subtlety of the implicit sentiments that we are asking them to annotate, it is reasonable that some annotators will pick up on more nuances than others. Overall, the percent agreement is encouraging that the connotative relationships are visible to human annotators.

Aggregating Annotations We aggregated over crowdsourced labels (fifteen annotations per verb) to create a polarity label for each aspect of a verb.[7]

Final distributions of the aggregated labels are included in the right-hand columns of Table 4.

Aspect	% Agreement		Distribution	
	Strict	NC	% +	% -
$\mathcal{P}(w \rightarrow o)$	75.6	95.6	36.6	4.6
$\mathcal{P}(w \rightarrow s)$	76.1	95.5	47.1	7.9
$\mathcal{P}(s \rightarrow o)$	70.4	91.9	45.8	5.0
$\mathcal{E}(o)$	52.3	94.6	50.3	20.24
$\mathcal{E}(s)$	53.5	96.5	45.1	4.7
$\mathcal{V}(o)$	65.2	-	78.64	2.7
$\mathcal{V}(s)$	71.9	-	90.32	1.4
$\mathcal{S}(o)$	79.9	98.0	12.8	14.5
$\mathcal{S}(s)$	70.4	92.5	50.72	8.6

Table 4: Label Statistics: % Agreement refers to pairwise inter-annotator agreement. The strict agreement counts agreement over 3 classes ("positive or neutral" was counted as agreeing with either + or neutral), while non-conflicting (NC) agreement also allows agreements between neutral and -/+ (no direct conflicts). Distribution shows the final class distribution of -/+ labels created by averaging annotations.

Notably, the distributions are skewed toward positive and neutral labels. The most skewed connotation frame aspect is the value $\mathcal{V}(x)$ which tends to be positive, especially for the subject argument. This makes some intuitive sense since, as the subject actively causes the predicate event to occur, they most likely have some intrinsic potential to be valuable. An example of a verb where the subject was labelled as not valuable is "contaminate". In the most generic case, the writer is using contaminate to frame the subject as being worthless (and even harmful) with regards to the other event participants. For example, in the sentence "his touch contaminated the food," it is clear that the writer considers "his touch" to be of negative value in the context of how it impacts the rest of the event.

4.2 Connotation Frame Prediction

Using our crowdsourced labels, we randomly divided the annotated verbs into training, dev, and held-out test sets of equal size (300 verbs each). For evaluation we measured average accuracy and F1 score over the 9 different Connotation Frame relationship types for which we have annotations: $\mathcal{P}(w \rightarrow o)$, $\mathcal{P}(w \rightarrow s)$, $\mathcal{P}(s \rightarrow o)$, $\mathcal{V}(o)$, $\mathcal{V}(s)$, $\mathcal{E}(o)$, $\mathcal{E}(s)$, $\mathcal{S}(o)$, and $\mathcal{S}(s)$.

Baselines To show the non-trivial challenge of learning Connotation Frames, we include a simple majority-class baselines. The MAJORITY classifier assigns each of the 9 relationships the label of the majority of that relationship type found in the training data. Some of these relationships (in particular, the Value of subject/object) have skewed

[6] Annotators were asked yes/no questions related to Value, so this does not have a corresponding NC agreement score.

[7] We take the average to obtain scalar value between $[-1., 1.]$ for each aspect of a verb's connotation frame. For simplicity, we cutoff the ranges of negative, neutral and positive polarities as $[-1, -0.25)$, $[-0.25, 0.25]$ and $(0.25, 1]$, respectively.

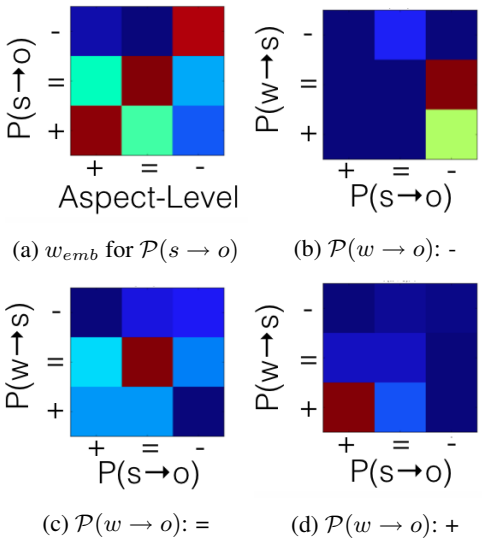
distributions, so we expect this classifier to achieve a much higher accuracy than random but a much lower overall F1 score.

Additionally, we add a GRAPH PROP baseline that is comparable to algorithms like graph propagation or label propagation which are often used for (sentiment) lexicon induction. We use a factor graph with nodes representing the polarity of each typed relation for each verb. Binary factors connect nodes representing a particular type of relation for two similar verbs (e.g. $\mathcal{P}(w \rightarrow o)$ for verbs *persuade* and *convince*). These binary factors have hand-tuned potentials that are proportional to the cosine similarity of the verbs' embeddings, encouraging similar verbs to have the same polarity for the various relational aspects. We use words in the training data as the seed set and use loopy belief propagation to propagate polarities from known nodes to the unknown relationships.

Finally, we use a 3-NEAREST NEIGHBOR baseline that labels relationships for a verb based on the predicate's 300-dimensional word embedding representation, using the same embeddings as in our aspect-level. 3-NEAREST NEIGHBOR labels each verb using the polarities of the three closest verbs found in the training set. The most similar verbs are determined using the cosine similarity between word embeddings.

Results As shown in Table 5, aspect-level and frame-level models consistently outperform all three baselines — MAJORITY, 3-NN, GRAPH PROP in the development set across the different types of relationships. In particular, the improved F1 scores show that these models are able to perform better across all three classes of labels even in the most skewed cases. The frame-level model also frequently improves the F1 scores of the labels from what they were in the aspect-level model. The summarized comparison of the classifiers' performance test set is shown in Table 6. As with the development set, aspect-level and frame-level are both able to outperform the baselines. Furthermore, the frame-level formulation is able to make improvement over the results of the aspect-level classification, indicating that the modelling of inter-dependencies between relationships did help correct some of the mistakes made.

One point of interest about the frame-level results is whether the learned weights over the consistency factors match our initial intuitions about interdependencies between relationships. The weights



(a) w_{emb} for $\mathcal{P}(s \rightarrow o)$ (b) $\mathcal{P}(w \rightarrow o)$: -

(c) $\mathcal{P}(w \rightarrow o)$: = (d) $\mathcal{P}(w \rightarrow o)$: +

Figure 3: Learned weights of embedding factor for the perspective of subject to object and the weights the perspective triad (PT) factor. Red is for weights that are more positive, whereas blue are more negative.

learned in our algorithm do tell us something interesting about the degree to which these interdependencies are actually found in our data.

We show the heat maps for some of the learned weights in Figure 3. In 3a, we show the weights of one of the embedding factors, and how the polarities are more strongly weighted when they match the relation-level output. In the rest of the figure, we show the weights for the other perspective relationships when $\mathcal{P}(w \rightarrow o)$ is negative (3b), neutral (3c), and positive (3d), respectively. Based on the expected interdependencies, when $\mathcal{P}(w \rightarrow o) : -$, the model should favor $\mathcal{P}(w \rightarrow s) \neq \mathcal{P}(s \rightarrow o)$ and when $\mathcal{P}(w \rightarrow o) : +$, the model should favor $\mathcal{P}(w \rightarrow s) = \mathcal{P}(s \rightarrow o)$. Our model does, in fact, learn a similar trend, with slightly higher weights along these two diagonals in the maps 3b and 3d. Interestingly, when $\mathcal{P}(w \rightarrow o)$ is neutral, weights slightly prefer for the other two perspectives to resemble one another, but with highest weights being when other perspectives are also neutral.

4.3 Analysis of a Large News Corpus

Using the connotation frame, we present measured implied sentiment in online journalism.

Data From the Stream Corpus (KBA, 2014), we select 70 million news articles. We extract subject-verb-object relations for this subset using the direct dependencies between noun phrases

Aspect	Algorithm	Acc.	Avg F_1
$\mathcal{P}(w \rightarrow o)$	Majority	56.52	24.07
	Graph Prop	59.53	50.20
	3-nn	62.88	47.93
	Aspect-Level	67.56	56.18
	Frame-Level	67.56	56.18
$\mathcal{P}(w \rightarrow s)$	Majority	49.83	22.17
	Graph Prop	52.84	42.93
	3-nn	55.18	45.88
	Aspect-Level	60.54	60.72
	Frame-Level	61.87	63.07
$\mathcal{P}(s \rightarrow o)$	Majority	49.83	22.17
	Graph Prop	52.17	46.57
	3-nn	56.52	52.94
	Aspect-Level	63.21	61.70
	Frame-Level	63.88	62.56
$\mathcal{E}(o)$	Majority	48.83	21.87
	Graph Prop	54.85	51.40
	3-nn	55.18	51.53
	Aspect-Level	64.21	63.63
	Frame-Level	65.22	64.67
$\mathcal{E}(s)$	Majority	49.83	22.17
	Graph Prop	52.17	35.56
	3-nn	54.85	42.63
	Aspect-Level	62.54	53.82
	Frame-Level	63.88	56.81
$\mathcal{V}(o)$	Majority	79.60	29.55
	Graph Prop	71.91	35.10
	3-nn	76.25	39.09
	Aspect-Level	75.92	45.45
	Frame-Level	76.25	48.13
$\mathcal{V}(s)$	Majority	89.30	31.45
	Graph Prop	84.62	38.82
	3-nn	85.62	38.45
	Aspect-Level	87.96	48.06
	Frame-Level	87.96	48.06
$\mathcal{S}(o)$	Majority	71.91	27.89
	Graph Prop	69.90	55.57
	3-nn	72.91	59.26
	Aspect-Level	81.61	72.85
	Frame-Level	81.61	72.85
$\mathcal{S}(s)$	Majority	50.84	22.47
	Graph Prop	48.83	35.40
	3-nn	54.85	45.51
	Aspect-Level	61.54	53.88
	Frame-Level	61.54	53.88

Table 5: Detailed breakdown of results on the development set using accuracy and average F1 over the three class labels (+,-,=).

Algorithm	Acc.	Avg F_1
Graph Prop	58.81	41.46
3-nn	63.71	47.30
Aspect-Level	67.93	53.17
Frame-Level	68.26	53.50

Table 6: Performance on the test set. Results are averaged over the different aspects.

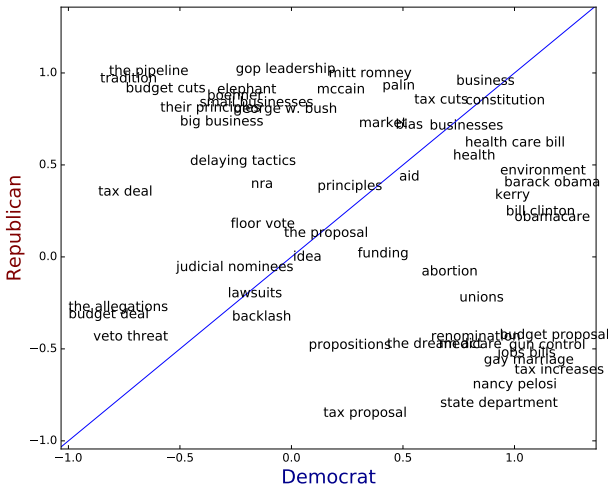


Figure 4: Average sentiment of Democrats and Republicans (as subjects) to selected nouns (as their objects), aggregated over a large corpus using the learned lexicon (§4.2). The line indicates identical sentiments, i.e. Republicans are more positive towards the nouns that are above the line.

and verbs as identified by the BBN Serif system, obtaining 1.2 billion unique tuples of the form *(url,subject,verb,object,count)*.We also extracted subject-verb-object tuples from news articles found in the Annotated English Gigaword Corpus (Napoles et al., 2012), which contains nearly 10 million articles. From the Gigaword corpus we extracted a further 120 million unique tuples.

Estimating Entity Polarities Using connotation frames, we can also measure entity-to-entity sentiment at a large scale. Figure 4, for example, presents the polarity of entities "Democrats" and "Republicans" towards a selected set of nouns, by computing the average estimated polarity (using our lexicon) over triples where one of these entities appears as part of the subject (e.g. "Democrats" or "Republican party"). Apart from nouns that both entities are positive ("business", "constitution") or negative ("the allegations","veto threat") towards, we can also see interesting examples in which Democrats feel more positively (below the line: "nancy pelosi", "unions", "gun control", etc.) and ones where Republicans are more positive ("the pipeline", "gop leaders", "budget cuts", etc.) Also, both entities are neutral towards "idea" and "the proposal", which probably owes to the fact that ideas or proposals can be good or bad for either entity depending on the context.

5 Related Work

Most prior work on sentiment lexicons focused on the overall polarity of words without taking into account their semantic arguments (Wilson et al., 2005; Baccianella et al., 2010; Wiebe et al., 2005; Velikovich et al., 2010; Kaji and Kitsuregawa, 2007; Kamps et al., 2004; Takamura et al., 2005; Adreevskaia and Bergler, 2006). Several recent studies began exploring more specific and nuanced aspects of sentiment such as connotation (Feng et al., 2013), good and bad effects (Choi and Wiebe, 2014), and evoked sentiment (Mohammad and Turney, 2010). Drawing inspirations from them, we present connotation frames as a unifying representation framework to encode the rich dimensions of implied sentiment, presupposed value judgements, and effect evaluation, and propose a factor graph formulation that captures the interplay among different types of connotation relations.

Goyal et al. (2010a; 2010b) investigated how characters (protagonists, villains, victims) in children's stories are affected by certain predicates, which is related to the effect relations studied in this work. While Klenner et al. (2014) similarly investigated the relation between the polarity of the verbs and arguments, our work introduces new perspective types and proposes a unified representation and inference model. Wiegand and Ruppenhofer (2015) also looked at perspective-based relationships induced by verb predicates with a focus on opinion roles. Building on this concept, our framework also incorporates information about the perspectives' polarities as well as information about other typed relations. There have been growing interests for modeling framing (Greene and Resnik, 2009; Hasan and Ng, 2013), biased language (Recasens et al., 2013) and ideology detection (Yano et al., 2010). All these tasks are relatively less studied, and we hope our connotation frame lexicon will be useful for them.

Sentiment inference rules have been explored by the recent work of Wiebe and Deng (2014) and Deng and Wiebe (2014). In contrast, we make a novel conceptual connection between inferred sentiments and frame semantics, organized as connotation frames, and present a unified model that integrates different aspects of the connotation frames. Finally, in a broader sense, what we study as connotation frames draws a connection to schema and script theory (Schank and Abelson, 1975). Unlike most prior work that focused on directly observable actions (Chambers and Jurafsky, 2009; Frermann et al., 2014; Bethard et al., 2008), we focus on implied sentiments that are framed by predicate verbs.

6 Conclusion

In this paper, we presented a novel system of connotative frames that define a set of implied sentiment and presupposed facts for a predicate. Our work also empirically explores different methods of inducing and modelling these connotation frames, incorporating the interplay between relations within frames. Our work suggests new research avenues on learning connotation frames, and their applications to deeper understanding of social and political discourse. All the learned connotation frames and annotations will be shared at `http://homes.cs.washington.edu/~hrashkin/connframe.html`.

Acknowledgements

We thank the anonymous reviewers for many insightful comments. We also thank members of UW NLP for discussions and support. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1256082. The work is also supported in part by NSF grants IIS-1408287, IIS-1524371 and gifts by Google and Facebook.

References

Alina Adreevskaia and Sabine Bergler. 2006. Mining wordnet for fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 209–216.

Cem Akkaya, Janyce Wiebe, and Rada Mihalcea. 2009. Subjectivity word sense disambiguation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, volume 2, pages 190–199.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, volume 1, pages 86–90.

Steven Bethard, William J Corvey, Sara Klingenstein, and James H Martin. 2008. Building a corpus of temporal-causal structure. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*.

Center for Media and Democracy. 2012. Sourcewatch: Conservative news outlets. http://www.sourcewatch.org/index.php/Conservative_news_outlets.

Center for Media and Democracy. 2013. Sourcewatch: Liberal news outlets. http://www.sourcewatch.org/index.php/Liberal_news_outlets.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 2 of *ACL '09*, pages 602–610.

Yoonjung Choi and Janyce Wiebe. 2014. +/-effectwordnet: Sense-level lexicon acquisition for opinion inference. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1181–1191. Association for Computational Linguistics, October.

Lingjia Deng and Janyce Wiebe. 2014. Sentiment propagation via implicature constraints. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Song Feng, Jun Seok Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, volume 1, pages 1774–1784. Association for Computational Linguistics.

Charles J. Fillmore. 1976. Frame semantics and the nature of language. In *In Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech, volume 280, pages 2032*.

Lea Frermann, Ivan Titov, and Manfred Pinkal. 2014. A hierarchical bayesian model for unsupervised induction of script knowledge. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

Yoav Goldberg and Jon Orwant. 2013. A dataset of syntactic-ngrams over time from a very large corpus of english books. In *Second Joint Conference on Lexical and Computational Semantics (*SEM)*, volume 1, pages 241–247, June.

Amit Goyal, Ellen Riloff, and Hal Daumé, III. 2010a. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 77–86.

Amit Goyal, Ellen Riloff, Hal Daumé III, and Nathan Gilbert. 2010b. Toward plot units: Automatic affect state analysis. In *Proceedings of HLT/NAACL Workshop on Computational Approaches to Analysis and Generation of Emotion in Text (CAET)*.

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 503–511.

Kazi Saidul Hasan and Vincent Ng. 2013. Frame semantics for stance classification. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning (CONLL)*, pages 124–132.

HWC Library. 2011. Consider the Source: A Resource Guide to Liberal, Conservative, and Nonpartisan Periodicals. www.ccc.edu/colleges/washington/departments/Documents/PeriodicalsPov.pdf. Compiled by HWC Librarians in January 2011.

Nobuhiro Kaji and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of html documents. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1075–1083.

Jaap Kamps, Maarten Marx, Robert J Mokken, and Maarten De Rijke. 2004. Using wordnet to measure semantic orientations of adjectives. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation(LREC'04)*, volume 4, pages 1115–1118.

TREC KBA. 2014. Knowledge Base Acceleration Stream Corpus. http://trec-kba.org/kba-stream-corpus-2014.shtml.

Manfred Klenner, Michael Amsler, and Nora Hollenstein. 2014. Verb polarity frames: a new resource and its application in target-specific polarity classification. In *Proceedings of KONVENS 2014*, pages 106–115.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 302–308.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*, pages 415–463. Springer.

Amy Mitchell, Jeffrey Gottfried, Jocelyn Kiley, and Katerina Eva Matsa. 2014. Political Polarization & Media Habits. www.journalism.org/2014/10/21/political-polarization-media-habits/. Produced by Pew Research Center in October, 2014.

Saif M Mohammad and Peter D Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Association for Computational Linguistics.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100. Association for Computational Linguistics.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic models for analyzing and detecting biased language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1650–1659.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6(12):e26752.

Roger C Schank and Robert P Abelson. 1975. *Scripts, plans, and knowledge*. Yale University.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124. Association for Computational Linguistics.

Hiroya Takamura, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 327–335.

Leonid Velikovich, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 777–785.

Janyce Wiebe and Lingjia Deng. 2014. An account of opinion implicatures. *CoRR*, abs/1404.6491.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Michael Wiegand and Josef Ruppenhofer. 2015. Opinion holder and target extraction based on the induction of verbal categories. *Proceedings of the 2015 Conference on Computational Natural Language Learning (CoNLL)*, page 215.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354.

Tae Yano, Philip Resnik, and Noah A. Smith. 2010. Shedding (a thousand points of) light on biased language. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 152–158.

Bi-Transferring Deep Neural Networks for Domain Adaptation

Guangyou Zhou[1], **Zhiwen Xie**[1], **Jimmy Xiangji Huang**[2], **and Tingting He**[1]
[1] School of Computer, Central China Normal University, Wuhan 430079, China
[2] School of Information Technology, York University, Toronto, Canada
{gyzhou,xiezhiwen,tthe}@mail.ccnu.edu.cn jhuang@yorku.ca

Abstract

Sentiment classification aims to automatically predict sentiment polarity (e.g., positive or negative) of user generated sentiment data (e.g., reviews, blogs). Due to the mismatch among different domains, a sentiment classifier trained in one domain may not work well when directly applied to other domains. Thus, domain adaptation for sentiment classification algorithms are highly desirable to reduce the domain discrepancy and manual labeling costs. To address the above challenge, we propose a novel domain adaptation method, called Bi-Transferring Deep Neural Networks (BTDNNs). The proposed BTDNNs attempts to transfer the source domain examples to the target domain, and also transfer the target domain examples to the source domain. The linear transformation of BTDNNs ensures the feasibility of transferring between domains, and the distribution consistency between the transferred domain and the desirable domain is constrained with a linear data reconstruction manner. As a result, the transferred source domain is supervised and follows similar distribution as the target domain. Therefore, any supervised method can be used on the transferred source domain to train a classifier for sentiment classification in a target domain. We conduct experiments on a benchmark composed of reviews of 4 types of Amazon products. Experimental results show that our proposed approach significantly outperforms the several baseline methods, and achieves an accuracy which is competitive with the state-of-the-art method for domain adaptation.

1 Introduction

With the rise of social media (e.g., blogs and social networks etc.), more and more user generated sentiment data have been shared on the Web (Pang et al., 2002; Pang and Lee, 2008; Liu, 2012; Zhou et al., 2011). They exist in the form of user reviews on shopping or opinion sites, in posts of blogs/questions or customer feedbacks. This has created a surge of research in sentiment classification (or sentiment analysis), which aims to automatically determine the sentiment polarity (e.g., positive or negative) of user generated sentiment data (e.g., reviews, blogs, questions).

Machine learning algorithms have been proved promising and widely used for sentiment classification (Pang et al., 2002; Pang and Lee, 2008; Liu, 2012). However, the performance of these models relies on manually labeled training data. In many practical cases, we may have plentiful labeled data in the source domain, but very few or no labeled data in the target domain with a different data distribution. For example, we may have many labeled books reviews, but we are interested in detecting the polarity of electronics reviews. Reviews for different products might have different vocabularies, thus classifiers trained on one domain often fail to produce satisfactory results when transferring to another domain. This has motivated much research on cross-domain (domain adaptation) sentiment classification which transfers the knowledge from the source domain to the target domain (Thomas et al., 2006; Snyder and Barzilay, 2007; Blitzer et al., 2006; Blitzer et al., 2007; Daume III, 2007; Li and Zong, 2008; Li et al., 2009; Pan et al., 2010; Kumar et al., 2010; Glorot et al., 2011; Chen et al., 2011a; Chen et al., 2012; Li et al., 2012; Xia et al., 2013a; Li et al., 2013; Zhou et al., 2015a; Zhuang et al., 2015).

Depending on whether the labeled data are available for the target domain, cross-domain sen-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 322–332,
Berlin, Germany, August 7-12, 2016.

timent classification can be divided into two categories: supervised domain adaptation and unsupervised domain adaptation. In scenario of supervised domain adaptation, labeled data is available in the target domain but the number is usually too small to train a good sentiment classifier, while in unsupervised domain adaptation only unlabeled data is available in the target domain, which is more challenging. This work focuses on the unsupervised domain adaptation problem of which the essence is how to employ the unlabeled data of target domain to guide the model learning from the labeled source domain.

The fundamental challenge of cross-domain sentiment classification lies in that the source domain and the target domain have different data distribution. Recent work has investigated several techniques for alleviating the domain discrepancy: instance-weight adaptation (Huang et al., 2007; Jiang and Zhai, 2007; Li and Zong, 2008; Mansour et al., 2009; Dredze et al., 2010; Chen et al., 2011b; Chen et al., 2011a; Chen et al., 2012; Li et al., 2013; Xia et al., 2013a) and feature representation adaptation (Thomas et al., 2006; Snyder and Barzilay, 2007; Blitzer et al., 2006; Blitzer et al., 2007; Li et al., 2009; Pan et al., 2010; Zhou et al., 2015a; Zhuang et al., 2015). The first kind of methods assume that some training data in the source domain are very useful for the target domain and these data can be used to train models for the target domain after re-weighting. In contrast, feature representation approaches attempt to develop an adaptive feature representation that is effective in reducing the difference between domains.

Recently, some efforts have been initiated on learning robust feature representations with deep neural networks (DNNs) in the context of cross-domain sentiment classification (Glorot et al., 2011; Chen et al., 2012). Glorot et al. (2011) proposed to learn robust feature representations with stacked denoising auto-encoders (SDAs) (Vincent et al., 2008). Denoising auto-encoders are one-layer neural networks that are optimized to reconstruct input data from partial and random corruption. These denoisers can be stacked into deep learning architectures. The outputs of their intermediate layers are then used as input features for SVMs (Fan et al., 2008). Chen et al. (2012) proposed a marginalized SDA (mSDA) that addressed the two crucial limitations of SDAs: high

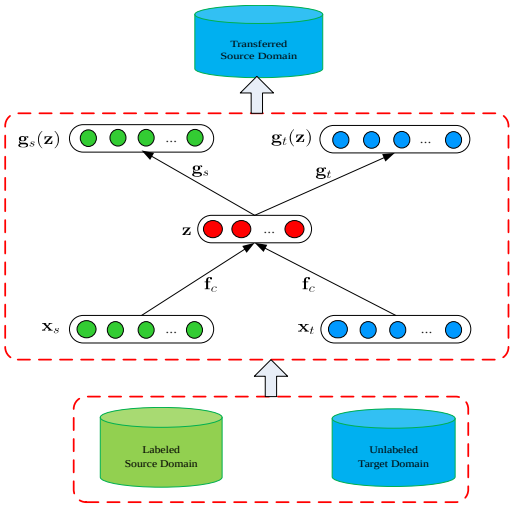


Figure 1: The framework of Bi-transferring Deep Neural Networks (BTDNNs). Through BTDNNs, a source domain example can be transferred to the target domain where it can be reconstructed by the target domain examples, and vice versa.

computational cost and lack of scalability to high-dimensional features. However, these methods learn the unified domain-invariable feature representations by combining the source domain data and that of the target domain data together, which cannot well characterize the domain-specific features as well as the commonality of domains.

To this end, we propose a Bi-Transferring Deep Neural Networks (BTDNNs) which can transfer the source domain examples to the target domain and also transfer the target domain examples to the source domain, as shown in Figure 1. In BTDNNs, the linear transformation makes the feasibility of transferring between domains, and the linear data reconstruction manner ensures the distribution consistency between the transferred domain and the desirable domain. Specifically, our BTDNNs has one common encoder $\mathbf{f}_c$, two decoders $\mathbf{g}_s$ and $\mathbf{g}_t$ which can map an example to the source domain and the target domain respectively. As a result, the source domain can be transferred to the target domain along with its sentiment label, and any supervised method can be used on the transferred source domain to train a classifier for sentiment classification in the target domain, as the transferred source domain data share the similar distribution as the target domain. Experimental results show that the proposed approach significantly outperforms several baselines, and achieves an accuracy which is competitive with the state-of-

the-art method for cross-domain sentiment classification.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes our proposed bi-transferring deep neural networks (BTDNNs). Section 4 presents the experimental results. In Section 5, we conclude with ideas for future research.

2 Related Work

Domain adaptation aims to generalize a classifier that is trained on a source domain, for which typically plenty of training data is available, to a target domain, for which data is scarce. Cross-domain generalization is important in many real applications, the key challenge is that data in the source and the target domain are often distributed differently.

Recent work has investigated several techniques for alleviating the difference in the context of cross-domain sentiment classification task. Blitzer et al. (2007) proposed a structural correspondence learning (SCL) algorithm to train a cross-domain sentiment classifier. SCL is motivated by a multi-task learning algorithm, alternating structural optimization (ASO), proposed by Ando and Zhang (2005). Given labeled data from a source domain and unlabeled data from both source and target domains, SCL attempts to model the relationship between "pivot features" and "non-pivot features". Pan et al. (2010) proposed a spectral feature alignment (SFA) algorithm to align the domain-specific words from the source and target domains into meaningful clusters, with the help of domain-independent words as a bridge. In the way, the cluster can be used to reduce the gap between domain-specific words of two domains. Dredze et al. (2010) combined classifier weights using confidence-weighted learning, which represented the covariance of the weight vectors. Xia et al. (2013a) proposed an instance selection and instance weighting method for cross-domain sentiment classification. After that, Xia et al. (2013b) proposed a feature ensemble plus sample selection method to further improve the sentiment classification adaptation. Zhou et al. (Zhou et al., 2015b) proposed to bridge the domain gap with the help of topical correspondence. Li et al. (2009) proposed to transfer common lexical knowledge across domains via matrix factorization techniques. Zhou et al. (2015a) further improved the matrix factorization techniques via a regularization term on the pivots and domain-specific words, ensuring that the pivots capture only correspondence aspects and the domain-specific words capture only individual aspects. Li and Zong (2008) proposed the multi-label consensus training approach which combined several base classifiers trained with SCL. Chen et al. (2012) proposed a domain adaptation algorithm based on sample and feature selection. Li et al. (2013) proposed an active learning algorithm for cross-domain sentiment classification. Xiao and Guo (2013) investigated the online active domain adaptation problem in a novel but practical setting where the labels can be acquired with a lower cost in the source domain than in the target domain.

There has also been research in exploring careful structuring of features or prior knowledge for domain adaptation. Daumé III (2007) proposed a kernel-mapping function which maps both source and target domains data to a high-dimensional feature space so that data points from the same domain are twice as similar as those form different domains. Dai et al. (2008) proposed translated learning which used a language model to link the class labels to the features in the source domain, which in turn is translated to the features in the target domain. Xia et al. (2010) proposed a POS-based ensemble model for cross-domain sentiment classification. Xiao et al. (2013) proposed a supervised representation learning method to tackle domain adaptation by inducing predictive latent features based on supervised word clustering. He et al. (2011) employed a joint sentiment-topic model for cross-domain sentiment classification; Bollegala et al. (2011) used a sentiment sensitive thesaurus to perform cross-domain sentiment classification. Xiao and Guo (2015) proposed to learn distributed state representations for cross-domain sequence predictions.

Recently, some efforts have been initiated on learning robust feature representations with deep neural networks (DNNs) for cross-domain natural language processing. Glorot et al. (2011) and Chen et al. (2012) proposed to use deep learning for cross-domain sentiment classification. Most recently, Yang and Eisenstein (2014) proposed an unsupervised domain adaptation method with marginalized structured dropout. Furthermore, Yang and Eisenstein (2015) proposed to use feature embeddings with metadata domain attributes for multi-domain adaptation. In this paper,

our proposed approach BTDNNs tackles the domain discrepancy with a linear data construction manner, which can effectively model the domain-specific features as well as the commonality of domains. Deep learning techniques have also been proposed to heterogeneous transfer learning (Socher et al., 2013; Zhou et al., 2014; Kan et al., 2015; Long et al., 2015), where knowledge is transferred from one modality to another based on the correspondences at hand. Our proposed framework can be considered as a more general case, where the bias of the correspondences between the source and target domains is constrained with a linear data reconstruction manner.

Besides, other researchers also explore the DNNs for sentiment analysis (Socher et al., 2011; Tang et al., 2014; Tang et al., 2015; Zhai and Zhang, 2016; Chandar et al., 2014). However, all these methods focus on the sentiment analysis without considering the domain discrepancy. In this paper, we focus on domain adaptation for sentiment classification with a different model formulation and task definition.

3 Bi-Transferring Deep Neural Networks

3.1 Problem Definition

Given two domains X_s and X_t, where X_s and X_t are referred to a source domain and a target domain, respectively. Suppose we have a set of labeled sentiment examples as well as some unlabeled examples in the source domain X_s with size n_s, containing terms from a vocabulary $\mathcal{V}$ with size m. The examples in the source domain X_s can be represented as a term-document matrix $\mathbf{X}_s = \{\mathbf{x}_1^s, \cdots, \mathbf{x}_{n_s}^s\} \in \mathbb{R}^{m \times n_s}$, with their sentiment labels $\mathbf{y}_s = \{y_1^s, \cdots, \mathbf{y}_{n_s}^s\}$, where $\mathbf{x}_i^s \in \mathbb{R}^m$ is the feature representation of the i-th source domain example with a tf-idf weight of the corresponding term and $y_i^s \in \{+1, -1\}$ is its sentiment label.[1]

Similarly, suppose we have a set of unlabeled examples in the target domain X_t with size n_t, containing terms from a vocabulary $\mathcal{V}$ with size m. The examples in target domain X_t can also be represented as a term-document matrix $\mathbf{X}_t = \{\mathbf{x}_1^{(t)}, \cdots, \mathbf{x}_{n_t}^{(t)}\} \in \mathbb{R}^{m \times n_t}$, where each example denotes a tf-idf weight of the corresponding term. The task of cross-domain sentiment classification is to learn a robust classifier to predict the polarity of unseen examples from X_t. Note that we only consider one source domain and one target domain in this paper. However, our proposed algorithm is a general framework and can be easily adapted to multi-domain problems.

[1]We use upper case and lower case characters represent the matrices and vectors respectively throughout the paper.

3.2 Basic Auto-Encoder

An auto-encoder is an unsupervised neural network which is trained to reconstruct a given input vector from its latent representation (Bengio et al., 2007). It can be seen as a special neural network with three layers: the input layer, the latent layer, and the reconstruction layer. An auto-encoder contains two parts: encoder and decoder. The encoder, denoted as $\mathbf{f}$, attempts to map an input vector $\mathbf{x} \in \mathbb{R}^{m \times 1}$ to the latent representation $\mathbf{z} \in \mathbb{R}^{k \times 1}$, in which k is the number of neurons in the latent layer. Usually, $\mathbf{f}$ is a nonlinear function as follows:

$$\mathbf{z} = \mathbf{f}(\mathbf{x}) = s_e(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{1}$$

where s_e is the activation function of the encoder, whose input is called the activation function, which is usually non-linear, such as sigmoid function or tanh function is a linear transform parameter, and $\mathbf{b} \in \mathbb{R}^{k \times 1}$ is the basis.

The decoder, denoted as $\mathbf{g}$, tries to map the latent representation $\mathbf{z}$ back to a reconstruction:

$$\mathbf{g}(z) = s_d(\mathbf{W}'\mathbf{z} + \mathbf{b}') \tag{2}$$

Similarly, s_d is the activation function of the decoder with parameters $\{\mathbf{W}', \mathbf{b}'\}$.

The training objective is the determination of parameters $\{\mathbf{W}, \mathbf{b}\}$ and $\{\mathbf{W}', \mathbf{b}'\}$ that minimize the average reconstruction errors:

$$\mathcal{L} = \min_{\mathbf{W},\mathbf{b},\mathbf{W}',\mathbf{b}'} \sum_{i=1}^{N} \left\| \mathbf{x}_i - \mathbf{g}(\mathbf{f}(\mathbf{x}_i)) \right\|_2^2 \tag{3}$$

where $\mathbf{x}_i$ represents the i-th one of N training examples. Parameters $\{\mathbf{W}, \mathbf{b}\}$ and $\{\mathbf{W}', \mathbf{b}'\}$ can be optimized by stochastic or mini-batch gradient descent. By minimizing the reconstruction error, we require the latent features should be able to reconstruct the original input as much as possible.

3.3 Bi-Transferring Deep Neural Networks

The traditional auto-encoder in subsection 3.2 attempts to reconstruct the input itself, which is usually used for feature representation learning. Nevertheless, our proposed bi-transferring deep neural networks (BTDNNs) attempts to transfer

examples between domains to deal with the domain discrepancy, with the inspiration of DNNs in computer vision (Kan et al., 2015). Motivated by the successful application in computer vision (Kan et al., 2015), we construct the architecture of BTDNNs with one encoder $\mathbf{f}_e$, and two decoders, $\mathbf{g}_s$ and $\mathbf{g}_t$ shown in Figure 1, which can transform an input example to the source domain and the target domain respectively.[2]

Specifically, the encoder $\mathbf{f}_c$ tries to map an input example $\mathbf{x}$ into the latent feature representation $\mathbf{z}$, which is common to both the source and target domains as follows:

$$\mathbf{z} = \mathbf{f}_c(\mathbf{x}) = s_e(\mathbf{W}_c\mathbf{x} + \mathbf{b}_c) \tag{4}$$

The decoder $\mathbf{g}_s$ attempts to map the latent representation to the source domain, and the decoder $\mathbf{g}_t$ attempts to map the latent representation to the target domain as follows:

$$\mathbf{g}_s(\mathbf{x}) = s_d(\mathbf{W}_s\mathbf{z} + \mathbf{b}_s) \tag{5}$$

$$\mathbf{g}_t(\mathbf{x}) = s_d(\mathbf{W}_t\mathbf{z} + \mathbf{b}_t) \tag{6}$$

where $s_e(\cdot)$ and $s_d(\cdot)$ are the element-wise nonlinear activation function, e.g., sigmoid or tanh function, $\mathbf{W}_c$ and $\mathbf{b}_c$ are the parameters for encoder $\mathbf{f}_c$, $\mathbf{W}_s$ and $\mathbf{b}_s$ are the parameters for decoder $\mathbf{g}_s$, $\mathbf{W}_t$ and $\mathbf{b}_t$ are the parameters for decoder $\mathbf{g}_t$.

Following the literature (Kan et al., 2015), we attempt to map the source domain examples $\mathbf{X}_s$ to the source domain (e.g., $\mathbf{X}_s$ itself) with an encoder $\mathbf{f}_c$ and a decoder $\mathbf{g}_s$. Similarly, given an encoder $\mathbf{f}_c$ and a decoder $\mathbf{g}_t$, we aim to map the source domain examples $\mathbf{X}_s$ to the target domain. Although it is unknown what the mapped examples look like, they are expected to follow the similar distribution as the target domain. This kind of distribution consistency between two domains can be characterized from the perspective of a linear data reconstruction manner.

The two domains $\mathbf{X}_s$ and $\mathbf{X}_t$ can be generally reconstructed from each other, and their distances can be used to measure the domain discrepancy. Following the literature (He et al., 2012), BTDNNs attempt to represent a transferred source domain $\mathbf{g}_t(\mathbf{f}_c(\mathbf{x}_i^s))$ with a linear reconstruction function from the target domain:

$$\|\mathbf{g}_t(\mathbf{f}_c(\mathbf{x}_i^s)) - \mathbf{X}_t\beta_i^t)\|_2^2 \tag{7}$$

[2]In the implementation, we use the stacked denoising auto-encoders (SDA) (Vincent et al., 2008) to model the source and the target domain data.

where β_i^t is the coefficients for the reconstruction of transferred source domain examples. Equation (7) enforces that each example of transferred domain is consistent with that of target domain, which ensures that the transferred source domain follows the similar distribution as the target domain. The overall objective for the examples of source domain $\mathbf{X}_s$ can be formulated as below:

$$\min_{\mathbf{f}_c,\mathbf{g}_s,\mathbf{g}_t,\beta_i^s} \|\mathbf{X}_s - \mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_s))\|_2^2 + \|\mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_s)) - \mathbf{X}_t\mathbf{B}_t)\|_2^2$$
$$s.t.\ \|\beta_i^t\|_2^2 < \tau,\ \ \mathbf{B}_t = [\beta_1^t, \beta_2^t, \cdots, \beta_{n_s}^t]^T \in \mathbb{R}^{n_s \times n_t}$$

where $\mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_s) = [\mathbf{g}_s(\mathbf{f}_c(\mathbf{x}_1^s)), \cdots, \mathbf{g}_s(\mathbf{f}_c(\mathbf{x}_{n_t}^s))]$ and $\mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_s) = [\mathbf{g}_t(\mathbf{f}_c(\mathbf{x}_1^t)), \mathbf{g}_t(\mathbf{f}_c(\mathbf{x}_{n_s}^t))]$. The same simplifications are used hereinafter if without misunderstanding.

Similarly, for the examples of target domain $\mathbf{X}_t$, with encoder $\mathbf{f}_c$ and decoder $\mathbf{g}_t$ they should be mapped on the target domain. Also, with encoder $\mathbf{f}_c$ and decoder $\mathbf{g}_s$ they should be mapped to the source domain, where they can be reconstructed by the source domain examples from the point of view of a linear data reconstruction manner (He et al., 2012), so as to ensure a similar distribution between the source domain and the transferred target domain. The overall objective for the examples of target domain $\mathbf{X}_t$ can be written as:

$$\min_{\mathbf{f}_c,\mathbf{g}_s,\mathbf{g}_t,\beta_i^t} \|\mathbf{X}_t - \mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_t))\|_2^2 + \|\mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_t)) - \mathbf{X}_s\mathbf{B}_s)\|_2^2$$
$$s.t.\ \|\beta_j^s\|_2^2 < \tau,\ \ \mathbf{B}_s = [\beta_1^s, \beta_2^s, \cdots, \beta_{n_t}^s]^T \in \mathbb{R}^{n_t \times n_s}$$

Combining the above equations, the overall objective of BTDNNs can be formulated as follows:

$$\begin{aligned}\min_{\mathbf{f}_c,\mathbf{g}_s,\mathbf{g}_t,\mathbf{B}_s,\mathbf{B}_t} & \|\mathbf{X}_s - \mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_s))\|_2^2 + \|\mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_s)) - \mathbf{X}_t\mathbf{B}_t)\|_2^2 \\ & + \|\mathbf{X}_t - \mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_t))\|_2^2 + \|\mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_t)) - \mathbf{X}_s\mathbf{B}_s)\|_2^2 \\ & + \gamma\Big(\sum_{i=1}^{n_s}\|\beta_i^t\|_2^2 + \sum_{j=1}^{n_t}\|\beta_j^s\|_2^2\Big)\end{aligned} \tag{8}$$

where γ is a regularization parameter controlling the amount of shrinkage. With the optimization of equation (8), our proposed approach BTDNNs can map any input examples to the source and target domains respectively. Especially, the source domain examples $\mathbf{X}_s$ can transferred to the target domain along with their sentiment labels. The transferred source domain data $\mathbf{g}_t(\mathbf{f}_s(\mathbf{X}_s))$ share the similar distribution as the target domain, so any supervised method can be used to learn a classifier for sentiment classification in the target domain. In this paper, a linear support vector machine (SVM) (Fan et al., 2008) is employed for building sentiment classification models.

3.4 Learning Algorithm

Note that the optimization problem in equation (8) is not convex in variables $\{\mathbf{f}_c, \mathbf{g}_s, \mathbf{g}_t, \mathbf{B}_s, \mathbf{B}_t\}$ together. However, when considering one variable at a time, the cost function turns out to be convex. For example, given $\{\mathbf{g}_s, \mathbf{g}_t, \mathbf{B}_s, \mathbf{B}_t\}$, the cost function is a convex function w.r.t. $\mathbf{f}_c$. Therefore, although we cannot expect to get a global minimum of the above problem, we shall develop a simple and efficient optimization algorithm via alternative iterations.

3.4.1 Optimize $\{\mathbf{f}_c, \mathbf{g}_s, \mathbf{g}_t\}$ given $\{\mathbf{B}_s, \mathbf{B}_t\}$

When $\mathbf{B}_s$ and $\mathbf{B}_t$ are fixed, the objective function in equation (8) can be formulated as:

$$\min_{\mathbf{f}_c,\mathbf{g}_s,\mathbf{g}_t} \|\mathbf{X}_s - \mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_s))\|_2^2 + \|\mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_s)) - \bar{\mathbf{X}}_t)\|_2^2 + \|\mathbf{X}_t - \mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_t))\|_2^2 + \|\mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_t)) - \bar{\mathbf{X}}_s)\|_2^2 \quad (9)$$

where $\bar{\mathbf{X}}_s = \mathbf{X}_s\mathbf{B}_s$ and $\bar{\mathbf{X}}_t = \mathbf{X}_t\mathbf{B}_t$. Equation (9) can easily optimized by gradient descent as the basic auto-encoder (Bengio et al., 2007).

3.4.2 Optimize $\{\mathbf{B}_s, \mathbf{B}_t\}$ given $\{\mathbf{f}_c, \mathbf{g}_s, \mathbf{g}_t\}$

When $\{\mathbf{f}_c, \mathbf{g}_s, \mathbf{g}_t\}$ are fixed, the objective function in equation (8) can be written as:

$$\min_{\mathbf{B}_s,\mathbf{B}_t} \|\mathbf{G}_t - \mathbf{X}_t\mathbf{B}_t)\|_2^2 + \|\mathbf{G}_s - \mathbf{X}_s\mathbf{B}_s)\|_2^2 + \gamma\Big(\sum_{i=1}^{n_s}\|\beta_i^t\|_2^2 + \sum_{j=1}^{n_t}\|\beta_j^s\|_2^2\Big)$$

where $\mathbf{g}_s(\mathbf{f}_c(\mathbf{X}_t)) = \mathbf{G}_s = [\mathbf{g}_1^s, \cdots, \mathbf{g}_{n_t}^s]$ and $\mathbf{g}_t(\mathbf{f}_c(\mathbf{X}_s)) = \mathbf{G}_t = [\mathbf{g}_1^t, \cdots, \mathbf{g}_{n_s}^t]$. Since $\mathbf{G}_s$ and $\mathbf{G}_t$ are independent with each other, so they can be optimized independently. The optimization of $\mathbf{G}_s$ with other variables fixed is a least squares problem with ℓ_2-regularization. It can also be decomposed into n_t optimization problems, with each corresponding to one β_j^s and can be solved in parallel:

$$\min_{\beta_j^s} \|\mathbf{g}_j^s - \mathbf{X}_s\beta_j^s\|_2^2 + \gamma\|\beta_j^s\|_2^2 \quad (10)$$

for $j = 1, 2, \cdots, n_t$. It is a standard ℓ_2-regularized least squares problem and the solution is:

$$\beta_j^s = \left(\mathbf{X}_s^T\mathbf{X}_s + \gamma\mathbf{I}\right)^{-1}\mathbf{X}_s^T\mathbf{g}_j^s \quad (11)$$

where $\mathbf{I}$ is an identity matrix with all entries equal to 1.

Similarly, The optimization of $\mathbf{G}_t$ can also be decomposed into n_s ℓ_2-regularized least squares problems and the solution of each one is:

$$\beta_i^t = \left(\mathbf{X}_t^T\mathbf{X}_t + \gamma\mathbf{I}\right)^{-1}\mathbf{X}_t^T\mathbf{g}_i^t \quad (12)$$

for $i = 1, 2, \cdots, n_s$. We repeat the above equations until $\mathbf{f}_c$, $\mathbf{g}_s$, $\mathbf{g}_t$, $\mathbf{B}_s$ and $\mathbf{B}_t$ converge or a maximum number of iterations is exceeded.

3.5 Algorithm Complexity

In this section, we analyze the computational complexity of the learning algorithm described in equations (9), (11) and (12). Besides expressing the complexity of the algorithm using big O notation, we also count the number of arithmetic operations to provide more details about the run time. Computational complexity of learning matrix $\mathbf{G}_s$ is $O(m \times n_s \times k)$ per iteration. Similarly, for each iteration, learning matrices $\mathbf{G}_t$ takes $O(m \times n_t \times k)$. Learning matrices $\mathbf{B}_s$ and $\mathbf{B}_t$ takes $O(m^2 \times n_s)$ and $O(m^2 \times n_t)$ operations per iteration. In real applications, we have $k \ll m$. Therefore, the overall complexity of the algorithm, dominated by computation of matrices $\mathbf{B}_s$ and $\mathbf{B}_t$, is $O(m^2 \times n)$ where $n = \max(n_s, n_t)$.

4 Experiments

4.1 Data Set

Domain adaptation for sentiment classification has been widely studied in the NLP community. A large majority experiments are performed on the benchmark made of reviews of Amazon products gathered by Blitzer et al. (2006). This data set contains 4 different domains: Book (B), DVDs (D), Electronics (E) and Kitchen (K). For simplicity and comparability, we follow the convention of (Blitzer et al., 2006; Pan et al., 2010; Glorot et al., 2011; Xiao and Guo, 2013) and only consider the binary classification problem whether a review is positive (higher than 3 stars) or negative (3 stars or lower). There are 1000 positive and 1000 negative reviews for each domain, as well as approximately 4,000 unlabeled reviews (varying slightly between domains). The positive and negative reviews are also exactly balanced.

Following the literature (Pan et al., 2010), we can construct 12 cross-domain sentiment classification tasks: D → B, E → B, K → B, K → E, D → E, B → E, B → D, K → D, E → D, B → K, D → K, E → K, where the word before an arrow corresponds with the source domain and the word after an arrow corresponds with the target domain. To be fair to other algorithms that we compare to, we use the raw bag-of-words unigram/bigram features as their input and pre-process with tf-idf (Blitzer et al., 2006). Table 1 presents the statistics of the data set.

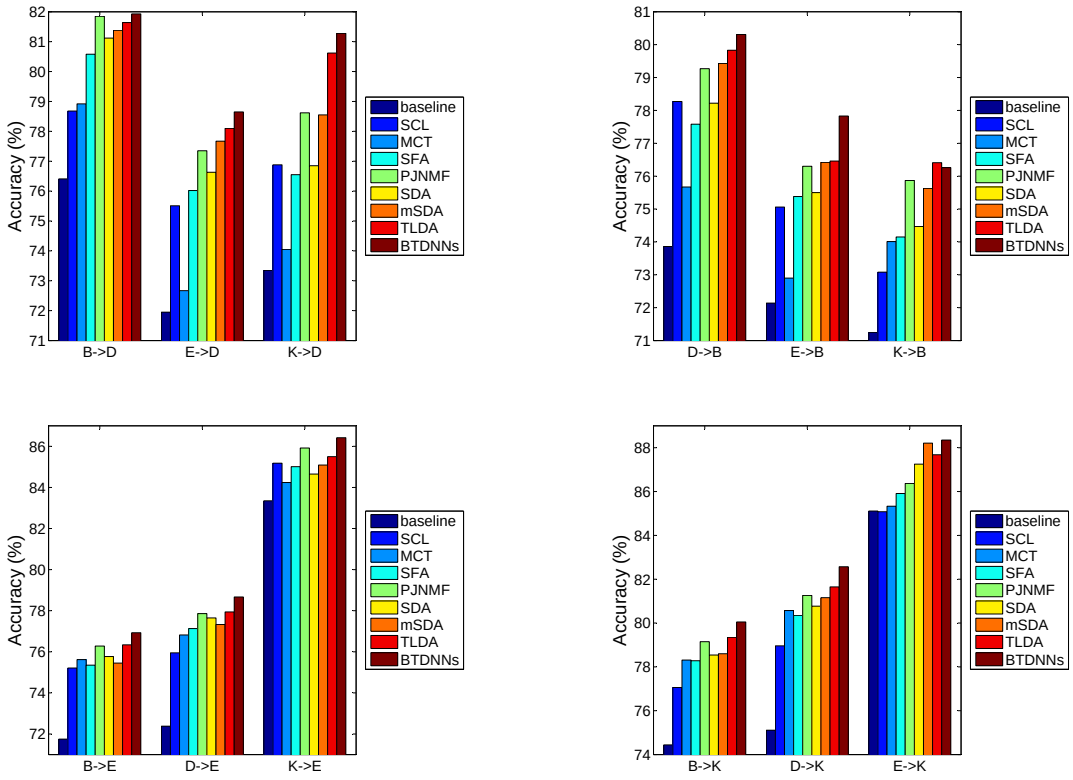


Figure 2: Average results for cross-domain sentiment classification on the Amazon product benchmark of 4 domains.

Domain	#Train	#Test	#Unlab.	% Neg.
Books	1600	400	4465	50%
DVDs	1600	400	5945	50%
Electronics	1600	400	5681	50%
Kitchen	1600	400	3586	50%

Table 1: Amazon review statistics. This table depicts the number of training, testing and unlabeled reviews for each domain, as well as the portion of negative training reviews of the data set.

4.2 Compared Methods

As a *baseline* method, we train a linear SVM (Fan et al., 2008) on the raw bag-of-words representation of the labeled *source* domain and test it on the *target* domain. In the original paper regarding the benchmark data set, Blitzer et al. (2006) adapted Structural Correspondence Learning (SCL) for sentiment analysis. Li and Zong (2008) proposed the Multi-label Consensus Training (MCT) approach which combined several base classifiers trained with SCL. Pan et al. (2010) first used a Spectral Feature Alignment (SFA) algorithm to align words from the source and target domains to help bridge the gap between them. Zhou et al. (2015a) proposed a method called PJNMF, which linked heterogeneous input features with pivots via joint non-negative matrix factorization.

Recently, some efforts have been initiated on learning robust feature representations with DNNs for cross-domain sentiment classification. Glorot et al. (2011) first employed stacked Denoising Auto-encoders (SDA) to extract meaningful representation for domain adaptation. Chen et al. (2012) proposed marginalized SDA (mSDA) that addressed the high computational cost and lack of scalability to high-dimensional features. Zhuang et al. (2015) proposed a state-of-the-art method called transfer learning with deep auto-encoders (TLDA).

For SCL, PJNMF, SDA, mSDA and TLDA, we use the source codes provided by the authors. For SFA and MCT, we re-implement them based on the original papers. The above methods serve as comparisons in our empirical evaluation. For fair comparison, all hyper-parameters are set by 5-fold cross validation on the training set from the source domain.[3] For our proposed BTDNNs, the number

[3]We keep the default value of some of the parameters in SCL and SFA, e.g., the number of stop-words removed and stemming parameters – as they were already tuned for this

of hidden neurons is set as 1000, the regularization parameter γ is tuned via 5-fold cross-validation.

For SDA, mSDA, TLDA and BTDNNs, we can construct the classifiers for the target domain in two ways. The first way is directly to use the stacking SVM on top of the output of the hidden layer. The second way is to apply the standard SVM to train a classifier for source domain in the embedding space. Then the classifiers is applied to predict sentiment labels for target domain data. For fair comparison with the shallow models, we choose the second way in this paper.

Figure 2 shows the accuracy of classification results for all methods and for all source-target domain pairs. We can check that all compared methods achieve the similar performance with the results reported in the original papers. From Figure 2, we can see that our proposed approach BTDNNs outperforms all other eight comparison methods in general. The `baseline` performs poorly on all the 12 tasks, while the other seven domain adaptation methods, SCL, MCT, SFA, PJNMF, SDA, mSDA and TLDA, consistently outperform the `baseline` method across all the 12 tasks, which demonstrates that the transferred knowledge from the source domain to the target domain is useful for sentiment classification. Nevertheless, the improvements achieved by these seven methods over the `baseline` are much smaller than the proposed approach BTDNNs.

Surprisingly, we note that the deep learning based methods (SDA, mSDA and TLDA) perform worse than our approach, the reason may be that SDA, mSDA and TLDA learn the unified domain-invariable feature representations by combining the source domain data and that of the target domain data together, which cannot well characterize the domain-specific features as well as the commonality of domains. On the contrary, our proposed BTDNNs ensures the feasibility of transferring between domains, and the distribution consistency between the transferred domain and the desirable domain is constrained with a linear data reconstruction manner.

We also conduct significance tests for our proposed approach BTDNNs and the state-of-the-art method (TLDA) using a McNemar paired test for labeling disagreements (Gillick and Cox, 1989). In general, the average result on the 12 source-target domain pairs indicates that the difference between BTDNNs and TLDA is mildly significant with $p < 0.08$. Furthermore, we also conduct the experiments on a much larger industrial-strength data set of 22 domains (Glorot et al., 2011). The preliminary results show that BTDNNs significantly outperforms TLDA ($p < 0.05$). Therefore, we will report our detailed results and discussions in our future work.

benchmark set by the authors.

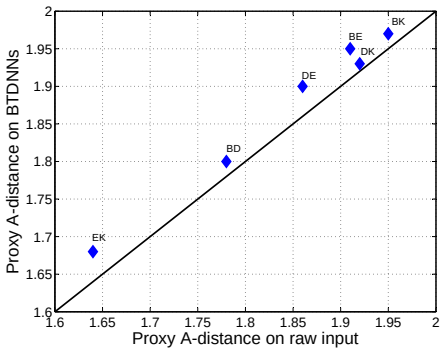


Figure 3: Proxy A-distance between domains of the Amazon benchmark for the 6 different pairs.

4.3 Domain Divergence

In this subsection, we look into how similar two domains are to each other. Ben-David et al. (2006) showed that the A-distance as a measure of how different between the two domains. They hypothesized that it should be difficult to discriminate between the source and target domains in order to have a good transfer between them. In practice, computing the exact A-distance is impossible and one has to compute a proxy. Similar to (Glorot et al., 2011), the proxy for the A-distance is then defined as $2(1 - 2\epsilon)$, where ϵ is the generalization error of a linear SVM classifier trained on the binary classification problem to distinguish inputs between the two domains.

Figure 3 presents the results for each pair of domains. Surprisingly, the distance is increased with the help of new feature representations, e.g., distinguishing between domains becomes easier with the BTDNNs features. We explain this effect through the fact that BTDNNs can ensure the feasibility of transferring between domains, and the distribution consistency between the transferred domain and the desirable domain is constrained with a linear data reconstruction manner, which can learn a generally better representations for the input data. This helps both tasks, distinguishing between domains and sentiment classification

(e.g., in the book domain BTDNNs might interpolate the feature "exciting" from "boring", both are not particularly relevant for sentiment classification but might help distinguish the review from the Electronic domain.).

5 Conclusions and Future Work

In this paper, we propose a novel Bi-Transferring Deep Neural Networks (BTDNNs) for cross-domain sentiment classification. The proposed BTDNNs attempts to transfer the source domain examples to the target domain, and also transfer the target domain examples to the source domain. The linear transformation of BTDNNs ensures the feasibility of transferring between domains, and the distribution consistency between the transferred domain and the desirable domain is constrained with a linear data reconstruction manner. Experimental results show that BTDNNs significantly outperforms the several baselines, and achieves an accuracy which is competitive with the state-of-the-art method for sentiment classification adaptation.

There are some ways in which this research could be continued. First, since deep learning may obtain better generalization on large-scale data sets (Bengio, 2009), a straightforward path of the future research is to apply the proposed BTDNNs for domain adaptation on a much larger industrial-strength data set of 22 domains (Glorot et al., 2011). Second, we will try to investigate the use of the proposed approach for other kinds of data set, such as 20 newsgroups and Reuters-21578 (Li et al., 2012; Zhuang et al., 2013).

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61303180 and No. 61573163), the Fundamental Research Funds for the Central Universities (No. CCNU15ZD003 and No. CCNU16A02024), and also supported by a Discovery grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada and an NSERC CREATE award. We thank the anonymous reviewers for their insightful comments.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853.

Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. 2006. Analysis of representations for domain adaptation. In *NIPS*, pages 137–144.

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle, Universite De Montreal, and Montreal Quebec. 2007. Greedy layer-wise training of deep networks. In *NIPS*, pages 153–160.

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

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*, pages 120–128.

John Blitzer, M. Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: domain adaptation for sentiment classification. In *EMNLP*, pages 120–128.

Danushka Bollegala, David Weir, and John Carroll. 2011. Using multiple sources to construct a sentiment sensitive thesaurus for cross-domain sentiment classification. In *ACL*, pages 132–141.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *NIPS*, pages 1–9.

Minmin Chen, John Blitzer, and Kilian Weinberger. 2011a. Co-training for domain adaptation. In *NIPS*, pages 1–9.

Minmin Chen, Kilian Weinberger, and Yixin Chen. 2011b. Automatic feature decomposition for single view co-training. In *ICML*, pages 953–960.

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. In *ICML*, pages 767–774.

W. Dai, Y. Chen, G. Xue, Q. Yang, and Y. Yu. 2008. Translated learning: transfer learning across different feature spaces. In *NIPS*, pages 353–360.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *ACL*, pages 256–263.

Mark Dredze, Alex Kulesza, and Koby Crammer. 2010. Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*, 79(12):123–149.

R. Fan, Chang K., Hsieh C., Wang X., and Lin C. 2008. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874.

L. Gillick and S. Cox. 1989. Some statistical issues in the comparison of speech recoginition algorithms. In *ICASSP*, pages 532–535.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, pages 513–520.

Yulan He, Chenghua Lin, and Harith Alani. 2011. Automatically extracting polarity-bearing topics for cross-domain sentiment classification. In *ACL*, pages 123–131.

Zhanying He, Chun Chen, Bu, Jiajun, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document summarization based on data reconstruction. In *AAAI*, pages 620–626.

J. Huang, A. Smola, A. Gretton, K. Bordwardt, and B. Scholkopf. 2007. Correcting samples selection bias by unlabeled data. In *NIPS*, pages 601–608.

J. Jiang and C. Zhai. 2007. Instance weighting for domain adaptation in nlp. In *ACL*, pages 264–271.

Meina Kan, Shiguang Shan, and Xilin Chen. 2015. Bi-shifting auto-encoder for unsupervised domain adaptation. In *ICCV*, pages 3846–3854.

Abhishek Kumar, Avishek Saha, and Hal Daumé III. 2010. A co-regularization based semi-supervised domain adaptation. In *NIPS*, pages 478–486.

Shoushan Li and Chengqing Zong. 2008. Multi-domain adaption for sentiment classification: Using multiple classifier combining classification. In *NLPKE*.

Tao Li, Vikas Sindhwani, Chris H. Q. Ding, and Yi Zhang 0005. 2009. Knowledge transformation for cross-domain sentiment classification. In *SIGIR*, pages 716–717.

Lianghao Li, Xiaoming Jin, and Mingsheng Long. 2012. Topic correlation analysis for cross-domain text classification. In *AAAI*, pages 998–1004.

Shoushan Li, Yunxia Xue, Zhongqing Wang, and Guodong Zhou. 2013. Active learning for cross-domain sentiment classification. In *IJCAI*, pages 2127–2133.

B. Liu. 2012. Sentiment analysis and opinion mining. *Morgan & Claypool Publishers*.

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. 2015. Learning transferable features with deep adaptation networks. In *ICML*, pages 97–105.

T. Mansour, M. Mohri, and A. Rostamizadeh. 2009. Domain adatation with multiple sources. In *NIPS*, pages 264–271.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *WWW*, pages 751–760.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(12):1–135.

B. Pang, L. Lee, and S. Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *EMNLP*, pages 79–86.

Benjamin Snyder and Regina Barzilay. 2007. Multiple aspect ranking using the good grief algorithm. In *NAACL*, pages 300–307.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP*, pages 151–161.

Richard Socher, Milind Ganjoo, Christopher D. Manning, and Andrew Y. Ng. 2013. Zero-shot learning through cross-modal transfer. In *NIPS*, pages 935–943.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *ACL*, pages 1555–1565.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *EMNLP*, pages 1422–1432.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *EMNLP*, pages 327–335.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103.

Rui Xia, , and Chengqing Zong. 2010. A pos-based ensemble model for cross-domain sentiment classification. In *IJCNLP*, pages 614–622.

Rui Xia, Xuelei Hu, Jianfeng Lu, and Chengqing Zong. 2013a. Instance selection and instance weighting for cross-domain sentiment classification via pu learning. In *IJCAI*, pages 2276–2182.

Rui Xia, Chengqing Zong, Xuelei Hu, and Cambria Erik. 2013b. Feature ensemble plus sample selection: Domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28(3):10–18.

Min Xiao and Yuhong Guo. 2013. Online active learning for cost-sensitive domain adaptation. In *CoNLL*, pages 1–9.

Min Xiao and Yuhong Guo. 2015. Learning hidden markov models with distributed state representations for domain adaptation. In *ACL*, pages 524–529.

Min Xiao, Feipeng Zhao, and Yuhong Guo. 2013. Learning latent word representations for domain adaptation using supervised word clustering. In *EMNLP*, pages 152–162.

Yi Yang and Jacob Eisenstein. 2014. Fast easy unsupervised domain adaptation with marginalized structured dropout. In *ACL*, pages 538–544.

Yi Yang and Jacob Eisenstein. 2015. Unsupervised multi-domain adaptation with feature embeddings. In *NAACL*, pages 672–682.

Shuangfei Zhai and Zhongfei (Mark) Zhang. 2016. Semi-supervised autoencoder for sentiment analysis. In *AAAI*, pages 1394–1400.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. In *ACL*, pages 653–662.

Joey Tianyi Zhou, Sinno Jialin Pan, IvorW. Tsang, and Yan Yan. 2014. Hybrid heterogeneous transfer learning through deep learning. In *AAAI*, pages 2213–2219.

Guangyou Zhou, Tingting He, Wensheng Wu, and Xiaohua Hu. 2015a. Linking heterogeneous input features with pivots for domain adaptation. In *IJCAI*, pages 1419–1425.

Guangyou Zhou, Yin Zhou, Xiyue Guo, Xinhui Tu, and Tingting He. 2015b. Cross-domain sentiment classification via topical correspondence transfer. *Neurocomputing*, 159:298–305.

Fuzhen Zhuang, Ping Luo, Peifeng Yin, Qing He, and Zhongzhi Shi. 2013. Concept learning for cross-domain text classification: A general probabilistic framework. In *IJCAI*, pages 1960–1966.

Fuzhen Zhuang, Xiaohu Cheng, Ping Luo, Sinno Jialin Pan, and Qing He. 2015. Supervised representation learning: Transfer learning with deep autoencoders. In *IJCAI*, pages 4119–4125.

Document-level Sentiment Inference with Social, Faction, and Discourse Context

Eunsol Choi **Hannah Rashkin** **Luke Zettlemoyer** **Yejin Choi**
Computer Science & Engineering
University of Washington
{eunsol,hrashkin,lsz,yejin}@cs.washington.edu

Abstract

We present a new approach for document-level sentiment inference, where the goal is to predict directed opinions (*who* feels positively or negatively towards *whom*) for all entities mentioned in a text. To encourage more complete and consistent predictions, we introduce an ILP that jointly models (1) sentence- and discourse-level sentiment cues, (2) factual evidence about entity factions, and (3) global constraints based on social science theories such as homophily, social balance, and reciprocity. Together, these cues allow for rich inference across groups of entities, including for example that CEOs and the companies they lead are likely to have similar sentiment towards others. We evaluate performance on new, densely labeled data that provides supervision for all pairs, complementing previous work that only labeled pairs mentioned in the same sentence. Experiments demonstrate that the global model outperforms sentence-level baselines, by providing more coherent predictions across sets of related entities.

1 Introduction

Documents often present a complex web of facts and opinions that hold among the entities they describe. Consider the international relations story in Figure 1. Representatives from three countries form factions and create a network of sentiment. While some opinions are relatively directly stated (e.g., Russia criticizes Belarus), many others must be inferred based on the factual ties among entities (e.g., Moscow, Gryzlov, and Russia probably share the same sentiment towards other entities) and known social context (e.g., Russia probably dislikes Saakhashvili since Russia criticized Belarus for supporting him). In this paper, we show that jointly reasoning about all of these factors can provide more complete and consistent document-level sentiment predictions.

Russia criticized *Belarus* for permitting *Georgian* President *Mikheil Saakhashvili* to appear on *Belorussian* television. "The appearance was an unfriendly step towards *Russia*," the speaker of *Russian* parliament *Boris Gryzlov* said. . . . *Saakhashvili* announced Thursday that *he* did not understand *Russia*'s claims. *Moscow* refused to have any business with *Georgia*'s president after the armed conflict in 2008 . . .

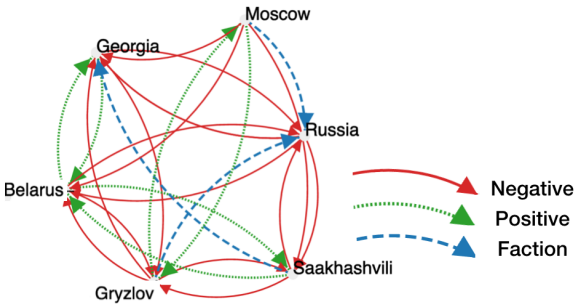


Figure 1: Example text excerpt paired with the document-level sentiment graph we aim to recover. The graph includes edges with direct textual support (e.g., from Russian to Belarus given the verb "criticized") as well as ones that must be inferred at the whole-document level (e.g., from Gryzlov to Saakhashvili given the web of relationships and opinions between them, Georgia, Russian, and Belarus).

More concretely, we present a global model for document-level entity-to-entity sentiment, i.e., *who* feels positively (or negatively) towards *whom*. Our goal is to make exhaustive predictions over all entity pairs, including those that require cross-sentence inference. We present a Integer Linear Programming (ILP) model that combines three complementary types of evidence: entity-pair sentiment classification, template-based faction extraction, and sentiment dynamics in social groups. Together, they allow for recovering more complete predictions of both the explicitly stated and im-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 333–343,
Berlin, Germany, August 7-12, 2016.

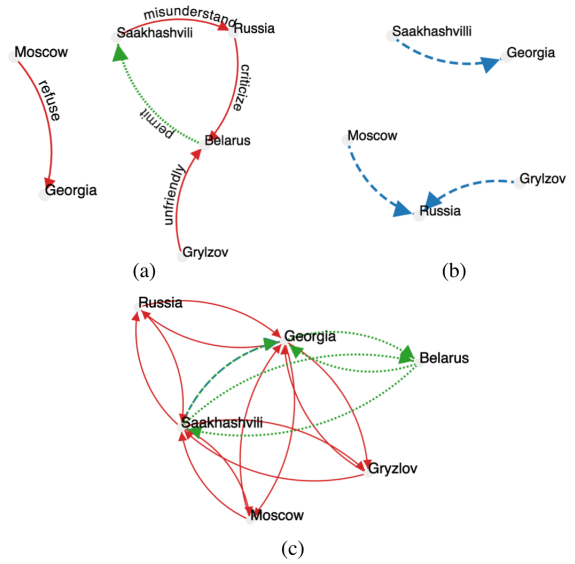


Figure 2: Entity subgraphs for the example in Figure 1: (a) shows explicitly stated sentiment, (b) shows faction relationships and (c) shows all edges for Georgia and its representative Saakhashvili. Through Saakhasvili's relationship with Belarus, Georgia forms an alliance with Belarus, providing evidence for an inferred negative stance towards Russia. Green dotted edges represent positive sentiment, red are negative, and blue dashed lines show faction relationship.

plicit sentiment, while preserving consistency.

The sentiment dynamics in social groups, motivated by social science theories, are encoded as soft ILP constraints. They include a notion of homophily, that entities in the same group tend to have similar opinions (Lazarsfeld and Merton, 1954). For example, Figure 2b shows directed faction edges, where one entity is likely to agree with the other's opinions. They also encode dyadic social constraints (i.e., the likely reciprocity of opinions (Gouldner, 1960)) and triadic social dynamics following social balance theory (Heider, 1946). For example, from Russia's criticism on Belarus and Belarus' positive attitude towards Saakhashvilli (in Figure 2a), we can infer that Russia is negative towards Saakhashvilli (in Figure 2c). When considered in aggregate, these constraints can greatly improve the consistency over the overall document-level predictions.

Our work stands in contrast to previous approaches in three aspects. First, we apply social dynamics motivated by social science theories to entity-entity sentiment analysis in unstructured text. In contrast, most previous studies focused on social media or dialogue data with overt social network structure when integrating social dynamics (Tan et al., 2011; Hu et al., 2013; West et al., 2014). Second, we aim to recover sentiment that can be inferred through partial evidence that spans multiple sentences. This complements prior efforts for accessing implied sentiment where the key evidence is, by and large, at the sentence level (Zhang and Liu, 2011; Yang and Cardie, 2013; Deng and Wiebe, 2015a). Finally, we present the first approach to model the relationship between factual and subjective relations.

We evaluate the approach on a newly gathered corpus with dense document-level sentiment labels in news articles.[1] This data includes comprehensively annotated sentiment between all entity pairs, including those that do not appear together in any single sentence. Experiments demonstrate that the global model significantly improves performance over a pairwise classifier and other strong baselines. We also perform a detailed ablation and error analysis, showing cases where the global constraints contribute and pointing towards important areas for future work.

2 A Document-level Sentiment Model

Given a news document d, and named entities e_1,...,e_n in d, where each entity e_i has mentions $m_{i1} \cdots m_{ik}$, the task is to decide directed sentiment between all pairs of entities. We predict the directed sentiment from e_i to e_j at the document level, i.e., sent($e_i \rightarrow e_j$) $\in$ {positive, unbiased, negative}, for all e_i, $e_j \in d$ where $i \neq j$, assuming that sentiment is consistent within the document.

We introduce a document-level ILP that includes base models and soft social constraints. ILP has been used successfully for a wide range of NLP tasks (Roth and Yih, 2004), perhaps because they easily support incorporating different types of global constraints. We use two base models: (1) a learned pairwise sentiment classifier (Sec 3.1) that combines sentence- and discourse-level features to make predictions for each entity pair and (2) a pattern-based faction extractor (Sec 3.2) that detects alliances among a subset of the entities.

The ILP is solved by maximizing:

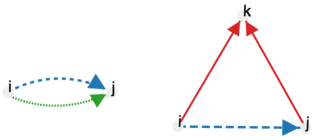
$$F = \psi_{social} + \psi_{fact} + \sum_{i=1}^{n}\sum_{j=1}^{n} \psi_{ij}$$

where F combines soft constraints (ψ_{social}, ψ_{fact} defined in detail in this section) with pairwise potentials ψ_{ij} defined as:

[1] All data will be made publicly available. You can browse it at `http://homes.cs.washington.edu/~eunsol/project_page/acl16`, and download it from the author's webpage.

Sentence	i	j
Canadian Prime Minister Harper. . .	Canada	Harper
. . . Reid, the Democratic leader. . .	Reid	Democratic
Goldman spokesman DuVally	Goldman	DuVally
. . . Djibouti, a key U.S. ally.	Djibouti	U.S.

(a) Detection examples



(b) Visual representation of common inference patterns.

Figure 3: An example sentiment inference from faction relationships. Pairs in factions are encouraged to share opinions, and to be positive towards other tied entities. On the right, sentiment edges can be both positive or both negative.

$$\psi_{ij} = \phi_{pos_{ij}} \cdot \texttt{pos}_{ij} + \phi_{neg_{ij}} \cdot \texttt{neg}_{ij} + \phi_{neu_{ij}} \cdot \texttt{neu}_{ij}$$

Each potential ψ_{ij} includes the sentiment classifier scores (ϕ_{pos}, ϕ_{neg}, ϕ_{neu}) with binary variables $\texttt{pos}_{ij}$, $\texttt{neu}_{ij}$ and $\texttt{neg}_{ij}$ where, for example, $\texttt{neg}_{ij}$=1 indicates that e_i is negative towards e_j. Decision variables $\texttt{pos}_{ij}$ and $\texttt{neu}_{ij}$ are defined analogously for positive and neutral opinion. Finally, we introduce a hard constraint:

$$\forall i, j \; \texttt{pos}_{ij} + \texttt{neg}_{ij} + \texttt{neu}_{ij} = 1$$

to ensure a single prediction is made per pair.

2.1 Inference with factions

Our first soft ILP constraint ψ_{fact} models that fact that entities in supportive social relations tend to share similar sentiment toward others (Lazarsfeld and Merton, 1954), and are often positive towards each other. For now, we assume access to a base extractor to provide such faction relations (Sec. 3.2 provides details of our pattern-based extractor). Figure 3a illustrates sample detections.

We introduce a binary variable $\texttt{tie}_{ij}$, where $\texttt{tie}_{ij}$=1 denotes an extracted faction relationship. These variables are tied to the variables regarding sentiment via the variables

$$\begin{aligned}
\texttt{tie_same}_{ijk} &= \texttt{tie}_{ij} \wedge \texttt{pos}_{ik} \wedge \texttt{pos}_{jk} \\
&\quad + \texttt{tie}_{ij} \wedge \texttt{neg}_{ik} \wedge \texttt{neg}_{jk} \\
\texttt{tie_diff}_{ijk} &= \texttt{tie}_{ij} \wedge \texttt{pos}_{ik} \wedge \texttt{neg}_{jk} \\
&\quad + \texttt{tie}_{ij} \wedge \texttt{neg}_{ik} \wedge \texttt{pos}_{jk} \\
\texttt{itself}_{ij} &= \texttt{tie}_{ij} \wedge \texttt{pos}_{ij} - \texttt{tie}_{ij} \wedge \texttt{neg}_{ij}
\end{aligned}$$

which are used in the following objective term:

$$\psi_{fact} = \sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_{itself} \cdot \texttt{itself}_{ij} + \sum_{k=1}^{n}(\alpha_{fact} \cdot (\texttt{tie\_same}_{ijk} - \texttt{tie\_diff}_{ijk})))$$

This formulation enables the model to predict implicit sentiment by jointly considering factual and sentiment relations among other entity pairs, essentially drawing a connection between sentiment analysis and information extraction. Figure 3 visualizes this inference pattern.

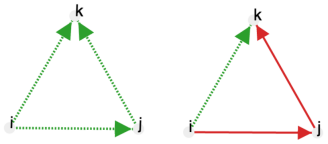


Figure 4: Balance Theory Constraints. When i is positive towards j, sharing same sentiment towards k define a balanced state. When i is negative towards j, differing opinions towards k define a balanced state.

2.2 Inference with sentiment relations

We also include constraints ψ_{social} in the objective that model social balance and reciprocity.

Balance theory constraints: Social balance theory (Heider, 1946) models the sentiment dynamics in an interpersonal network. In particular, in balanced states, entities on positive terms have similar opinions towards other entities and those on negative terms have opposing opinions. We introduce a set of variables to capture this insight: for example, the case where e_i is positive towards e_j is shown below (analogous when negative).

$$\begin{aligned}
\texttt{pos_same}_{ijk} &= \texttt{pos}_{ij} \wedge \texttt{pos}_{ik} \wedge \texttt{pos}_{jk} \\
&\quad + \texttt{pos}_{ij} \wedge \texttt{neg}_{ik} \wedge \texttt{neg}_{jk} \\
\texttt{pos_diff}_{ijk} &= \texttt{pos}_{ij} \wedge \texttt{neg}_{ik} \wedge \texttt{pos}_{ik} \\
&\quad + \texttt{pos}_{ij} \wedge \texttt{pos}_{ik} \wedge \texttt{neg}_{ik}
\end{aligned}$$

and add the term ψ_{bl} to ψ_{social}.

$$\psi_{bl} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(\alpha_{bl} \cdot (\texttt{pos\_same}_{ijk} + \texttt{neg\_diff}_{ijk}) + \alpha_{bad_{bl}} \cdot (\texttt{pos\_diff}_{ijk} + \texttt{neg\_same}_{ijk}))$$

A visualization of these constraints is in Figure 4.

	Faction	Balance	Reciprocity
POS	57%	64%	73%
NEG	60%	61%	78%

Table 1: Percentage of labels where each constraint holds. For example, positive on reciprocity means when $pos(e_i, e_j)$ is true, 73% of times $pos(e_j, e_i)$ is also true.

Reciprocity constraint: Reciprocity of sentiment has been recognized as a key aspect of social stability (Johnston, 1916; Gouldner, 1960). To model reciprocity among the real world entities, we introduce variables:

$$\texttt{r\_same}_{ij} = \texttt{pos}_{ij} \wedge \texttt{pos}_{ji} + \texttt{neg}_{ij} \wedge \texttt{neg}_{ji}$$
$$\texttt{r\_diff}_{ij} = \texttt{pos}_{ij} \wedge \texttt{neg}_{ji} + \texttt{neg}_{ij} \wedge \texttt{pos}_{ji}$$
$$\psi_r = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_r(\texttt{r\_same}_{ij}) + \alpha_{bad_r}(\texttt{r\_diff}_{ij})$$

and add the term ψ_r to the ψ_{social}.

2.3 Discussion

While many studies exist on homophily, social balance, and reciprocity, no prior work has reported quantitative analysis on the sentiment dynamics among the real world entities that appear in unstructured text. Thus we report the data statistics based on the development set in Table 1. We find that the global constraints hold commonly but are not universal, motivating the use of soft constraints (see Sec. 6).

3 Pairwise Base Models

The global model in Sec. 2 uses two base models, one for pairwise sentiment classification and the other for detecting faction relationships.

3.1 Sentiment Classifier

The entity-pair classifier considers a holder entity e_i, its mentions $m_{i1} \cdots m_{ip}$, a target entity e_j, its mentions $m_{j1} \cdots m_{jq}$, and document d. It predicts sent($e_i \rightarrow e_j$) $\in$ {positive, unbiased, negative}. The input is plain text and no gold labels are assumed; entity detection, dependency parse and co-reference resolution are automatic, and include common nouns and pronoun mentions (details in Sec. 4.1). We trained separate classifiers for pairs that co-occur in a sentence and those that do not, using a linear class-weighted SVM classifier with crowd-sourced data described in Sec. 4.2.

In what follows, we describe three different types of features we developed: dependency features, document features, and quotation features. Many of the features test the overall sentiment of a set of words (e.g., the complete document, a dependency path, or a quotation). In each case, we define the *sentiment label* for the text to be positive if it contains more words that appear in the positive sentiment lexicon than that appear in the negative one (and similarly for the negative label). We used MPQA sentiment lexicon (Wilson et al., 2005) for our study, which contains 2,718 positive and 4,912 negative lexicons.

Dependency Features We consider all dependency paths between the head word of e_i and e_j in each sentence, and aggregate over all co-occurring sentences. The features compute: (1) The sentiment label of the path containing `dobj` and `nsubj_rev`, up to length three if the path contains sentiment lexicon words (e.g., *Olympic* hero *Skah* accuses *Norway* over custody battle.) (2) The sentiment label of the path e_i $\uparrow$ `nsubj` $\downarrow$ `ccomp` $\downarrow$ `nsubj` $\downarrow e_j$, when it exists (e.g., *McCully* said any action against *Henry* is a matter entirely for TVNZ) (3) The sentiment label of path when the path does not contain any named entity (e.g., *Nobel* winner , *Shirin Ebadi*) (4) An indicator for the link `nmod:against`.

Document Features Previous work has shown that notions related to salience (e.g., proximity to sentiment words) can help to detect sentiment targets (Ben-Ami et al., 2014). In our data, we found that an entity's occurrence pattern is highly indicative of being involved in sentiment, for example the most frequently mentioned entity is 3.4 times more likely to be polarized and an entity in the headline is two times more likely to be polarized.

Pairwise features include the NER type of e_i and e_j and the percentage of sentences they co-occur in. We also use features indicating whether e_i and e_j (1) are mentioned in the headline and (2) appear only once in the document. When they are the two most frequent entities, we add the document sentiment label as a feature. For entity pairs that do not appear together in any sentence, we also include the rank of holder and target in terms of overall number of mentions in the document.

Quotation Features Quotations often involve subjective opinions towards prominent entities in news articles. Thus we include document-level

features encoding this intuition. For example, the sentence *"We're pleased to put this behind us," said Michael DuVally* implies positive sentiment from DuVally. We extract direct quotations using regular expressions. We include the sentiment label of the direct quotation from the speaker to the entities in it, excluding entities that appear less than three times in the document. We add the sentiment label of the quotation as a feature to (speaker, the most frequent entity) pair as well.

To extract indirect quotations, we follow studies (Bethard et al., 2004; Lu, 2010) and use a list of 20 verbs indicating speech events (e.g., say, speak, and announce) to detect direct quotations and their opinion holders. We then add the sentiment label of words connected to e_j via a dependency path of length up to two that also includes the subject of quotation verb to e_j (e.g. *Hassanal* said that cooperation between *Brunei* and *China* were fruitful). We also include an indicator feature for whether e_i is the subject of the quotation verb.

3.2 Faction Detector

We use a simple pattern-based detector that extracts a faction relationship between a pair of entities if the dependency path between them either:

1. contains only one link of modifier or compound label (`nmod`, `nmod : poss`, `amod`, `nn`, or `compound`).
2. or contains less than three links and has a possessive or appositive label (`poss` or `appos`).

Example extractions for this approach, which we adopted for its simplicity and the fact that it works reasonably well in practice, are shown in Figure 3a. On average we detect 1.7 ties per document on a small development set with roughly 30% recall and 60% precision. Improving performance and adding more relation types is an important area for future work.[2]

4 Data

We collected new datasets that densely label sentiment among entities in news articles, including: 208 documents, 2,226 sentences, and 15,185 entity pair labels. It complements existing datasets such as MPQA which provides rich annotations at the sentence-level (Deng and Wiebe, 2015b) and the recent KBP challenge which provides sparse annotations at the corpus-level (Ellis et al., 2014), by providing document-level annotations for all entity pairs (see Sec. 7 for discussion).

	KBP	MPQA	Crowdsourced
Document count	154	54	914
Avg. sentence count	10.0	12.7	14.8
Avg. entity count	7.9	10.6	8.8
Avg. mentions / entity	3.6	2.7	3.5

Table 2: Corpus Statistics

4.1 Document Preprocessing

All-pair annotation can be expensive, as there are N^2 pairs to annotate for each document with N entities. We determined that it would be more cost efficient to cover a large number of short documents than a small number of very long documents. We therefore selected articles with less than eleven entities from KBP and less than fifteen from MPQA and took the first 15 sentences for annotation. We used Stanford CoreNLP (Manning et al., 2014) for sentence splitting, part-of-speech tagging, named entity recognition, co-reference resolution and dependency parsing. We discarded entities of type date, duration, money, time and number and merged named entities using several heuristics, such as merging acronyms, merging named entity of person type with the same last name (e.g., Tiger Woods to Woods). We merged names listed as alias in when there is an exact match from Freebase. We included all mentions in a co-reference chain with the named entity, discarding chains with more than one entity. The corpus statistics are shown in Table 2.

4.2 Sentiment Data Collection

We annotated data using two methods: freelancers ($7.6 per article on average) covering all entity pairs and crowd-sourcing ($1.6 per article on average) covering a subset of entity pairs.

Evaluation Dataset We provide exhaustive annotations covering all pairs for the evaluation set. We hired freelancers from UpWork,[3] after examining performance on five documents. They labeled entity pairs with one of the following classes.

POS: positive towards the target.
NOTNEG: positive or unbiased towards the target.

[2] We experimented with using relations from an external knowledge base (Freebase), but KB sparsity and entity linking errors posed major challenges.

[3] `https://www.upwork.com`

Label	KBP	MPQA
POS	3.93	3.52
NOT NEG	5.73	8.06
UNBIASED	44.64	91.04
NOT POS	2.73	6.70
NEG	2.27	2.94

Table 3: Sentiment Label Statistics. Each count represents the average number per document.

UNB: unbiased towards the target
NOTPOS: negative or unbiased towards the target.
NEG: negative towards the target.

Here, we introduced the NOTPOS and NOTNEG classes to mark more subjective cases where we expect agreement might be lower. For example, one assigned NOTPOS to sentiment(Goldman, FINRA), *The FINRA said Goldman lacked adequate procedures to ...* and another assigned NOTNEG to sentiment(Macalintal, Arroyo) in the next example. *... Arroyo's election lawyer, Romulo Macalintal*. Arguments could be made for NEG or POS, respectively, but the decision is inherently subjective and requires careful reading.[4]

We also asked annotators to mark the label as inferred when not explicitly stated but implied from the context or world knowledge. Allowing for inferred labels and finer-grained labels encouraged annotators to capture implicit sentiment. For each judgement, we acquired two labels. Inter-annotator agreement, in Table 4, is high for the relaxed metrics, confirming our intuitions about the ambiguity of the NOTNEG and NOTPOS labels.

For experiments, we combine the fine grained labels as follows: POS or NEG is assigned when both marked it as such. When only one of the annotators marked it, we assigned the weaker sentiment (POS to NOTNEG, NEG to NOTPOS). NOTNEG and NOTPOS are assigned when either annotator marked it without 'Inferred' label. When the labels contradict in polarity or the labels are inferred weaker sentiment, UNB was assigned.

Crowdsourced Dataset We also randomly selected news articles from the Gigaword corpus,[5] and collected labels to train the base sentiment classifier (Sec. 3.1). We designed a pipelined approach, with three steps:

1. Document selection: Is there sentiment among entities in this document?
2. Entity selection: (1) Select all entities **holding** sentiment towards any other entities., and (2) Select all entities which are the **target** of sentiment by any other entity.
3. Sentiment label collection: Choose the sentiment A has towards B, from {Positive, No Sentiment, Negative}

We used CrowdFlower,[6] where annotators were randomly presented test questions for quality control. We collected labels from three annotators for each entity pair, and considered labels when at least two agreed. The resulting annotation contains total 2,995 labels on 914 documents, 682 positive, 836 negative and 474 without sentiment, which we discarded.

	Exact	Strict	Relaxed
Positive	0.35	0.54	0.67
Negative	0.50	0.64	0.74

Table 4: Inter-annotator Agreement. Cohen's kappa score: Exact counts only exact matches, Strict counts allows NOT NEG labels to match POS, and Relaxed allows NOT NEG to match POS or UNBIASED (analogously for negative).

	POS	NOT NEG	NOT POS	NEG
KBP	25%	29%	30%	28%
MPQA	35%	49%	46%	50%

Table 5: Percentage of entity pairs that do not co-occur in a sentence.

	POS	NOTNEG	NOTPOS	NEG
KBP	70%	94%	88%	58%
MPQA	68%	74%	83%	66%

Table 6: Percentage of labels marked as inferred.

4.3 Insights Into Data

This data supports the study of sentiment-laden entity pairs across sentence boundaries and inferred labels among entities, as we show here.

Sentiment Beyond Sentence Boundary Approximately 25% of polarized sentiment labels are between entities that do not co-occur[7] in a sentence (see Table 5). For example, in the article

[4] In the construction of MPQA3.0 dataset, entity-entity/event sentiment corpus, even with iterative expert annotation, 31% of disagreements are caused by negligence.

[5] LDC2014E13:TAC2014KBP English Corpus

[6] http://www.crowdflower.com

[7] This is an estimate due to co-reference resolution errors.

with headline 'Russia heat, smog trigger health problems',

> . . . "We never care to work with a future perspective in mind," *Alexei Skripkov* of the *Federal Medical and Biological Agency* said. "It's a big systemic mistake."

Skripkov never appears together with Russia in any sentence, but he manifests negative sentiment towards it. When a document revolves around a theme (in this example Russia), sentiment is often directed to it without being explicitly mentioned.

Inferred sentiment Annotators marked labels as inferred frequently, especially on less polarized sentiment (see Table 6). Various clues led to sentiment inference. For example, in the following document, we can read *Sam Lake*'s positive attitude towards *Paul Auster* from his 'citing' action:

> Ask most video-game designers about their inspirations . . . *Sam Lake* cites *Paul Auster*'s "*The Book of Illusions*"

Sentiment can also be inferred through reasoning over another entity.

> The *U.N.* imposed an embargo against *Eritrea* for helping insurgents opposed to the *Somali* government.

By considering relations with Eritrea, we can infer U.N. would be positive towards Somalia.

5 Experimental Setup

Data and Metrics We randomly split the densely labeled KBP document set, using half as a test data and half as a development data. One half of the development set was used to tune hyper parameters,[8] and the other for error analysis and ablations. After development, we ran on the test sets composed of KBP documents and MPQA documents. For MPQA we did not create a separate development set and reserved all of the relatively modest amount of data for a more reliable test set. For the pairwise classifier, we report development results using five-fold cross validation on the training data.

We report macro-averaged precision, recall, and F-measure for both sentiment labels.

Comparison Systems We compare performance to two simple baselines and two adaptations of existing sentiment classifiers. The baselines include our base pairwise classifier (Pair) and randomly assigning labels according to their empirical distribution (Random).

The first existing method adaptation (Sentence) uses the publicly released sentence-level RNN sentiment model from Socher et al (2013). For each entity pair, we collect sentiment labels from sentences they co-occur in and assign a positive label if a positive-labeled sentence exists, negative if there exists more than one sentence with a negative label and no positives.[9]

We also report a proxy for doing similar aggregation over a state-of-the-art entity-entity sentiment classifier. Here, because we added our new labels to the original KBP and MPQA3.0 annotations, we can simply predict the union of the original gold annotations using mention string overlap to align the entities (KM_Gold). This provides a reasonable upper bound on the performance of any extractor trained on this data.[10]

Implementation Details We use CPLEX4[11] to solve the ILP described in Sec. 2. For computational efficiency and to avoid erroneous propagation, soft constraints associated with reciprocity and balance theory are introduced only on pairs for which a high-precision classifier assigned polarity. For the pairwise classifier, we use a class-weighted linear SVM.[12] We include annotated pairs, and randomly sample negative examples from pairs without a label in the crowd-sourced training dataset. We made two versions of pairwise classifiers by tuning weight on polarized classes and negative sampling ratio by grid search. One is tuned for high precision to be used as a base classifier for ILP (ILP base), and the other is tuned for the best F1 (Pairwise).[13]

6 Results

Table 7 shows results on the evaluation datasets. The global model achieves the best F1 on both labels. All systems do significantly better than the random baseline but, overall, we see that entity-entity sentiment detection is challenging, requir-

[8] We used the following values (α_r, α_{bad_r}, α_{itself}, $\alpha_{faction}$, α_{bl}, $\alpha_{bad_{bl}}$) = (0.7, -0.8, 0.4, 0.5, 0.1, -0.5).

[9] Due to domain difference, the system predicted negative labels more (73% of sentences were classified as negative).

[10] We consider this gold evaluation a direct proxy for the recent work Deng and Wiebe (2015a), which is the most related recent entity-entity sentiment model trained on the gold data whose predictions we are evaluating against.

[11] http://tinyurl.com/joccfqy

[12] http://scikit-learn.org/

[13] We use 10 as the weights for the polarized classes. Pairwise and base classifier for MPQA sampled 4%, base classifier for KBP sampled 10% of unlabeled pairs.

	Development Set (KBP)						KBP						MPQA					
	Positive			Negative			Positive			Negative			Positive			Negative		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
KM_Gold	90.9	2.5	4.8	93.8	8.6	15.8	93.9	4.3	8.3	93.5	6.6	12.4	61.5	1.3	2.5	90.0	5.2	9.8
Random	16.6	13.1	14.7	4.9	4.0	4.4	13.3	12.7	13.0	10.1	6.9	8.2	10.9	15.4	12.8	8.9	6.7	7.7
Sentence	**60.0**	16.3	25.7	21.7	**43.1**	28.8	40.9	20.6	27.4	21.0	31.4	25.2	18.9	3.7	6.2	16.7	18.2	17.4
Pairwise	47.3	36.9	41.4	25.6	36.8	30.2	36.2	**35.5**	35.9	27.6	**41.2**	33.1	**28.7**	23.0	25.6	**23.2**	16.3	19.2
Global	58.2	**37.9**	**45.9**	**37.2**	35.1	**36.1**	**45.5**	32.7	**38.1**	**34.6**	36.8	**35.7**	25.2	**29.3**	**27.1**	17.6	**24.4**	**20.4**

Table 7: Performance on the evaluation datasets: including implicit and explicit sentiment.

	Positive			Negative		
	P	R	F1	P	R	F1
ILP base	56.7	25.2	34.9	36.9	27.6	31.6
+ Reci.	53.5	30.0	38.4	33.9	33.9	33.9
+ Balance	49.6	30.4	37.7	32.0	32.8	32.4
+ Faction	58.9	30.2	39.9	37.6	33.9	35.6

Table 8: ILP constraints ablation study.

	Positive			Negative		
	P	R	F1	P	R	F1
All	34.5	39.7	36.9	35.7	37.6	36.6
- Depend.	32.9	32.1	32.5	31.7	38.5	34.8
- Doc.	32.6	41.0	35.8	39.4	23.8	28.0
- Quotation	33.6	39.5	36.3	34.5	34.6	34.6

Table 9: Pairwise classifier feature ablation study.

ing identification of holders, targets, and sentiment jointly. While the numbers are not directly comparable, the best performing system for KBP 2014 sentiment task achieved F1 score of 25.7.

The first row (KM_Gold) shows the comparison against gold annotations from different datasets, highlighting the differences between the task definitions. Our annotations are much more dense, while KBP focuses on specific query entities and MPQA has a much broader focus with less emphasis on covering all entity pairs. The high precision suggests that all of the approaches agree when considering the same entity pairs.

The global model also improves performance over the pairwise classifier (Pairwise) for both datasets, but we see very different behavior due to the different sentiment label distributions (see Table 3). The KBP data has many fewer unbiased pairs and many mistakes are from choosing the wrong polarity. For the pairwise classifier 17% of all predictions were assigned the opposite polarity. After the global inference, it is reduced to 11%, contributing to the gain in overall precision. For MPQA the base classifier has a more challenging detection task, due to relatively large amount of the unbaised pairs. Here, the best base classifier misses many pairs and the global model helps to fill in some of these gaps in recall.

In both cases, the document-level model often propagates correct labels by detecting easier, explicit expressions. For example, given the sentence *Buphavanh said Laos creates favorable conditions for Vietnamese companies*, the base classifier detected positive sentiment from Buphavanh to Vietnam, but not between Vietnam and Laos. By detecting the fact that Buphavanh is the prime minister of Laos, it infers the extra sentiment pairs.

Sentiment expression detection error	21.0%
Missing world knowledge	19.3%
Named entity detection error	17.5%
Co-reference failure	14.8%
Propagation error	12.3%
Missing faction	7.0%

Table 10: Error Analysis on the development set.

We also did ablation studies to measure the contributions of different components. Table 8 shows ablations of each soft constraint. The faction constraint is the most helpful, improving both precision and recall for both labels. The reciprocity and social balance constraints tend to improve recall at the cost of precision. Table 9 shows ablations of the base classifier features. All features are helpful, with dependency features most helpful for positive labels, and quotation and document-level features more with negatives.

Error Analysis We manually analyzed errors on 20 articles from the development set (Table 10). Our system failed when there were sentiment words not in the lexicon, or negated sentiment words. Capturing subtle sentiment expressions beyond sentiment lexicon should improve the performance. Preprocessing, as a whole, was the largest source of error. It includes co-reference failure and named entity error. Co-reference mistakes happen as a result of not resolving pronouns, referring expressions, as well as named entities co-references (e.g., Financial Industry Regulatory Authority to FINRA), or erroneously merging them. Lengthy quotations or nested mentions triggered co-reference error, affecting mostly recall. Named entity errors includes incorrect named

entity detection (e.g., pro-Israel) and mention detection boundary errors. For example, we detected negative sentiment from Mexico to Pakistan from *Mexico condemns Pakistan series suicide bomb attacks.* While actual sentiment is positive. Finally, the ILP propagates sentiment labels erroneously at times. Our constraints often hold among entities of the same type, but are less predictive among entities of different types. For example, when a person supports a peace treaty, the treaty does not have sentiment towards him/her. For future work refining constraints based on entity type should help performance.

7 Related Work

Sentiment Inference Our sentiment inference task is related to the recent KBP sentiment task,[14] in that we aim to find opinion target and holder. While we study the complete document-level analysis over all entity pairs, the KBP task is formulated as query-focused retrieval of entity sentiment from a large pool of potentially relevant documents. Thus, their annotations focus only on query entities and relatively sparse compared to ours (see Sec. 6). Another recent dataset is MPQA 3.0 (Deng and Wiebe, 2015b), which captures various aspects of sentiment. Their sentiment pair annotations are only at the sentence-level and are therefore much sparser than we provide (see Sec. 6) for entity-entity relation analysis.

Several recent studies focused on various aspects of implied sentiment (Greene and Resnik, 2009; Mohammad and Turney, 2010; Zhang and Liu, 2011; Feng et al., 2013; Deng and Wiebe, 2014; Deng et al., 2014). Deng and Wiebe (2015a) in particular introduced sentiment implicature rules relevant for sentence-level entity-entity sentiment. Our work contributes to these recent efforts by presenting a new model and dataset for document-level sentiment inference over all entity pairs.

Document-level Analysis Stoyanov and Claire (2011) also studied document-level sentiment analysis based on fine-grained detection of directed sentiment. They aggregate sentence-level detections to make document-level predictions, while our we model global coherency among entities and can discover implied sentiment without direct sentence-level evidence. In the event extraction domain, previous research showed the effectiveness of jointly considering multiple sentences. Yang and Mitchell (2016) proposed joint extraction of entities and events with the document context, improving on the event extraction. Most work focuses on events, while we primarily study sentiment relations.

Social Network Analysis While many previous studies considered the effect of social dynamics for social media analysis, most relied on an explicitly available social network structure or considered dialogues and speech acts for which opinion holders are given (Tan et al., 2011; Hu et al., 2013; Li et al., 2014; West et al., 2014; Krishnan and Eisenstein, 2015). Compared to the recent work that focused on relationships among fictional characters in movie summaries and stories (Chaturvedi et al., 2016; Srivastava et al., 2016; Iyyer et al., 2016), we consider a broader types of named entities on news domains.

8 Conclusion

We presented an approach to interpreting sentiment among entities in news articles, with global constraints provided by social, faction and discourse context. Experiments demonstrated that the approach can infer implied sentiment and point toward potential directions for future work, including joint entity detection and incorporation of more varied types of factual relationships.

Acknowledgments

This research was supported in part by the NSF (IIS-1252835, IIS-1408287, IIS-1524371), DARPA under the DEFT program through the AFRL (FA8750-13-2-0019), an Allen Distinguished Investigator Award, and a gift from Google. This material is also based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1256082. The authors thank the members of UW NLP group for discussions and support. We also thank the anonymous reviewers for insightful comments. Finally, we thank the annotators from the CrowdFlower and UpWork.

[14] http://www.nist.gov/tac/2014/KBP/Sentiment

References

Zvi Ben-Ami, Ronen Feldman, and Binyamin Rosenfeld. 2014. Entities' sentiment relevance. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Steven Bethard, Hong Yu, Ashley Thornton, Vasileios Hatzivassiloglou, and Dan Jurafsky. 2004. Automatic extraction of opinion propositions and their holders. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text: Theories and Applications*.

Snigdha Chaturvedi, Shashank Srivastava, Hal Daumé III, and Chris Dyer. 2016. Modeling evolving relationships between characters in literary novels. In *Proceedings of the National Conference on Artificial Intelligence*.

Lingjia Deng and Janyce Wiebe. 2014. Sentiment propagation via implicature constraints. In *Proceedings of the Conference of the European Chapter of the Association for Computational Linguistics*.

Lingjia Deng and Janyce Wiebe. 2015a. Joint prediction for entity/event-level sentiment analysis using probabilistic soft logic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Lingjia Deng and Janyce Wiebe. 2015b. Mpqa 3.0: An entity/event-level sentiment corpus. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Lingjia Deng, Janyce Wiebe, and Yoonjung Choi. 2014. Joint inference and disambiguation of implicit sentiments via implicature constraints. In *Proceedings of International Conference on Computational Linguistics*.

Joe Ellis, Jeremy Getman, and Stephanie M Strassel. 2014. Overview of linguistic resources for the tac kbp 2014 evaluations: Planning, execution, and results. In *Proceedings of TAC KBP 2014 Workshop, National Institute of Standards and Technology*, pages 17–18.

Song Feng, Jun Seok Kang, Polina Kuznetsova, and Yejin Choi. 2013. Connotation lexicon: A dash of sentiment beneath the surface meaning. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August. Association for Computational Linguistics.

Alvin W. Gouldner. 1960. The norm of reciprocity: A preliminary statement. *American Sociological Review*, 25(2).

Stephan Greene and Philip Resnik. 2009. More than words: Syntactic packaging and implicit sentiment. In *Proceedings the North American Chapter of the Association for Computational Linguistics*, Boulder, Colorado, June. Association for Computational Linguistics.

Fritz Heider. 1946. Attitudes and cognitive organization. *The Journal of psychology*, 21(1).

Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of the ACM international conference on Web search and data mining*. ACM.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of North American Association for Computational Linguistics*.

G. A. Johnston. 1916. *International Journal of Ethics*, 26(2).

Vinodh Krishnan and Jacob Eisenstein. 2015. "You're Mr. Lebowski, I'm The Dude": Inducing address term formality in signed social networks. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Paul F Lazarsfeld and Robert K Merton. 1954. Friendship as a social process: A substantive and methodological analysis. *Freedom and control in modern society*, 18:18–66.

Jiwei Li, Alan Ritter, and Eduard Hovy. 2014. Weakly supervised user profile extraction from twitter. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Bin Lu. 2010. Identifying opinion holders and targets with dependency parser in chinese news texts. In *Proceedings of the NAACL HLT Student Research Workshop*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Saif M. Mohammad and Peter D. Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, CAAGET '10.

D. Roth and W. Yih. 2004. A linear programming formulation for global inference in natural language tasks. In *Proceedings of Conference on Natural Language Learning*.

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

Shashank Srivastava, Snigdha Chaturvedi, and Tom Mitchell. 2016. Inferring interpersonal relations in narrative summaries. In *Proceedings of the National Conference on Artificial Intelligence*.

Veselin Stoyanov and Claire Cardie. 2011. Automatically Creating General-Purpose Opinion Summaries from Text. In *Proceedings of Recent Advances in Natural Language Processing*.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of Knowldege Discovery and Data Mining*.

Robert West, Hristo S Paskov, Jure Leskovec, and Christopher Potts. 2014. Exploiting social network structure for person-to-person sentiment analysis. *In the Proceedings of Transactions of the Association for Computational Linguistics*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Human Language Technologies Conference/Conference on Empirical Methods in Natural Language Processing*.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of Association for Computational Linguistics*.

Bishan Yang and Tom Mitchell. 2016. Joint extraction of events and entities within a document context. In *North American Association for Computational Linguistics*.

Lei Zhang and Bing Liu. 2011. Identifying noun product features that imply opinions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Active Learning for Dependency Parsing with Partial Annotation

Zhenghua Li[†], **Min Zhang**[†*], **Yue Zhang**[†], **Zhanyi Liu**[‡],
Wenliang Chen[†], **Hua Wu**[‡], **Haifeng Wang**[‡]

† Soochow University, Suzhou, China

{zhli13,minzhang,wlchen}@suda.edu.cn, zhangyue1107@qq.com

‡ Baidu Inc., Beijing, China

{liuzhanyi,wu_hua,wanghaifeng}@baidu.com

Abstract

Different from traditional active learning based on sentence-wise full annotation (FA), this paper proposes active learning with dependency-wise partial annotation (PA) as a finer-grained unit for dependency parsing. At each iteration, we select a few most uncertain words from an unlabeled data pool, manually annotate their syntactic heads, and add the partial trees into labeled data for parser retraining. Compared with sentence-wise FA, dependency-wise PA gives us more flexibility in task selection and avoids wasting time on annotating trivial tasks in a sentence. Our work makes the following contributions. First, we are the first to apply a probabilistic model to active learning for dependency parsing, which can 1) provide tree probabilities and dependency marginal probabilities as principled uncertainty metrics, and 2) directly learn parameters from PA based on a forest-based training objective. Second, we propose and compare several uncertainty metrics through simulation experiments on both Chinese and English. Finally, we conduct human annotation experiments to compare FA and PA on real annotation time and quality.

1 Introduction

During the past decade, supervised dependency parsing has gained extensive progress in boosting parsing performance on canonical texts, especially on texts from domains or genres similar to existing manually labeled treebanks (Koo and Collins, 2010; Zhang and Nivre, 2011). However, the upsurge of web data (e.g., tweets, blogs, and product comments) imposes great challenges to existing parsing techniques. Meanwhile, previous research on out-of-domain dependency parsing gains little success (Dredze et al., 2007; Petrov and McDonald, 2012). A more feasible way for open-domain parsing is to manually annotate a certain amount of texts from the target domain or genre. Recently, several small-scale treebanks on web texts have been built for study and evaluation (Foster et al., 2011; Petrov and McDonald, 2012; Kong et al., 2014; Wang et al., 2014).

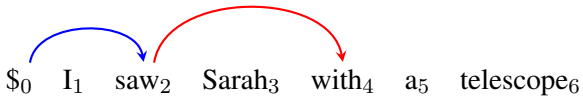


Figure 1: A partially annotated sentence, where only the heads of "*saw*" and "*with*" are decided.

Meanwhile, active learning (AL) aims to reduce annotation effort by choosing and manually annotating unlabeled instances that are most valuable for training statistical models (Olsson, 2009). Traditionally, AL utilizes full annotation (FA) for parsing (Tang et al., 2002; Hwa, 2004; Lynn et al., 2012), where a whole syntactic tree is annotated for a given sentence at a time. However, as commented by Mejer and Crammer (2012), the annotation process is complex, slow, and prone to mistakes when FA is required. Particularly, annotators waste a lot of effort on labeling trivial dependencies which can be well handled by current statistical models (Flannery and Mori, 2015).

Recently, researchers report promising results with AL based on partial annotation (PA) for dependency parsing (Sassano and Kurohashi, 2010; Mirroshandel and Nasr, 2011; Majidi and Crane, 2013; Flannery and Mori, 2015). They find

*Correspondence author.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 344–354,
Berlin, Germany, August 7-12, 2016.

that smaller units rather than sentences provide more flexibility in choosing potentially informative structures to annotate.

Beyond previous work, this paper endeavors to more thoroughly study this issue, and has made substantial progress from the following perspectives.

(1) This is the first work that applies a state-of-the-art probabilistic parsing model to AL for dependency parsing. The CRF-based dependency parser on the one hand allows us to use probabilities of trees or marginal probabilities of single dependencies for uncertainty measurement, and on the other hand can directly learn parameters from partially annotated trees. Using probabilistic models may be ubiquitous in AL for relatively simpler tasks like classification and sequence labeling, but is definitely novel for dependency parsing which is dominated by linear models with perceptron-like training.

(2) Based on the CRF-based parser, we make systematic comparison among several uncertainty metrics for both FA and PA. Simulation experiments show that compared with using FA, AL with PA can greatly reduce annotation effort in terms of dependency number by 62.2% on Chinese and by 74.2% on English.

(3) We build a visualized annotation platform and conduct human annotation experiments to compare FA and PA on real annotation time and quality, where we obtain several interesting observations and conclusions.

All codes, along with the data from human annotation experiments, are released at `http://hlt.suda.edu.cn/~zhli` for future research study.

2 Probabilistic Dependency Parsing

Given an input sentence $\mathbf{x} = w_1...w_n$, the goal of dependency parsing is to build a directed dependency tree $\mathbf{d} = \{h \curvearrowright m : 0 \leq h \leq n, 1 \leq m \leq n\}$, where $|\mathbf{d}| = n$ and $h \curvearrowright m$ represents a dependency from a *head* word h to a *modifier* word m. Figure 1 depicts a partial tree containing two dependencies.[1]

In this work, we for the first time apply a probabilistic CRF-based parsing model to AL for dependency parsing. We adopt the second-order graph-based model of McDonald and Pereira (2006), which casts the problem as finding an optimal tree from a fully-connect directed graph and factors the score of a dependency tree into scores of pairs of sibling dependencies.

$$\begin{aligned} \mathbf{d}^* &= \arg\max_{\mathbf{d}\in\mathcal{Y}(\mathbf{x})} Score(\mathbf{x}, \mathbf{d}; \mathbf{w}) \\ Score(\mathbf{x}, \mathbf{d}; \mathbf{w}) &= \sum_{\substack{(h,s,m): h\curvearrowright s\in\mathbf{d}, \\ h\curvearrowright m\in\mathbf{d}}} \mathbf{w}\cdot\mathbf{f}(\mathbf{x}, h, s, m) \end{aligned} \tag{1}$$

where s and m are adjacent siblings both modifying h; $\mathbf{f}(\mathbf{x}, h, s, m)$ are the corresponding feature vector; $\mathbf{w}$ is the feature weight vector; $\mathcal{Y}(\mathbf{x})$ is the set of all legal trees for $\mathbf{x}$ according to the dependency grammar in hand; $\mathbf{d}^*$ is the 1-best parse tree which can be gained efficiently via a dynamic programming algorithm (Eisner, 2000). We use the state-of-the-art feature set listed in Bohnet (2010).

Under the log-linear CRF-based model, the probability of a dependency tree is:

$$p(\mathbf{d}|\mathbf{x}; \mathbf{w}) = \frac{e^{Score(\mathbf{x},\mathbf{d};\mathbf{w})}}{\sum_{\mathbf{d}'\in\mathcal{Y}(\mathbf{x})} e^{Score(\mathbf{x},\mathbf{d}';\mathbf{w})}} \tag{2}$$

Ma and Zhao (2015) give a very detailed and thorough introduction to CRFs for dependency parsing.

2.1 Learning from FA

Under the supervised learning scenario, a labeled training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{d}_i)\}_{i=1}^N$ is provided to learn $\mathbf{w}$. The objective is to maximize the log likelihood of $\mathcal{D}$:

$$\mathcal{L}(\mathcal{D}; \mathbf{w}) = \sum_{i=1}^{N} \log p(\mathbf{d}_i|\mathbf{x}_i; \mathbf{w}) \tag{3}$$

which can be solved by standard gradient descent algorithms. In this work, we adopt stochastic gradient descent (SGD) with L2-norm regularization for all CRF-based parsing models.[2]

[1] In this work, we follow many previous works to focus on unlabeled dependency parsing (constructing the skeleton dependency structure). However, the proposed techniques explored in this paper can be easily extended to the case of labeled dependency parsing.

[2] We borrow the implementation of SGD in CRFsuite (`http://www.chokkan.org/software/crfsuite/`), and use 100 sentences for a batch.

2.2 Marginal Probability of Dependencies

Marcheggiani and Artières (2014) shows that marginal probabilities of local labels can be used as an effective uncertain metric for AL for sequence labeling problems. In the case of dependency parsing, the marginal probability of a dependency is the sum of probabilities of all legal trees that contain the dependency.

$$p(h \curvearrowright m|\mathbf{x};\mathbf{w}) = \sum_{\mathbf{d}\in\mathcal{Y}(\mathbf{x}):h\curvearrowright m\in\mathbf{d}} p(\mathbf{d}|\mathbf{x};\mathbf{w}) \quad (4)$$

Intuitively, marginal probability is a more principled metric for measuring reliability of a dependency since it considers all legal parses in the search space, compared to previous methods based on scores of local classifiers (Sassano and Kurohashi, 2010; Flannery and Mori, 2015) or votes of n-best parses (Mirroshandel and Nasr, 2011). Moreover, Li et al. (2014) find strong correlation between marginal probability and correctness of a dependency in cross-lingual syntax projection.

3 Active Learning for Dependency Parsing

This work adopts the standard pool-based AL framework (Lewis and Gale, 1994; McCallum and Nigam, 1998). Initially, we have a small set of labeled seed data $\mathcal{L}$, and a large-scale unlabeled data pool $\mathcal{U}$. Then the procedure works as follows.

(1) Train a new parser on the current $\mathcal{L}$.

(2) Parse all sentences in $\mathcal{U}$, and select a set of the most informative tasks $\mathcal{U}'$

(3) Manually annotate: $\mathcal{U}' \rightarrow \mathcal{L}'$

(4) Expand labeled data: $\mathcal{L} \cup \mathcal{L}' \rightarrow \mathcal{L}$

The above steps loop for many iterations until a predefined stopping criterion is met.

The key challenge for AL is how to measure the informativeness of structures in concern. Following previous work on AL for dependency parsing, we make a simplifying assumption that if the current model is most uncertain about an output (sub)structure, the structure is most informative in terms of boosting model performance.

3.1 Sentence-wise FA

Sentence-wise FA selects K most uncertain sentences in Step (2), and annotates their whole tree structures in Step (3). In the following, we describe several uncertainty metrics and investigate their practical effects through experiments. Given an unlabeled sentence $\mathbf{x} = w_1...w_n$, we use $\mathbf{d}^*$ to denote the 1-best parse tree produced by the current model as in Eq. (1). For brevity, we omit the feature weight vector $\mathbf{w}$ in the equations.

Normalized tree score. Following previous works that use scores of local classifiers for uncertainty measurement (Sassano and Kurohashi, 2010; Flannery and Mori, 2015), we use $Score(\mathbf{x}, \mathbf{d}^*)$ to measure the uncertainty of $\mathbf{x}$, assuming that the model is more uncertain about $\mathbf{x}$ if $\mathbf{d}^*$ gets a smaller score. However, we find that directly using $Score(\mathbf{x}, \mathbf{d}^*)$ always selects very short sentences due to the definition in Eq. (1). Thus we normalize the score with the sentence length n as follows.[3]

$$Confi(\mathbf{x}) = \frac{Score(\mathbf{x}, \mathbf{d}^*)}{n^{1.5}} \quad (5)$$

Normalized tree probability. The CRF-based parser allows us, for the first time in AL for dependency parsing, to directly use tree probabilities for uncertainty measurement. Unlike previous approximate methods based on k-best parses (Mirroshandel and Nasr, 2011), tree probabilities globally consider all parse trees in the search space, and thus are intuitively more consistent and proper for measuring the reliability of a tree. Our initial assumption is that the model is more uncertain about $\mathbf{x}$ if $\mathbf{d}^*$ gets a smaller probability. However, we find that directly using $p(\mathbf{d}^*|\mathbf{x})$ would select very long sentences because the solution space grows exponentially with sentence length. We find that the normalization strategy below works well.[4]

$$Confi(\mathbf{x}) = \sqrt[n]{p(\mathbf{d}^*|\mathbf{x})} \quad (6)$$

Averaged marginal probability. As discussed in Section 2.2, the marginal probability of a dependency directly reflects its reliability, and thus can be regarded as another global measurement besides tree probabilities.In fact, we find that the effect of sentence length is naturally handled with the following metric.[5]

$$Confi(\mathbf{x}) = \frac{\sum_{h\curvearrowright m\in\mathbf{d}^*} p(h \curvearrowright m|\mathbf{x})}{n} \quad (7)$$

[3]We have also tried replacing $n^{1.5}$ with n (still prefer short sentences) and n^2 (bias to long sentences).

[4]We have also tried $p(\mathbf{d}^*|\mathbf{x}) \times f(n)$, where $f(n) = \log n$ or $f(n) = \sqrt{n}$, but both work badly.

[5]We have also tried $\sqrt[n]{\prod_{h\curvearrowright m\in\mathbf{d}^*} p(h \curvearrowright m|\mathbf{x})}$, leading to slightly inferior results.

3.2 Single Dependency-wise PA

AL with single dependency-wise PA selects M most uncertain words from $\mathcal{U}$ in Step (2), and annotates the heads of the selected words in Step (3). After annotation, the newly annotated sentences with partial trees $\mathcal{L}'$ are added into $\mathcal{L}$. Different from the case of sentence-wise FA, $\mathcal{L}'$ are also put back to $\mathcal{U}$, so that new tasks can be further chosen from them.

Marcheggiani and Artières (2014) make systematic comparison among a dozen uncertainty metrics for AL with PA for several sequence labeling tasks. We borrow three effective metrics according to their results.

Marginal probability max. Suppose $h^0 = \arg\max_h p(h \curvearrowright i|\mathbf{x})$ is the most likely head for i. The intuition is that the lower $p(h^0 \curvearrowright i)$ is, the more uncertain the model is on deciding the head of the token i.

$$Confi(\mathbf{x}, i) = p(h^0 \curvearrowright i|\mathbf{x}) \quad (8)$$

Marginal probability gap. Suppose $h^1 = \arg\max_{h \neq h^0} p(h \curvearrowright i|\mathbf{x})$ is the second most likely head for i. The intuition is that the smaller the probability gap is, the more uncertain the model is about i.

$$Confi(\mathbf{x}, i) = p(h^0 \curvearrowright i|\mathbf{x}) - p(h^1 \curvearrowright i|\mathbf{x}) \quad (9)$$

Marginal probability entropy. This metric considers the entropy of all possible heads for i. The assumption is that the smaller the *negative* entropy is, the more uncertain the model is about i.

$$Confi(\mathbf{x}, i) = \sum_h p(h \curvearrowright i|\mathbf{x}) \log p(h \curvearrowright i|\mathbf{x}) \quad (10)$$

3.3 Batch Dependency-wise PA

In the framework of single dependency-wise PA, we assume that the selection and annotation of dependencies in the same sentence are strictly independent. In other words, annotators may be asked to annotate the head of one selected word after reading and understanding a whole (sometimes partial) sentence, and may be asked to annotate another selected word in the same sentence in next AL iteration. Obviously, frequently switching sentences incurs great waste of cognitive effort,

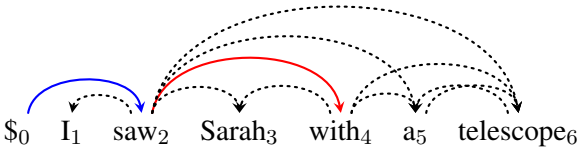


Figure 2: An example parse forest converted from the partial tree in Figure 1.

and annotating one dependency can certainly help decide another dependency in practice.

Inspired by the work of Flannery and Mori (2015), we propose AL with batch dependency-wise PA, which is a compromise between sentence-wise FA and single dependency-wise PA. In Step 2, AL with batch dependency-wise PA selects K most uncertain sentences from $\mathcal{U}$, and also determines $r\%$ most uncertain words from each sentence at the same time. In Step 3, annotators are asked to label the heads of the selected words in the selected sentences. We propose and experiment with the following three strategies based on experimental results of sentence-wise FA and single dependency-wise PA.

Averaged marginal probability & gap. First, select K sentences from $\mathcal{U}$ using averaged marginal probability. Second, select $r\%$ words using marginal probability gap for each selected sentence.

Marginal probability gap. First, for each sentence in $\mathcal{U}$, select $r\%$ most uncertain words according to marginal probability gap. Second, select K sentences from $\mathcal{U}$ using the averaged marginal probability gap of the selected $r\%$ words in a sentence as the uncertainty metric.

Averaged marginal probability. This strategy is the same with the above strategy, except it measures the uncertainty of a word i according to the marginal probability of the dependency pointing to i in $\mathbf{d}^*$, i.e., $p(j \curvearrowright i|\mathbf{x})$, where $j \curvearrowright i \in \mathbf{d}^*$.

3.4 Learning from PA

A major challenge for AL with PA is how to learn from partially labeled sentences, as depicted in Figure 1. Li et al. (2014) show that a probabilistic CRF-based parser can naturally and effectively learn from PA. The basic idea is converting a partial tree into a forest as shown in Figure 2,

and using the forest as the gold-standard reference during training, also known as *ambiguous labeling* (Riezler et al., 2002; Täckström et al., 2013).

For each remaining word without head, we add all dependencies linking to it as long as the new dependency does not violate the existing dependencies. We denote the resulting forest as $\mathcal{F}$j, whose probability is naturally the sum of probabilities of each tree $\mathbf{d}$ in $\mathcal{F}$.

$$\begin{aligned} p(\mathcal{F}|\mathbf{x};\mathbf{w}) &= \sum_{\mathbf{d}\in\mathcal{F}} p(\mathbf{d}|\mathbf{x};\mathbf{w}) \\ &= \frac{\sum_{\mathbf{d}\in\mathcal{F}} e^{Score(\mathbf{x},\mathbf{d};\mathbf{w})}}{\sum_{\mathbf{d}'\in\mathcal{Y}(\mathbf{x})} e^{Score(\mathbf{x},\mathbf{d}';\mathbf{w})}} \end{aligned} \tag{11}$$

Suppose the partially labeled training data is $\mathcal{D} = \{(\mathbf{x}_i, \mathcal{F}_i)\}_{i=1}^{N}$. Then its log likelihood is:

$$\mathcal{L}(\mathcal{D};\mathbf{w}) = \sum_{i=1}^{N} \log p(\mathcal{F}_i|\mathbf{x}_i;\mathbf{w}) \tag{12}$$

Täckström et al. (2013) show that the partial derivative of the $\mathcal{L}(\mathcal{D};\mathbf{w})$ with regard to $\mathbf{w}$ (a.k.a the gradient) in both Equation (3) and (12) can be efficiently solved with the classic Inside-Outside algorithm.[6]

4 Simulation Experiments

We use Chinese Penn Treebank 5.1 (*CTB*) for Chinese and Penn Treebank (*PTB*) for English. For both datasets, we follow the standard data split, and convert original bracketed structures into dependency structures using Penn2Malt with its default head-finding rules. To be more realistic, we use automatic part-of-speech (POS) tags produced by a state-of-the-art CRF-based tagger (94.1% on *CTB*-test, and 97.2% on *PTB*-test, n-fold jackknifing on training data), since POS tags encode much syntactic annotation. Because AL experiments need to train many parsing models, we throw out all training sentences longer than 50 to speed up our experiments. Table 1 shows the data statistics.

Following previous practice on AL with PA (Sassano and Kurohashi, 2010; Flannery and Mori, 2015), we adopt the following AL settings for both Chinese and English . The first 500 training sentences are used as the seed labeled data $\mathcal{L}$. In the case of FA, $K = 500$ new sentences are selected and annotated at each iteration. In the case of single dependency-wise PA, we select and annotate $M = 10,000$ dependencies, which roughly correspond to 500 sentences considering that the averaged sentence length is about 22.3 in *CTB*-train and 23.2 in *PTB*-train. In the case of batch dependency-wise PA, we set $K = 500$, and $r = 20\%$ for Chinese and $r = 10\%$ for English, considering that the parser trained on all data achieves about 80% and 90% accuracies.

		Train	Dev	Test
Chinese	#Sentences	14,304	803	1,910
	#Tokens	318,408	20,454	50,319
English	#Sentences	39,115	1,700	2,416
	#Tokens	908,154	40,117	56,684

Table 1: Data statistics.

We measure parsing performance using the standard unlabeled attachment score (UAS) *including punctuation marks*. Please note that we always treat punctuation marks as ordinary words when selecting annotation tasks and calculating UAS, in order to make fair comparison between FA and PA.[7]

4.1 FA vs. Single Dependency-wise PA

First, we make comparison on the performance of AL with FA and with single dependency-wise PA.

Results on Chinese are shown in Figure 3. Following previous work, we use the number of annotated dependencies (x-axis) as the annotation cost in order to fairly compare FA and PA. We use FA with random selection as a baseline. We also draw the accuracy of the CRF-based parser trained on all training data, which can be regarded as the upper bound.

For FA, the curve of the normalized tree score intertwines with that of random selection. Meanwhile, the performance of normalized tree probability is very close to that of averaged marginal probability, and both are clearly superior to the baseline with random selection.

For PA, the difference among the three uncertainty metrics is small. The marginal probability gap clearly outperforms the other two metrics before 50,000 annotated dependencies, and remains

[6]This work focuses on projective dependency parsing. Please refer to Koo et al. (2007), McDonald and Satta (2007), and Smith and Smith (2007) for building a probabilistic non-projective parser.

[7]Alternatively, we can exclude punctuation marks for task selection in AL with PA. Then, to be fair, we have to discard all dependencies pointing to punctuation marks in the case of FA. This makes the experiment setting more complicated.

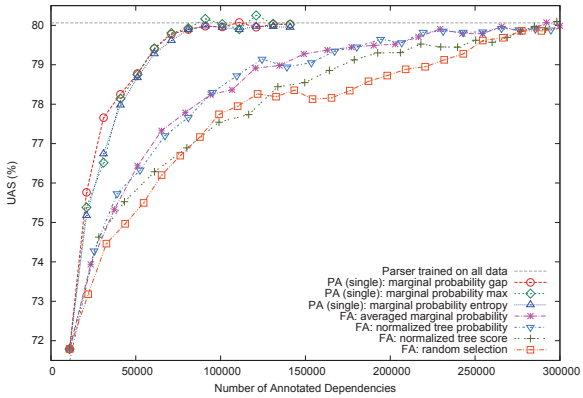


Figure 3: FA vs. PA on *CTB*-dev.

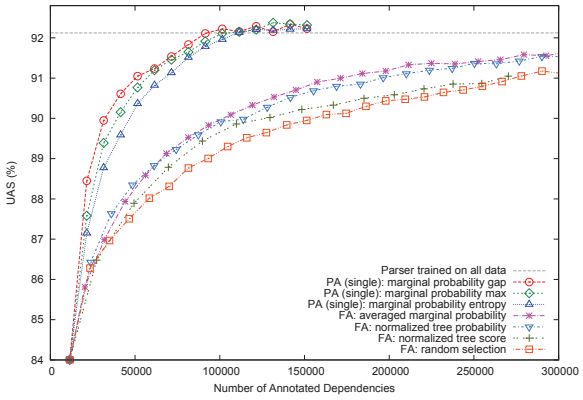


Figure 4: FA vs. PA on *PTB*-dev.

very competitive at all other points. The marginal probability max achieves best peak UAS, and even outperforms the parser trained on all data, which can be explained by small disturbance during complex model training. The marginal probability entropy, although being the most complex metric among the three, seems inferior all the time.

It is clear that using PA can greatly reduce annotation effort compared with using FA in terms of annotated dependencies.

Results on English are shown in Figure 4. The overall findings are similar to those in Figure 3, except that the distinction among different methods is more clear. For FA, normalized tree score is consistently better than the random baseline. Normalized tree probability always outperforms normalized tree score. Averaged marginal probability performs best, except being slightly inferior to normalized tree probability in earlier stages.

For PA, it is consistent that marginal probability gap is better than marginal probability max, and marginal probability entropy is the worst.

In summary, based on the results on the development data in Figure 3 and 4, the best AL method with PA only needs about $\frac{80{,}000}{318{,}408} = 25\%$ annotated dependencies on Chinese, and about $\frac{90{,}000}{908{,}154} = 10\%$ on English, to reach the same performance with parsers trained on all data. Moreover, the PA methods converges much faster than the FA ones, since for the same x-axis number, much more sentences (with partial trees) are used as training data for AL with PA than FA.

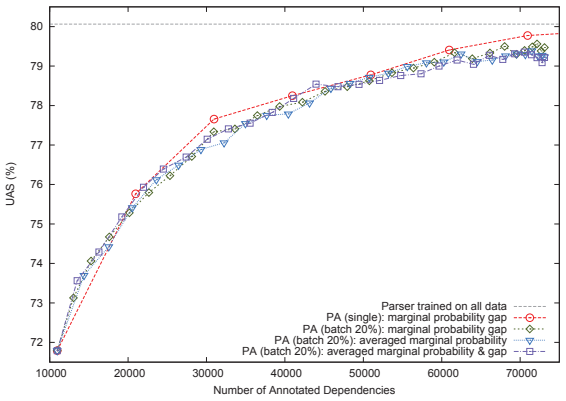


Figure 5: Single vs. batch dependency-wise PA on *CTB*-dev.

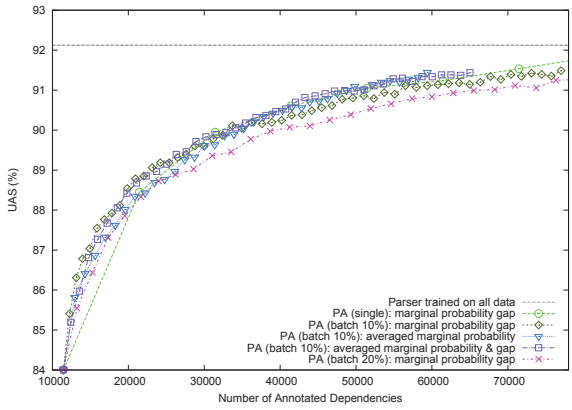


Figure 6: Single vs. batch dependency-wise PA on *PTB*-dev.

4.2 Single vs. Batch Dependency-wise PA

Then we make comparison on AL with single dependency-wise PA and with the more practical batch dependency-wise PA.

Results on Chinese are shown in Figure 5. We can see that the three strategies achieve very similar performance and are also very close to single dependency-wise PA. AL with batch dependency-wise PA even achieves higher accuracy before $20,000$ annotated dependencies, which should be caused by the smaller active learning steps (about

$2,000$ dependencies at each iteration, contrasting $10,000$ for single dependency-wise PA). When the training data runs out at about $7,300$ dependencies, AL with batch dependency-wise PA only lags behind with single dependency-wise PA by about 0.3%, which we suppose can be reduced if larger training data is available.

Results on English are shown in Figure 6, and are very similar to those on Chinese. One tiny difference is that the marginal probability gap is slightly worse that the other two metrics. The three uncertainty metrics have very similar accuracy curves, which are also very close to the curve of single dependency-wise PA. In addition, we also try $r = 20\%$ and find that results are inferior to $r = 10\%$, indicating that the extra 10% annotation tasks are less valuable and contributive.

4.3 Main Results on Test Data

Table 2 shows the results on test data. We compare our CRF-based parser with ZPar v6.0[8], a state-of-the-art transition-based dependency parser (Zhang and Nivre, 2011). We train ZPar with default parameter settings for 50 iterations, and choose the model that performs best on dev data. We can see that when trained on all data, our CRF-based parser outperforms ZPar on both Chinese and English.

To compare FA and PA, we report the number of annotated dependencies needed under each AL strategy to achieve an accuracy lower by about 1% than the parser trained on all data.[9]

FA (best) refers to FA with averaged marginal probability, and it needs $\frac{187,123-149,051}{187,123} = 20.3\%$ less annotated dependencies than FA with random selection on Chinese, and $\frac{395,199-197,907}{395,199} = 50.0\%$ less on English.

PA (single) with marginal probability gap needs $\frac{149,051-50,958}{149,051} = 65.8\%$ less annotated dependencies than FA (best) on Chinese, and $\frac{197,907-61,448}{197,907} = 69.0\%$ less on English.

PA (batch) with marginal probability gap needs slightly more annotation than PA (single) on Chinese but slightly less annotation on English, and can reduce the amount of annotated dependencies by $\frac{149,051-56,389}{149,051} = 62.2\%$ over FA (best) on Chinese and by $\frac{197,907-51,016}{197,907} = 74.2\%$ on English.

	Chinese		English	
	#Dep labeled	UAS	#Dep labeled	UAS
ZPar	318,408	77.97	908,154	91.45
This parser	318,408	78.36	908,154	91.66
FA (random)	187,123	77.43	395,199	90.67
FA (best)	149,051	77.32	197,907	90.66
PA (single)	50,958	77.22	61,448	90.72
PA (batch)	56,389	77.38	51,016	90.70

Table 2: Results on test data.

5 Human Annotation Experiments

So far, we measure annotation effort in terms of the number of annotated dependencies and assume that it takes the same amount of time to annotate different words, which is obviously unrealistic. To understand whether active learning based on PA can really reduce annotation time over based on FA in practice, we build a web browser based annotation system,[10] and conduct human annotation experiments on Chinese.

In this part, we use *CTB* 7.0 which is a newer and larger version and covers more genres, and adopt the newly proposed Stanford dependencies (de Marneffe and Manning, 2008; Chang et al., 2009) which are more understandable for annotators.[11] Since manual syntactic annotation is very difficult and time-consuming, we only keep sentences with length $[10, 20]$ in order to better measure annotation time by focusing on sentences of reasonable length, which leave us $12,912$ training sentences under the official data split. Then, we use a random half of training sentences to train a CRF-based parser, and select 20% most uncertain words with marginal probability gap for each sentence of the left half.

We employ 6 postgraduate students as our annotators who are at different levels of familiarity in syntactic annotation. Before annotation, the annotators are trained for about two hours by introducing the basic concepts, guidelines, and illustrating examples. Then, they are asked to practice on the annotation system for about another two hours. Finally, all annotators are required to

[8] http://people.sutd.edu.sg/~yue_zhang/doc/

[9] The gap 1% is chosen based on the curves on development data (Figure 3 and 4) with the following two considerations: 1) larger gap may lead to wrong impression that AL is weak; 2) smaller gap (e.g., 0.5%) cannot be reached for the worst AL method (FA: random).

[10] http://hlt-service.suda.edu.cn/syn-dep-batch. Please try.

[11] We use Stanford Parser 3.4 (2014-06-16) for constituent-to-dependency structure conversion.

	Time: Sec/Dep		Annotation accuracy	
	FA	PA	FA (on 20%)	PA (diff)
Annotator #1	**4.0**	**7.9**	**84.65** (**73.41**)	**75.28** (+1.87)
Annotator #2	7.5	16.0	78.90 (72.22)	62.18 (*-10.04*)
Annotator #3	*10.0*	*22.2*	69.75 (59.77)	56.91 (-2.86)
Annotator #4	5.1	8.7	66.75 (49.19)	61.77 (+**12.58**)
Annotator #5	7.0	17.3	65.47 (48.50)	48.39 (-0.11)
Annotator #6	7.0	10.6	*58.05* (*43.28*)	*48.37* (+5.09)
Overall	6.7	13.6	70.36 (57.28)	59.06 (+1.78)

Table 3: Statistics of human annotation.

formally annotate the same 100 sentences. The system is programed that each sentence has 3 FA submissions and 3 PA submissions. During formal annotation, the annotators are not allowed to discuss with each other or look up any guideline or documents, which may incur unnecessary inaccuracy in timing. Instead, the annotators can only decide the syntactic structures based on the basic knowledge of dependency grammar and one's understanding of the sentence structure. The annotation process lasts for about 5 hours. On average, each annotator completes 50 sentences with FA (763 dependencies) and 50 sentences with PA (178 dependencies).

Table 3 lists the results in descending order of an annotator's experience in syntactic annotation. The first two columns compare the time needed for annotating a dependency in seconds. On average, *annotating a dependency in PA takes about twice as much time as in FA*, which is reasonable considering the words to be annotated in PA may be more difficult for annotators while the annotation of some tasks in FA may be very trivial and easy. Combined with the results in Table 2, we may infer that to achieve 77.3% accuracy on *CTB*-test, AL with FA requires $149,051 \times 6.7 = 998,641.7$ seconds of annotation, whereas AL with batch dependency-wise PA needs $56,389 \times 13.6 = 766,890.4$ seconds. Thus, we may roughly say that *AL with PA can reduce annotation time over FA by* $\frac{998,641.7-766,890.4}{998,641.7} = 23.2\%$.

We also report annotation accuracy according to the gold-standard Stanford dependencies converted from bracketed structures.[12] Overall, the accuracy of FA is $70.36 - 59.06 = 11.30\%$ higher than that of PA, which should be due to the trivial tasks in FA. To be more fair, we compare the accuracies of FA and PA on the same 20% selected difficult words, and find that annotators exhibit different responses to the switch. Annotator #4 achieve 12.58% higher accuracy when under PA than under FA. The reason may be that under PA, annotators can be more focused and therefore perform better on the few selected tasks. In contrast, some annotators may perform better under FA. For example, annotation accuracy of annotator #2 increases by 10.04% when switching from PA to FA, which may be due to that FA allows annotators to spend more time on the same sentence and gain help from annotating easier tasks. Overall, we find that the accuracy of PA is $59.06 - 57.28 = 1.78\%$ higher than that of FA, indicating that *PA actually can improve annotation quality*.

[12] An anonymous reviewer commented that the direct comparison between an annotator's performance on PA and FA based on accuracy may be misleading since the FA and PA sentences for one annotator are mutually exclusive.

6 Related Work

Recently, AL with PA attracts much attention in sentence-wise natural language processing such as sequence labeling and parsing. For sequence labeling, Marcheggiani and Artières (2014) systematically compare a dozen uncertainty metrics in token-wise AL with PA (without comparison with FA), whereas Settles and Craven (2008) investigate different uncertainty metrics in AL with FA. Li et al. (2012) propose to only annotate the most uncertain word boundaries in a sentence for Chinese word segmentation and show promising results on both simulation and human annotation experiments. All above works are based on CRFs and make extensive use of sequence probabilities and token marginal probability.

In parsing community, Sassano and Kurohashi (2010) select bunsetsu (similar to phrases) pairs with smallest scores from a local classifier, and let annotators decide whether the pair composes a dependency. They convert partially annotated instances into local dependency/non-dependency classification instances to help a simple shift-reduce parser. Mirroshandel and Nasr (2011) select most uncertain words based on votes of n-best parsers, and convert partial trees into full trees by letting a baseline parser perform constrained decoding in order to preserve partial annotation. Under a different query-by-committee AL framework, Majidi and Crane (2013) select most uncertain words using a committee of diverse parsers, and convert partial trees into full trees by letting

the parsers of committee to decide the heads of remaining tokens. Based on a first-order (pointwise) Japanese parser, Flannery and Mori (2015) use scores of a local classifier for task selection, and treat PA as dependency/non-dependency instances (Flannery et al., 2011). Different from above works, this work adopts a state-of-the-art probabilistic dependency parser, uses more principled tree probabilities and dependency marginal probabilities for uncertainty measurement, and learns from PA based on a forest-based training objective which is more theoretically sound.

Most previous works on AL with PA only conduct simulation experiments. Flannery and Mori (2015) perform human annotation to measure true annotation time. A single annotator is employed to annotate for two hours alternating FA and PA (33% batch) every fifteen minutes. Beyond their initial expectation, they find that the annotation time per dependency is nearly the same for FA and PA (different from our findings) and gives a few interesting explanations.

Under a non-AL framework, Mejer and Crammer (2012) propose an interesting light feedback scheme for dependency parsing by letting annotators decide the better one from top-2 parse trees produced by the current parsing model.

Hwa (1999) pioneers the idea of using PA to reduce manual labeling effort for constituent grammar induction. She uses a variant Inside-Outside re-estimation algorithm (Pereira and Schabes, 1992) to induce a grammar from PA. Clark and Curran (2006) propose to train a Combinatorial Categorial Grammar parser using partially labeled data only containing predicate-argument dependencies. Tsuboi et al. (2008) extend CRF-based sequence labeling models to learn from incomplete annotations, which is the same with Marcheggiani and Artières (2014). Li et al. (2014) propose a CRF-based dependency parser that can learn from partial tree projected from source-language structures in the cross-lingual parsing scenario. Mielens et al. (2015) propose to impute missing dependencies based on Gibbs sampling in order to enable traditional parsers to learn from partial trees.

7 Conclusions

This paper for the first time applies a state-of-the-art probabilistic model to AL with PA for dependency parsing. It is shown that the CRF-based parser can on the one hand provide tree probabilities and dependency marginal probabilities as principled uncertainty metrics and on the other hand elegantly learn from partially annotated data. We have proposed and compared several uncertainty metrics through simulation experiments, and show that AL with PA can greatly reduce the amount of annotated dependencies by 62.2% on Chinese 74.2% on English. Finally, we conduct human annotation experiments on Chinese to compare PA and FA on real annotation time and quality. We find that annotating a dependency in PA takes about 2 times long as in FA. This suggests that AL with PA can reduce annotation time by 23.2% over with FA on Chinese. Moreover, the results also indicate that annotators tend to perform better under PA than FA.

For future work, we would like to advance this study in the following directions. The first idea is to combine uncertainty and representativeness for measuring informativeness of annotation targets in concern. Intuitively, it would be more profitable to annotate instances that are both difficult for the current model and representative in capturing common language phenomena. Second, we so far assume that the selected tasks are equally difficult and take the same amount of effort for human annotators. However, it is more reasonable that human are good at resolving some ambiguities but bad at others. Our plan is to study which syntactic structures are more suitable for human annotation, and balance informativeness of a candidate task and its suitability for human annotation. Finally, one anonymous reviewer comments that we may use automatically projected trees (Rasooli and Collins, 2015; Guo et al., 2015; Ma and Xia, 2014) as the initial seed labeled data, which is cheap and interesting.

Acknowledgments

The authors would like to thank the anonymous reviewers for the helpful comments. We also thank Junhui Li and Chunyu Kit for reading our paper and giving many good suggestions. Particularly, Zhenghua is very grateful to many of his students: Fangli Lu, Qiuyi Yan, and Yue Zhang build the annotation system; Jiayuan Chao, Wei Chen, Ziwei Fan, Die Hu, Qingrong Xia, and Yue Zhang participate in data annotation. This work was supported by National Natural Science Foundation of China (Grant No. 61502325, 61525205, 61572338).

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING*, pages 89–97.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative reordering with Chinese grammatical relations features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation (SSST-3) at NAACL HLT 2009*, pages 51–59.

Stephen Clark and James Curran. 2006. Partial training for a lexicalized-grammar parser. In *Proceedings of the Human Language Technology Conference of the NAACL*, pages 144–151.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, João Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62.

Daniel Flannery and Shinsuke Mori. 2015. Combining active learning and partial annotation for domain adaptation of a japanese dependency parser. In *Proceedings of the 14th International Conference on Parsing Technologies*, pages 11–19.

Daniel Flannery, Yusuke Miayo, Graham Neubig, and Shinsuke Mori. 2011. Training dependency parsers from partially annotated corpora. In *Proceedings of IJCNLP*, pages 776–784.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From news to comment: Resources and benchmarks for parsing the language of web 2.0. In *Proceedings of IJCNLP*, pages 893–901.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of ACL*, pages 1234–1244.

Rebecca Hwa. 1999. Supervised grammar induction using training data with limited constituent information. In *Proceedings of ACL*, pages 73–79.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computional Linguistics*, 30(3):253–276.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proceedings of EMNLP*, pages 1001–1012.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *ACL*, pages 1–11.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of EMNLP-CoNLL*, pages 141–150.

David D. Lewis and William A. Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Shoushan Li, Guodong Zhou, and Chu-Ren Huang. 2012. Active learning for Chinese word segmentation. In *Proceedings of COLING 2012: Posters*, pages 683–692.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Soft cross-lingual syntax projection for dependency parsing. In *COLING*, pages 783–793.

Teresa Lynn, Jennifer Foster, Mark Dras, and Elaine UÍ Dhonnchadha. 2012. Active learning and the irish treebank. In *Proceedings of ALTA*.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL*, pages 1337–1348.

Xuezhe Ma and Hai Zhao. 2015. Probabilistic models for high-order projective dependency parsing. *Arxiv*, abs/1502.04174.

Saeed Majidi and Gregory Crane. 2013. Active learning for dependency parsing by a committee of parsers. In *Proceedings of IWPT*, pages 98–105.

Diego Marcheggiani and Thierry Artières. 2014. An experimental comparison of active learning strategies for partially labeled sequences. In *Proceedings of EMNLP*, pages 898–906.

Andrew McCallum and Kamal Nigam. 1998. Employing EM and pool-based active learning for text classification. In *Proceedings of ICML*, pages 350–358.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proceedings of the Tenth International Conference on Parsing Technologies*, pages 121–132.

Avihai Mejer and Koby Crammer. 2012. Training dependency parser using light feedback. In *Proceedings of NAACL*.

Jason Mielens, Liang Sun, and Jason Baldridge. 2015. Parse imputation for dependency annotations. In *Proceedings of ACL-IJCNLP*, pages 1385–1394.

Seyed Abolghasem Mirroshandel and Alexis Nasr. 2011. Active learning for dependency parsing using partially annotated sentences. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 140–149.

Fredrik Olsson. 2009. A literature survey of active machine learning in the context of natural language processing. Technical report, Swedish Institute of Computer Science.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proceedings of the Workshop on Speech and Natural Language (HLT)*, pages 122–127.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Mohammad Sadegh Rasooli and Michael Collins. 2015. Density-driven cross-lingual transfer of dependency parsers. In *Proceedings of EMNLP*, pages 328–338.

Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard Crouch, John T. III Maxwell, and Mark Johnson. 2002. Parsing the wall street journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 271–278.

Manabu Sassano and Sadao Kurohashi. 2010. Using smaller constituents rather than sentences in active learning for japanese dependency parsing. In *Proceedings of ACL*, pages 356–365.

Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of EMNLP*, pages 1070–1079.

David A. Smith and Noah A. Smith. 2007. Probabilistic models of nonprojective dependency trees. In *Proceedings of EMNLP-CoNLL*, pages 132–140.

Oscar Täckström, Ryan McDonald, and Joakim Nivre. 2013. Target language adaptation of discriminative transfer parsers. In *Proceedings of NAACL*, pages 1061–1071.

Min Tang, Xiaoqiang Luo, and Salim Roukos. 2002. Active learning for statistical natural language parsing. In *Proceedings of ACL*, pages 120–127.

Yuta Tsuboi, Hisashi Kashima, Hiroki Oda, Shinsuke Mori, and Yuji Matsumoto. 2008. Training conditional random fields using incomplete annotations. In *Proceedings of COLING*, pages 897–904.

William Yang Wang, Lingpeng Kong, Kathryn Mazaitis, and William W Cohen. 2014. Dependency parsing for weibo: An efficient probabilistic logic programming approach. In *Proceedings of EMNLP*, pages 1152–1158.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL*, pages 188–193.

Dependency Parsing with Bounded Block Degree and Well-nestedness via Lagrangian Relaxation and Branch-and-Bound

Caio Corro **Joseph Le Roux** **Mathieu Lacroix**
Antoine Rozenknop **Roberto Wolfler Calvo**
Laboratoire d'Informatique de Paris Nord,
Université Paris 13 – SPC, CNRS UMR 7030,
F-93430, Villetaneuse, France
{corro,leroux,lacroix,rozenknop,wolfler}@lipn.fr

Abstract

We present a novel dependency parsing method which enforces two structural properties on dependency trees: bounded block degree and well-nestedness. These properties are useful to better represent the set of admissible dependency structures in treebanks and connect dependency parsing to context-sensitive grammatical formalisms. We cast this problem as an Integer Linear Program that we solve with Lagrangian Relaxation from which we derive a heuristic and an exact method based on a Branch-and-Bound search. Experimentally, we see that these methods are efficient and competitive compared to a baseline unconstrained parser, while enforcing structural properties in all cases.

1 Introduction

We address the problem of enforcing two structural properties on dependency trees, namely bounded block degree and well-nestedness, without sacrificing algorithmic efficiency. Intuitively, bounded block degree constraints force each subtree to have a yield decomposable into a limited number of blocks of contiguous words, while well-nestedness asserts that every two distinct subtrees must not interleave: either the yield of one subtree is entirely inside some gap of the other or they are completely separated. These two types of constraints generalize the notion of projectivity: projective trees actually have a block degree bounded to one and are well-nested.

Our first motivation is the fact that most dependency trees in NLP treebanks are well-nested and have a low block degree which depends on the language and the linguistic representation, as shown in (Pitler et al., 2012). Unfortunately, although polynomial algorithms exist for this class of trees (Gómez-Rodríguez et al., 2009), they are not efficient enough to be of practical use in applications requiring syntactic structures. In addition, if either property is dropped, but not the other, then the underlying decision problem becomes harder. That is why practical parsing algorithms are either completely unconstrained (McDonald et al., 2005) or enforce strict projectivity (Koo and Collins, 2010). This work is, to the best of our knowledge, the first attempt to build a discriminative dependency parser that enforces well-nestedness and/or bounded block degree and to use it on treebank data.

We base our method on the following observation: a non-projective dependency parser, thus not requiring neither well-nestedness nor bounded block degree, returns dependency trees satisfying these constraints in the vast majority of sentences. This would tend to indicate that the heavy machinery involved to parse with these constraints is only needed in very few cases.

We consider arc-factored dependency parsing with well-nestedness and bounded block degree constraints. We formulate this problem as an Integer Linear Program (ILP) and apply Lagrangian Relaxation where the dualized constraints are those associated with bounded block degree and well-nestedness. The Lagrangian dual objective then reduces to a maximum spanning arborescence and can be solved very efficiently. This provides an efficient heuristic for our problem. An exact method can be derived by embedding this Lagrangian Relaxation in a Branch-and-Bound procedure to solve the problem with an optimality certificate. Despite the exponential worst-time complexity of the Branch-and-Bound procedure, it is tractable in practice. Our formulation can enforce both types of constraints or only one of them without changing the resolution method.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 355–366,
Berlin, Germany, August 7-12, 2016.

As stated in (Bodirsky et al., 2009), well-nested dependency trees with 2-bounded block degree are structurally equivalent to derivations in Lexicalized Tree Adjoining Grammars (LTAGs) (Joshi and Schabes, 1997).[1][2] While LTAGs can be parsed in polynomial time, developing an efficient parser for these grammars remains an open problem (Eisner and Satta, 2000) and we believe that this work could be a useful step in that direction.

Related work is reviewed in Section 2. We define arc-factored dependency parsing with block degree and well-nestedness constraints in Section 3. We derive an ILP formulation of this problem in Section 4 and then present our method based on Lagrangian Relaxation in Section 5 and Branch-and-Bound in Section 6. Section 7 contains experimental results on several languages.

2 Related Work

A dynamic programming algorithm has been proposed for parsing well-nested and k-bounded block degree dependency trees in (Gómez-Rodríguez et al., 2009; Gómez-Rodríguez et al., 2011). Unfortunately, it has a prohibitive $O(n^{3+2k})$ time complexity, equivalent to Lexicalized TAG parsing when $k = 2$. Variants of this algorithm have also been proposed for further restricted classes of dependency trees: 1-inherit ($O(n^6)$) (Pitler et al., 2012), head-split ($O(n^6)$) (Satta and Kuhlmann, 2014) and both 1-inherit and head-split ($O(n^5)$) (Satta and Kuhlmann, 2014). Although those restricted classes have good empirical coverage, they do not cover the exact search space of Lexicalized TAG derivation and their time complexity is still prohibitive. Spinal TAGs, described as a dependency parsing task in (Carreras et al., 2008), weaken even more the empirical coverage in practice, restricted to projective trees, but still remain hardly tractable with a complexity of $O(n^4)$. On the contrary, the present work does not restrict the search space.

Parsing mildly context-sensitive languages with dependencies has been explored in (Fernández-González and Martins, 2015) but the resulting parser cannot guarantee compliance with strict structural properties. On the other hand, the present method enforces the well-nestedness and bounded block degree of solutions.

The methods mentioned above all use the graph-based approach and rely on dynamic programming to achieve tractability. There is also a line of of work in transition-based parsing for various dependency classes. Systems have been proposed for projective dependency trees (Nivre, 2003), non-projective, or even unknown classes (Attardi, 2006). Pitler and McDonald (2015) propose a transition system for crossing interval trees, a more general class than well-nested trees with bounded block degree. In the case of spinal TAGs, we can mention the work of Ballesteros and Carreras (2015) and Shen and Joshi (2007). Transition-based algorithms offer low space and time complexity, typically linear in the length of sentences usually by relying on local predictors and beam strategies and thus do not provide any optimality guarantee on the produced structures. The present work follows the graph-based approach, but replaces dynamic programming with a greedy algorithm and Lagrangian Relaxation.

The use of Lagrangian Relaxation to elaborate sophisticated parsing models based on plain maximum spanning arborescence solutions originated in (Koo et al., 2010) where this method was used to parse with higher-order features. This technique has been explored to parse CCG dependencies in (Du et al., 2015) without a precise definition of the class of trees. We can also draw connections between our problem reduction procedure and the use of Lagrangian Relaxation to speed up dynamic programming and beam search with exact pruning in (Rush et al., 2013).

In this work we rely on Non-Delayed Relax-and-Cut for lazy constraint generation (Lucena, 2006). This can be linked to (Riedel, 2009) which uses a cutting plane algorithm to solve MAP inference in Markov Logic and (Riedel et al., 2012) which uses column and row generation for higher-order dependency parsing.

In NLP, the Branch-and-Bound framework (Land and Doig, 1960) has previously been used for dependency parsing with high order features in (Qian and Liu, 2013), and Das et al. (2012) combined Branch-and-Bound to Lagrangian Relaxation in order to retrieve integral solutions for shallow semantic parsing.

[1] It is possible to express a wider class of dependencies with LTAG if we allow dependencies direction to be different from the derivation tree (Kallmeyer and Kuhlmann, 2012).

[2] In order to be fully compatible with LTAGs, we must ensure that the root has only one child. For algorithmic issues see (Fischetti and Toth, 1992) or (Gabow and Tarjan, 1984).

3 Dependency Parsing

We model the dependency parsing problem using a graph-based approach. Given a sentence $s = \langle s_0, \dots, s_n \rangle$ where s_0 is a dummy root symbol, we consider the directed graph $D = (V, A)$ with $V = \{0, \dots n\}$ and $A \subseteq V \times V$. Vertex $i \in V$ corresponds to word s_i and arc $(i, j) \in A$ models a dependency from word s_i to word s_j. In the rest of the paper, we denote $V \setminus \{0\}$ by V^+.

An *arborescence* is a set of arcs T inducing a connected graph with no circuit such that every vertex has at most one entering arc. The set of vertices incident with any arc of T is denoted by $V[T]$. If $V[T] = V$, then T is a *spanning arborescence*. Among the vertices of $V[T]$, the one with no entering arc is called the *root* of T. A vertex t is *reachable* from a vertex s with respect to T if there exists a path from s to t using only arcs of T. The *yield* of a vertex $v \in V$ corresponds to the set of vertices reachable from v with respect to T.

It is well-known that there is a bijection between dependency trees for s and spanning arborescences with root 0 (McDonald et al., 2005). In what follows, the term dependency tree will refer to both the dependency tree of s and its associated spanning arborescence of D with root 0.

In the dependency parsing problem, one has to find a dependency tree with maximal score. Several scores can be associated with each dependency tree and different conditions can restrict the set of valid dependency trees.

In this paper, we consider an *arc-factored* model: each arc $(i, j) \in A$ is assigned a score w_{ij}; the score of a dependency tree is defined as the sum of the scores of the arcs it contains. This model can be computed in $O(n^2)$ with Chu–Liu-Edmonds' algorithm for Maximum Spanning Arborescence (MSA) (McDonald et al., 2005). Unfortunately, this algorithm forbids any modification of the score function, for example adding score contribution for combinations of arcs (i.e. grand-parent or sibling models). Moreover, adding score contribution for combinations of couple of arcs makes the problem NP-hard (McDonald and Pereira, 2006), although several methods have been developed to tackle this problem, for instance dual decomposition (Koo et al., 2010).

Likewise, restrictions on the tree structure such as the well-nestedness and bounded block degree conditions are not permitted in the MSA algorithm. We first give a precise definition of these structural properties, equivalent to (Bodirsky et al., 2009), before we present a method to take them into account. From now on, we suppose that instances are equipped with a positive integer k and we call *valid dependency trees* those satisfying the k-bounded block degree and well-nestedness conditions. A graph-theoretic definition of these two conditions can be given as follows.

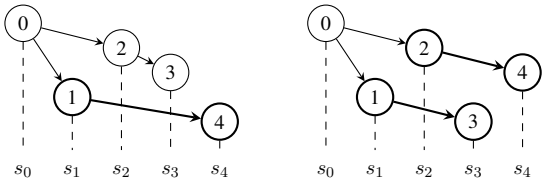


Figure 1: (Left) A 2-BBD arborescence: the block degree of vertex 1 is 2 (its yield is $\{1, 4\}$) whereas the block degree of all other vertices is 1. (Right) A not well-nested arborescence: the yields of vertices 1 and 2 interleave.

Block degree The *block degree* of a vertex set $W \subseteq V$ is the number of vertices of W without a predecessor[3] inside W. Given an arborescence T, the *block degree* of a vertex is the block degree of its yield and the *block degree* of T is the maximum block degree of its incident vertices. An arborescence satisfies the k-bounded block degree condition if its block degree is less than or equal to k. We then say it is *k-BBD* for short. Figure 1 (left) gives an example of a 2-BBD arborescence.

Well-nestedness Two disjoint subsets $I_1, I_2 \subset V^+$ *interleave* if there exist $i, j \in I_1$ and $k, l \in I_2$ such that $i < k < j < l$. An arborescence is *well-nested* if it is not incident to two vertices whose yields interleave. Figure 1 (right) shows an arborescence which is not well-nested.

4 ILP Formulation

In this section we formulate the dependency parsing problem described in Section 3 as an ILP. We start with some notation and two theorems characterizing k-BBD and well-nested dependency trees.

Given a subset $W \subseteq V$, the set of arcs entering W is denoted by $\delta^{\text{in}}(W)$ and the set of arcs leaving W is denoted by $\delta^{\text{out}}(W)$. The set $\delta(W) = \delta^{\text{in}}(W) \cup \delta^{\text{out}}(W)$ is called the *cut* of W. Given a positive integer l, let $\mathcal{W}^{\geq l}$ be the family of vertex subsets of V^+ with block degree greater than or equal to l. For instance, given any sentence with more than 6 words, $\{1, 3, 5, 6\} \in \mathcal{W}^{\geq 3}$,

[3]The predecessor of a vertex $v \in V$ is $v - 1$.

while $\{1,2,5,6\} \notin \mathcal{W}^{\geq 3}$. We also denote by $\mathcal{I}$ the family of couples of disjoint interleaving vertex subsets of V^+. For instance, $(\{1,4\},\{2,3,5\})$ belongs to $\mathcal{I}$. Finally, given a vector $x \in \mathbb{R}^A$ and a subset $B \subseteq A$, $x(B)$ corresponds to $\sum_{a\in B} x_a$.

Theorem 1. *A dependency tree T is not k-BBD iff there exists a vertex subset $W \in \mathcal{W}^{\geq k+1}$ whose cut $\delta(W)$ contains a unique arc of T.*

Proof. By definition of block degree, a dependency tree is not k-BBD iff it is incident with a vertex whose yield W belongs to $\mathcal{W}^{\geq k+1}$. It is equivalent to say that T contains a subarborescence T' such that $V[T']$ equals W. This holds iff W has one entering arc (since $0 \notin W$) and no leaving arc belonging to T. □

Theorem 2. *A dependency tree T is not well-nested iff there exists $(I_1, I_2) \in \mathcal{I}$ such that $\delta(I_1) \cap T$ and $\delta(I_2) \cap T$ are singletons.*

Proof. $\delta(I_1)$ and $\delta(I_2)$ both intersect T only once iff T contains two arborescences T_1 and T_2 such that $V[T_1] = I_1$ and $V[T_2] = I_2$. This means that T is incident with two vertices whose yields are I_1 and I_2, respectively. Result follows from the definition of $\mathcal{I}$ and well-nested arborescences. □

The dependency parsing problem can be formulated as follows. A dependency tree will be represented by its incidence vector. Hence, we use variables $z \in \mathbb{R}^A$ such that $z_a = 1$ if arc a belongs to the dependency tree and 0 otherwise.

$$\max_z \sum_{a\in A} w_a z_a \tag{1}$$

$$z(\delta^{\text{in}}(v)) = 1 \qquad \forall v \in V^+ \tag{2}$$

$$z(\delta^{\text{in}}(W)) \geq 1 \qquad \forall W \subseteq V^+ \tag{3}$$

$$z(\delta(W)) \geq 2 \qquad \forall\, W \in \mathcal{W}^{\geq k+1} \tag{4}$$

$$z(\delta(I_1)) + z(\delta(I_2)) \geq 3 \quad \forall (I_1, I_2) \in \mathcal{I} \tag{5}$$

$$z \in \{0,1\}^A \tag{6}$$

The objective function (1) maximizes the score of the dependency tree. Inequalities (2) ensure that all vertices but the root have one entering arc. Inequalities (3) force the set of arcs associated with z to induce a connected graph. Inequalities (2) and (3), together with $z \geq 0$, give a linear description of the convex hull of the incidence vectors of the spanning arborescences with root 0 — see *e.g.,* (Schrijver, 2003). Inequalities (4) ensure that the dependency tree is k-BBD and inequalities (5) impose well-nestedness. The validity of (4) and (5) follows from Theorems 1 and 2, respectively.

Remark that (3) could be replaced by a polynomial number of additional flow variables and constraints, see (Martins et al., 2009).[4]

5 Lagrangian Relaxation

Solving this ILP using an off-the-shelf solver is ineffective due to the huge number of constraints. We tackle this problem with Lagrangian Relaxation, which has become popular in the NLP community, see for instance (Rush and Collins, 2012). Note that contrary to most previous work on Lagrangian Relaxation for NLP, we do not use it to derive a decomposition method.

We note that optimizing objective (1) subject to constraints (2), (3) and (6) amounts to finding a MSA and can be solved combinatorially (McDonald et al., 2005). Thus, since formulation (1)–(6) is based only on arc variables, by relaxing constraints (4) and (5), one obtains a Lagrangian dual objective which is nothing but a MSA problem with reparameterized arc scores. Our Lagrangian approach relies on a subgradient descent where a MSA problem is solved at each iteration. We give more details in the rest of the section.

5.1 Dual Problem

Let Z be the set of the incidence vectors of dependency trees. Keeping tree shape constraints (2), (3) and (6) while dualizing k-bounded block degree constraints (4) and well-nestedness constraints (5), we build the following Lagrangian (Lemaréchal, 2001):

$$\begin{aligned} L(z,u) = &\sum_{a\in A} w_a z_a \\ &+ \sum_{W\in\mathcal{W}^{\geq k}} u_1^W \times (z(\delta(W)) - 2) \\ &+ \sum_{(I_1,I_2)\in\mathcal{I}} u_2^{I_1,I_2} \times (z(\delta(I_1)) + z(\delta(I_2)) - 3) \end{aligned} \tag{7}$$

[4] Based on this remark, we also developed a formulation of this problem with a polynomial number of variables and constraints. However it requires adding many more variables than (Martins et al., 2009). This leads to a formulation which is not tractable, see Section 7.2. Moreover, it cannot be tackled by our Lagrangian Relaxation approach.

with $z \in Z$ and $u = (u_1, u_2) \geq 0$ is a vector of Lagrangian multipliers. We refactor to:

$$L(z, u) = \sum_{a \in A} \theta_a z_a + c \tag{8}$$

where θ are modified scores and c a constant term.

The dual objective is $L^*(u) = \max_z L(z, u)$ with $z \in Z$. Note that computing $L^*(u)$ amounts to solving the MSA problem with modified scores θ and can be efficiently computed. The dual problem is $\min_{u \geq 0} L^*(u)$. L^* is a non-differentiable convex piece-wise linear function and one can find its minimum via subgradient descent. For any vector u, we use the following subgradient. Denote $Mz \leq b$ the set of constraints given by (4) and (5) and $z^* = \arg\max_z L(z, u)$. Let $g = b - Mz^*$ be a subgradient at u, see (Lemaréchal, 2001) for more details. From this subgradient, we compute the descent direction following (Camerini et al., 1975), which aggregates information during the iteration of the subgradient descent algorithm. Unfortunately, optimizing the dual is expensive with so many relaxed constraints. We handle this problem in the next subsection.

5.2 Efficient Optimization with Many Constraints

The *Non Delayed Relax-and-Cut* (NDRC) method (Lucena, 2005) tackles the problem of optimizing a Lagrangian dual problem with exponentially many relaxed constraints. In standard subgradient descent, at each iteration p of the descent, the Lagrangian update can be formulated as:

$$u^{p+1} = (u^p - s^p \times g^p)_+ \tag{9}$$

where $s^p > 0$ is the stepsize[5] and $()_+$ denotes the projection onto $\mathbb{R}^+$, which replaces each negative component by 0. If all Lagrangian multipliers are initialized to 0, the compononent corresponding to a constraint will not be changed until this constraint is violated for the first time. Indeed, by definition of g, we have $[g^p]_i \geq 0$ if constraint i is satisfied at iteration p: the projection on $\mathbb{R}^+$ ensure that $[u^{p+1}]_i$ stays at 0.[6] Thus we do not need to know constraints that have not been violated yet in order to correctly update the Lagrangian multipliers: this is the main intuition behind the NDRC method. However, s^p may depend on the full subgradient information. A common step size (Fisher, 1981) is:

$$s^p = \alpha^p \times \frac{L^*(u^p) - LB^p}{\|g^p\|^2} \tag{10}$$

with α^p a scalar and LB^p the best known lower bound. This is also the case with more recent approaches like AdaGrad (Duchi et al., 2011) and AdaDelta (Zeiler, 2012). As reported in (Beasley, 1993; Lucena, 2006), when dealing with many relaxed constraints, the $\|g^p\|^2$ term can result in each Lagrangian update being almost equal to 0. Therefore, a good practice is to modify the subgradient such that if $[g^p]_i > 0$ and $[u^p]_i = 0$, then we set $[g^p]_i = 0$: this has the same effect on the multipliers as the projection on $\mathbb{R}^+$ in (9), but it prevents the stepsize from becoming too small. Hence, instead of generating a full subgradient at each iteration, which is an expensive operation because we would need to consider all multipliers associated with constraints, we process only a subpart, namely the one associated with constraints that have been violated.

Following (Lucena, 2005), at each iteration p of the subgradient descent we define two sets: *Currently Violated Active Constraints* (CA^p) and *Previously Violated Active Constraints* (PA^p). CA^p and PA^p are not necessarily disjoint. The subgradient is computed only for constraints in $CA^p \cup PA^p$. At each iteration p, we update $PA^p = PA^{p-1} \cup CA^{p-1}$ and a *violation detection step*, similar to the separation step in a cutting plane algorithm, generates CA^p. Two strategies are possible for the detection: (1) adding to CA^p all the constraints violated by the current dual solution; (2) adding only a subset of them. The latter is justified by the fact that many constraints may overlap thus leading to exageration of modified scores on some arcs. We found that strategy (2) gives better convergence results.

Detection for violated block degree constraints (4) can be done with the algorithm described in (Möhl, 2006) in $O(n^2)$. If no violated block degree constraint is found, we search for violated well-nestedness constraints (5) using the $O(n^2)$ algorithm described in (Havelka, 2007).

5.3 Lagrangian Heuristic

We derive a heuristic from the Lagragian Relaxation. First, a dependency tree is computed with

[5] As stated above, instead of the subgradient we follow an improved descent direction which aggregates information of previous iterations. However, this does not change the proposal of this subsection.

[6] $[x]_i$ denotes the ith component of vector x.

the MSA algorithm. If it is valid, it then corresponds to the optimal solution. Otherwise, we proceed as follows. The computation of the step size in (10) in the subgradient descent needs a lower bound which can be given by the score of any valid dependency tree. In our experiments, we compute the best projective spanning arborescence (Eisner, 2000). Each iteration of the subgradient descent computes a spanning arborescence. Since violating (4) and (5) is penalized in the objective function, it tends to produce valid dependency trees. The heuristic returns the highest scoring one.

6 Branch and Bound

Solving the Lagrangian dual problem may not always give an optimal solution to the original problem because of the potential duality gap. Still, we always obtain an upper bound on the optimal solution and if a dual solution satisfies constraints (4) and (5), its score with the original weights provides a lower bound.[7]

Moreover, the subgradient descent algorithm theoretically converges but we have no guarantee that this will happen in a realistic number of iterations. Therefore, in order to retrieve an optimal solution in all cases, we embed the Lagrangian Relaxation of the problem within a Branch-and-Bound procedure (Land and Doig, 1960).

The search space is recursively split according to an arc variable, creating two subspaces, one where it is fixed to 1 and the other to 0 (branching step). The procedure returns a candidate solution when all arc variables are fixed and constraints are satisfied, and the optimal solution is the highest-scoring candidate solution.

For each subspace, we estimate an upper bound using the Lagrangian Relaxation (bounding step). The recursive exploration of a subspace stops (pruning step) if (1) we can prove that all candidate solutions it contains have a score lower than the best found so far, or (2) we detect an unsastifiable constraint.

The branching strategy is built upon Lagrangian multipliers: we branch on the variable z_a with highest value $\theta_a - w_a$. Intuitively, if the branching step sets $z_a = 1$, it indicates that we add a *hard* constraint on an arc which has been strongly promoted by Lagrangian Relaxation. This strategy, compared to other variants, gave the best parsing time on development data.

6.1 Problem Reduction

The efficiency of the Branch-and-Bound procedure crucially depends on the number of free variables. To prune the search space, we rely on problem reduction (Beasley, 1993), once again based on duality and Lagrangian Relaxation, which provides certificates on optimal variable settings.

We fix a variable to 1 (resp. 0), and compute an upper bound on the optimal score with this new constraint. If it is lower than the score of the best solution found so far without this constraint, we can guarantee that this variable cannot (resp. must) be in the optimal solution and safely set it to 0 (resp. 1).

Problem reduction is performed at each node of the Branch-and-Bound tree after computing the upper bound with subgradient descent.

6.2 Fixing Variables to 1

Since a node in V^+ must have exactly one parent, fixing $z_{ij} = 1$ for an arc $a = (i, j)$ greatly reduces the problem size, as it will also fix $z_{hj} = 0, \forall h \neq i$. Among all arc variables that can be set to 1, promising candidates are the arcs in a solution of the unconstrained MSA and the arcs obtained in a solution after the subgradient descent.

There are exactly n such arcs in each set of candidates, so we test fixation for less than $2n$ variables. In this case, we are ready to pay the price of a quadratic computation for each of these arcs.

Hence, for each candidate arc we obtain an upper bound by seeking the (unconstrained) MSA on the graph where this arc is removed. If this upper bound is lower than the score of the best solution found so far, we can safely say that this arc is in the optimal solution.

6.3 Fixing Variables to 0

We could apply the same strategy for fixing variables to 0. However, this reduction is less rewarding and there are many more variables set to 0 than 1 in a MSA solution. Instead, we solve an easier problem, at the expense of a looser upper bound.

For each arc a which is not in the MSA, we compute a maximum directed graph that contains this arc and where all nodes but the root have one parent. Remark that if this graph is connected then it corresponds to a dependency tree. Therefore, the score of this directed graph provides an upper bound on a solution containing arc a. If this upper

[7] Because relaxed constraints are inequalities, constraint satisfaction does not guarantee optimality (Beasley, 1993).

bound is lower than the score of the best solution found so far, we can fix the variable z_a to 0.

Note that this whole fixing procedure can be done in $O(n^2)$.

7 Experiments

We ran a series of experiments to test our method in the case of unlabelled dependency parsing. Our prototype has been developped in Python with some parts in Cython and C++. We use the MSA implementation available in the LEMON library.[8]

7.1 Datasets

We ran experiments on 5 different corpora:

English: Dependencies were extracted from the WSJ part of the Penn Treebank (PTB) with additional NP bracketings (Vadas and Curran, 2007) with the *LTH* converter[9] (default options). Sections 02-21 are used for training, 22 for development and 23 for testing. POS tags were predicted by the Stanford tagger[10] trained with 10-jackkniffing.[11]

German: We used dependencies from the SPMRL dataset (Seddah et al., 2014), with predicted POS tags and the official split. We removed sentences of length greater than 100 in the test set.

Dutch, Spanish and Portuguese: We used the Universal Dependency Treebank 1.2 (Van der Beek et al., 2002; McDonald et al., 2013; Afonso et al., 2002) with gold POS tags and the official split. We removed sentences of length greater than 100 in the test sets.

Those datasets contain different structure distributions as shown in Table 1. Fortunately, our method allows us to easily change the bounded degree constraint or toggle the well-nestedness one. For each language, we decided to use the most constrained combination of bounded block degree constraints and well-nestedness which covers over 99% of the data. Therefore, we chose to enforce 2-BBD and well-nestedness for English and Spanish, 3-BBD and well-nestedness for Dutch and Portuguese and 3-BBD only for German.

7.2 Decoding

In order to compare our methods with previous approaches, we tested five decoding strategies. (`MSA`) computes the best unconstrained dependency tree. (`Eisner`) computes the best projective tree. (`LR`) and (`B&B`) are the heuristic and the exact method presented in Sections 5.3 and 6 respectively.[12] Finally (`MSA/Eisner`) consists in running the MSA algorithm and, if the solution is invalid, returning the (`Eisner`) solution instead.

Our attempt to run the dynamic programming algorithm of (Gómez-Rodríguez et al., 2009) was unsuccessful. Even with heavy pruning we were not able to run it on sentences above 20 words. We also tried to use CPLEX on a compact ILP formulation based on multi-commodity flows (see footnote 4). Parsing time was also prohibitive: a total of 3473 seconds on English data without the well-nestedness constraint, 7298 for German.

We discuss the efficiency of our methods on data for English and German. Other languages give similar results. Optimality rate after the subgradient descent are reported in Figure 2. We see that Lagrangian Relaxation often returns optimal solutions but fails to give a certificate of their optimality. Table 2 shows parsing times. We see that (`LR`) and (`B&B`), while slower than (`MSA`), are fast in the majority of cases, below the third quartile. Inevitably, there are some rare cases where a large portion of the search space is explored, and thus their parsing time is high. Let us remark that these algorithms are run only when (`MSA`) returns an invalid structure, and so total time is very acceptable compared to the baseline.

Finally, we stress the importance of problem reduction as a pre-processing step in `B&B`: after subgradient descent is performed, it removes an average of 83.97% (resp. 76.59%) of arc variables in the English test set (resp. German test set).

7.3 Training

Feature weights are trained using the averaged structured perceptron (Collins, 2002) with 10 iterations where the best iteration is selected on the development set. We used the same feature set as in TurboParser (Martins et al., 2010), including features for lemma. For German, we additionally use morpho-syntactic features.

The decoding algorithm used at training time is the MSA. We experimented with Branch-and-Bound and Lagrangian Relaxation decoding dur-

[8] https://lemon.cs.elte.hu/trac/lemon

[9] http://nlp.cs.lth.se/software/treebank_converter/

[10] http://nlp.stanford.edu/software/tagger.shtml

[11] Prediction precision: 97.40%

[12] In both methods, the subgradient descent is stopped after a fixed maximum number of iterations. We chose 100 for English and 200 for other languages after tuning on the development set.

	English		German		Dutch		Spanish		Portuguese	
	WN	IL	WN	IL	WN	IL	WN	IL	WN	IL
Block degree 1	92.26	-	67.60	-	69.13	-	93.95	-	81.56	0.05
Block degree 2	7.58	0.12	27.12	0.79	28.50	0.08	5.99	0.04	13.92	0.02
Block degree 3	0.12	0.01	3.86	0.30	2.24	0.01	0.02	-	3.76	-
Block degree 4	-	-	0.19	<0.01	0.04	-	-	-	0.54	-
Block degree > 4	-	-	0.11	<0.01	-	-	-	-	0.14	-

Table 1: Distribution of dependency tree characteristics in datasets.

	English (96 sentences)			German (59 sentences)		
	MSA	LR	B&B	MSA	LR	B&B
Mean	0.02	0.26	0.53	0.04	0.51	0.71
Std.	0.01	0.20	0.86	0.02	0.41	1.39
Med.	0.02	0.21	0.27	0.03	0.47	0.47
3rd	0.03	0.34	0.53	0.05	0.71	0.80
Total	1.81	25.09	50.52	2.18	30.19	42.20

Table 2: Timings for strategies (see Section 7.2) on test for solutions which do not satisfy constraints after running MSA. We give (in seconds) average time, standard deviation, median time, time to parse up to the 3rd quartile and total time.

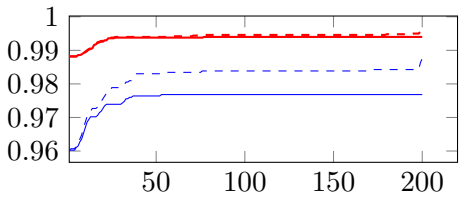


Figure 2: Optimality rate (y-axis) vs number of subgradient iterations (x-axis) for English (thin blue) and German (thick red). Solid line is the optimal rate with certificate, dashed is without.

ing training. It did not significantly improve accuracy and made training and decoding slower.

7.4 Parsing Results

Table 3 shows attachment score (UAS), percentage of valid dependency trees and relative time to (MSA) for different systems for our five decoding strategies. We can see (B&B) is on a par with (LR) on some corpora and more accurate on the others. The former takes more time, and the improvement is correlated with time difference for all corpora but the PTB. Dividing the five corpora in three cases, we can see that:

1. For English and Spanish, where projective dependency trees represent more than 90% of the data, (Eisner) outperforms (MSA). Our methods lie between the two. Here it is better to search for projective trees and (LR) and (B&B) are not interesting in terms of UAS. This is confirmed by the results of (MSA/Eisner).

2. For German and Dutch, where projective dependency trees represent less than 70% of the data, (MSA) outperforms (Eisner). For German, where well-nestedness is not required, our methods are as accurate as (MSA)[13], while for Dutch our methods seem to be useful, as (B&B) outperforms all systems. Moreover, our two methods guarantee validity.

3. For Portuguese, where projective dependency trees represent around 80% of the data, (MSA) is as accurate as (Eisner). In this case we see that, while our heuristic is below, the exact method is more accurate. This seems to be an edge case where neither unconstrained nor projective dependency trees seem to adequately capture the solution space. We also see that it is harder for our methods to give solutions (longer computation times, which tend to indicate that LR cannot guarantee optimality). Our methods are best fitted for this case.

In order to see how much well-nested and bounded block-degree structures are missed by a state-of-the-art parser, we compare our results with TurboParser.[14] We run the parser with three different feature sets: arc-factored, standard (second-order features), and full (third-order features). The results are shown in Table 4. Our model, by enforcing strict compliance to structural rules (100% valid dependency trees), is closer to the empirical distribution than TurboParser in arc-factored mode on all languages but German. Higher-order scoring functions manage to get more similar to the treebank data than our strict thresholds for all languages but Portuguese, at the expense of a significative computational burden.

[13] For German, we notice a small regression which we attribute to the representation of enumerations in the corpus: for enumerations of k elements, k-bounded block-degree subtrees must be generated.

[14] We used 2.1.0 and all defaults but the feature set.

		MSA	Eisner	LR	B&B	MSA/Eisner
English	UAS	89.45	**89.82**	89.54	89.53	89.60
	2-BBD/WN	96.02	–	–	–	–
	Relative Time	1	2.5	1.8	2.5	1.2
German	UAS	**87.79**	86.97	87.78	87.78	87.46
	3-BBD	98.81	–	–	–	–
	Relative Time	1	2.1	1.5	1.7	1.3
Dutch	UAS	77.30	76.62	76.96	**77.40**	76.79
	3-BBD/WN	94.82	–	–	–	–
	Relative Time	1	1.5	1.7	5	1.3
Spanish	UAS	83.37	**83.56**	83.37	83.44	83.48
	2-BBD/WN	92.62	–	–	–	–
	Relative Time	1	2.8	2.7	3	1.5
Portuguese	UAS	83.13	83.14	82.99	**83.21**	82.90
	3-BBD/WN	87.84	–	–	–	–
	Relative Time	1	2.7	5.7	19.7	1.7

Table 3: UAS, percentage of valid structure and decoding time for test data. Time is relative to MSA decoding. The percentage of valid structure is always 100% except for MSA decoding.

	English (99.84)			**German (99.27)**			**Dutch (99.87)**			**Spanish (99.94)**			**Portuguese (99.24)**		
Order	UAS	VDT	RT	UAS	VDT	RT	UAS	VDT	RT	UAS	VDT	RT	UAS	VDT	RT
1st	89.29	94.87	1	87.97	98.74	1	76.10	93.26	1	83.11	93.43	1	83.53	94.79	1
2nd	92.04	99.75	16	89.83	99.28	16	79.05	97.93	18	86.61	98.54	10	87.35	98.96	15
3rd	92.37	99.75	34	90.35	99.24	36	79.68	97.41	37	87.31	99.64	18	88.09	98.98	32

Table 4: UAS, percentage of valid dependency trees (VDT) and relative time (RT) obtained by Turboparser for different score functions on test sets. For each language we give the percentage of valid dependency structures in the data, according to the constraints postulated in Section 7.1.

We interpret this fact as an indication that adding higher order features into our system would make the relaxation method converge more often and faster.

8 Conclusion

We presented a novel characterization of dependency trees complying with two structural rules: bounded block degree and well-nestedness from which we derived two methods for arc-factored dependency parsing. The first one is a heuristic which relies on Lagrangian Relaxation and the Chu-Liu-Edmonds efficient maximum spanning arborescence algorithm. The second one is an exact Branch-and-Bound procedure where bounds are provided by Lagrangian Relaxation. We showed experimentally that these methods give results comparable with state-of-the-art arc-factored parsers, while enforcing constraints in all cases.

In this paper we focused on arc-factor models, but our method could be extended to higher order models, following the dual decomposition method presented in (Koo et al., 2010) in which the maximum-weight spanning arborescence component would be replaced by our constrained model.

Our method opens new perspectives for LTAG parsing, in particular using decomposition techniques where dependencies and templates are predicted separately. Moreover, while well-nested dependencies with 2-bounded block degree can represent LTAG derivations, toggling the well-nestedness or setting the block degree bound allows to express the whole range of derivations in lexicalized LCFRS, whether well-nested or with a bounded fan-out. Our algorithm can exactly represent these settings with a comparable complexity.

Acknowledgements

We thank the anonymous reviewers for their insightful comments which let us significantly improve the submitted paper. This work is supported by a public grant overseen by the French National Research Agency (ANR) as part of the Investissements d'Avenir program (ANR-10-LABX-0083).

References

[Afonso et al.2002] Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. 2002. Floresta sintá (c) tica: A treebank for portuguese. In *LREC*.

[Attardi2006] Giuseppe Attardi. 2006. Experiments with a multilanguage non-projective dependency parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 166–170. Association for Computational Linguistics.

[Ballesteros and Carreras2015] Miguel Ballesteros and Xavier Carreras. 2015. Transition-based spinal

parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 289–299, Beijing, China, July. Association for Computational Linguistics.

[Beasley1993] John E Beasley. 1993. Lagrangian relaxation. In *Modern heuristic techniques for combinatorial problems*, pages 243–303. John Wiley & Sons, Inc.

[Bertsekas1999] Dimitri P Bertsekas. 1999. *Nonlinear programming*. Athena scientific.

[Bodirsky et al.2009] Manuel Bodirsky, Marco Kuhlmann, and Mathias Möhl. 2009. Well-nested drawings as models of syntactic structure. In *Tenth Conference on Formal Grammar and Ninth Meeting on Mathematics of Language*, pages 195–203.

[Camerini et al.1975] Paolo M Camerini, Luigi Fratta, and Francesco Maffioli. 1975. On improving relaxation methods by modified gradient techniques. In *Nondifferentiable optimization*, pages 26–34. Springer.

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

[Collins2002] Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

[Das et al.2012] Dipanjan Das, André FT Martins, and Noah A Smith. 2012. An exact dual decomposition algorithm for shallow semantic parsing with constraints. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 209–217. Association for Computational Linguistics.

[Du et al.2015] Yantao Du, Weiwei Sun, and Xiaojun Wan. 2015. A data-driven, factorization parser for CCG dependency structures. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference of Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1545–1555.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

[Eisner and Satta2000] Jason Eisner and Giorgio Satta. 2000. A faster parsing algorithm for lexicalized tree-adjoining grammars. In *Proceedings of the 5th Workshop on Tree-Adjoining Grammars and Related Formalisms (TAG+ 5)*, pages 14–19.

[Eisner2000] Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In *Advances in probabilistic and other parsing technologies*, pages 29–61. Springer.

[Fernández-González and Martins2015] Daniel Fernández-González and André F. T. Martins. 2015. Parsing as Reduction. In *Annual Meeting of the Association for Computational Linguistics (ACL'15)*, Beijing, China, July.

[Fischetti and Toth1992] Matteo Fischetti and Paolo Toth. 1992. An additive bounding procedure for the asymmetric travelling salesman problem. *Mathematical Programming*.

[Fisher1981] Marshall L Fisher. 1981. The lagrangian relaxation method for solving integer programming problems. *Management science*, 27.

[Gabow and Tarjan1984] Harold N. Gabow and Robert E. Tarjan. 1984. Efficient algorithms for a family of matroid intersection problems. *Journal of Algorithms*, 5(1):80–131.

[Gómez-Rodríguez et al.2009] Carlos Gómez-Rodríguez, David Weir, and John Carroll. 2009. Parsing mildly non-projective dependency structures. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 291–299. Association for Computational Linguistics.

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

[Havelka2007] Jiří Havelka. 2007. Relationship between non-projective edges, their level types, and well-nestedness. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 61–64. Association for Computational Linguistics.

[Joshi and Schabes1997] Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

[Kallmeyer and Kuhlmann2012] Laura Kallmeyer and Marco Kuhlmann. 2012. A formal model for plausible dependencies in lexicalized tree adjoining grammar. In *Proceedings of TAG*, volume 11, pages 108–116.

[Koo and Collins2010] Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*.

[Koo et al.2010] Terry Koo, Alexander M Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298. Association for Computational Linguistics.

[Land and Doig1960] Ailsa H. Land and Alison G. Doig. 1960. An automatic method of solving discrete programming problems. *Econometrica: Journal of the Econometric Society*, 28(3):497–520.

[Lemaréchal2001] Claude Lemaréchal. 2001. Lagrangian relaxation. In *Computational combinatorial optimization*, pages 112–156. Springer.

[Lucena2005] Abilio Lucena. 2005. Non delayed relax-and-cut algorithms. *Annals of Operations Research*, 140(1):375–410.

[Lucena2006] Abilio Lucena. 2006. Lagrangian relax-and-cut algorithms. In *Handbook of Optimization in Telecommunications*, pages 129–145. Springer.

[Martins et al.2009] André FT Martins, Noah A Smith, and Eric P Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 342–350. Association for Computational Linguistics.

[Martins et al.2010] André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 34–44.

[McDonald and Pereira2006] Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*.

[McDonald et al.2005] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 523–530. Association for Computational Linguistics.

[McDonald et al.2013] Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97. Citeseer.

[Möhl2006] Mathias Möhl. 2006. *Drawings as models of syntactic structure: Theory and algorithms*. Ph.D. thesis, Saarland University.

[Nivre2003] Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*.

[Pitler and McDonald2015] Emily Pitler and Ryan McDonald. 2015. A linear-time transition system for crossing interval trees. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 662—-671.

[Pitler et al.2012] Emily Pitler, Sampath Kannan, and Mitchell Marcus. 2012. Dynamic programming for higher order parsing of gap-minding trees. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 478–488. Association for Computational Linguistics.

[Qian and Liu2013] Xian Qian and Yang Liu. 2013. Branch and bound algorithm for dependency parsing with non-local features. *Transactions of the Association for Computational Linguistics*, 1:37–48.

[Riedel et al.2012] Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut: delayed column and row generation for graph based parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 732–743. Association for Computational Linguistics.

[Riedel2009] Sebastian Riedel. 2009. Cutting plane map inference for markov logic. In *SRL 2009*.

[Rush and Collins2012] Alexander M Rush and Michael Collins. 2012. A tutorial on dual decomposition and lagrangian relaxation for inference in natural language processing. *Journal of Artificial Intelligence Research*.

[Rush et al.2013] Alexander M. Rush, Yin-Wen Chang, and Michael Collins. 2013. Optimal beam search for machine translation. In *Proceedings of EMNLP*.

[Satta and Kuhlmann2014] Giorgio Satta and Marco Kuhlmann. 2014. Efficient parsing for head-split dependency trees. *Transactions of the Association for Computational Linguistics*, 1:267–278.

[Schrijver2003] A. Schrijver. 2003. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer.

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

[Shen and Joshi2007] Libin Shen and Aravind K Joshi. 2007. Bidirectional ltag dependency parsing. Technical report, Technical Report 07-02, IRCS, University of Pennsylvania.

[Vadas and Curran2007] David Vadas and James R. Curran. 2007. Adding noun phrase structure to the penn treebank. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 240–247.

[Van der Beek et al.2002] Leonoor Van der Beek, Gosse Bouma, Rob Malouf, and Gertjan Van Noord. 2002. The alpino dependency treebank. *Language and Computers*, 45(1):8–22.

[Zeiler2012] Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Query Expansion with Locally-Trained Word Embeddings

Fernando Diaz
Microsoft
fdiaz@microsoft.com

Bhaskar Mitra
Microsoft
bmitra@microsoft.com

Nick Craswell
Microsoft
nickr@microsoft.com

Abstract

Continuous space word embeddings have received a great deal of attention in the natural language processing and machine learning communities for their ability to model term similarity and other relationships. We study the use of term relatedness in the context of query expansion for *ad hoc* information retrieval. We demonstrate that word embeddings such as word2vec and GloVe, when trained globally, underperform corpus and query specific embeddings for retrieval tasks. These results suggest that other tasks benefiting from global embeddings may also benefit from local embeddings.

1 Introduction

Continuous space embeddings such as word2vec (Mikolov et al., 2013b) or GloVe (Pennington et al., 2014a) project terms in a vocabulary to a dense, lower dimensional space. Recent results in the natural language processing community demonstrate the effectiveness of these methods for analogy and word similarity tasks. In general, these approaches provide global representations of words; each word has a fixed representation, regardless of any discourse context. While a global representation provides some advantages, language use can vary dramatically by topic. For example, ambiguous terms can easily be disambiguated given local information in immediately surrounding words (Harris, 1954; Yarowsky, 1993). The window-based training of word2vec style algorithms exploits this distributional property.

A global word embedding, even when trained using local windows, risks capturing only coarse representations of those topics dominant in the corpus. While a particular embedding may be appropriate for a specific word within a sentence-length context globally, it may be entirely inappropriate within a specific topic. Gale *et al.* refer to this as the 'one sense per discourse' property (Gale et al., 1992). Previous work by Yarowsky demonstrates that this property can be successfully combined with information from nearby terms for word sense disambiguation (Yarowsky, 1995). Our work extends this approach to word2vec-style training in the context word similarity.

For many tasks that require topic-specific linguistic analysis, we argue that topic-specific representations should outperform global representations. Indeed, it is difficult to imagine a natural language processing task that would *not* benefit from an understanding of the local topical structure. Our work focuses on a query expansion, an information retrieval task where we can study different lexical similarity methods with an extrinsic evaluation metric (i.e. retrieval metrics). Recent work has demonstrated that similarity based on global word embeddings can be used to outperform classic pseudo-relevance feedback techniques (Sordoni et al., 2014; al Masri et al., 2016).

We propose that embeddings be learned on topically-constrained corpora, instead of large topically-unconstrained corpora. In a retrieval scenario, this amounts to retraining an embedding on documents related to the topic of the query. We present local embeddings which capture the nuances of topic-specific language better than global embeddings. There is substantial evidence that global methods underperform local methods for information re-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 367–377,
Berlin, Germany, August 7-12, 2016.

trieval tasks such as query expansion (Xu and Croft, 1996), latent semantic analysis (Hull, 1994; Schütze et al., 1995; Singhal et al., 1997), cluster-based retrieval (Tombros and van Rijsbergen, 2001; Tombros et al., 2002; Willett, 1985), and term clustering (Attar and Fraenkel, 1977). We demonstrate that the same holds true when using word embeddings for text retrieval.

2 Motivation

For the purpose of motivating our approach, we will restrict ourselves to word2vec although other methods behave similarly (Levy and Goldberg, 2014). These algorithms involve discriminatively training a neural network to predict a word given small set of context words. More formally, given a target word w and observed context c, the instance loss is defined as,

$$\ell(w,c) = \log \sigma(\phi(w) \cdot \psi(c)) + \eta \cdot \mathbb{E}_{\overline{w} \sim \theta_C} \left[\log \sigma(-\phi(w) \cdot \psi(\overline{w}))\right]$$

where $\phi : \mathcal{V} \rightarrow \Re^k$ projects a term into a k-dimensional embedding space, $\psi : \mathcal{V}^m \rightarrow \Re^k$ projects a set of m terms into a k-dimensional embedding space, and $\overline{w}$ is a randomly sampled 'negative' context. The parameter η controls the sampling of random negative terms. These matrices are estimated over a set of contexts sampled from a large corpus and minimize the expected loss,

$$\mathcal{L}_c = \mathbb{E}_{w,c \sim p_c} \left[\ell(w,c)\right] \quad (1)$$

where p_c is the distribution of word-context pairs in the training corpus and can be estimated from corpus statistics.

While using corpus statistics may make sense absent any other information, oftentimes we know that our analysis will be topically constrained. For example, we might be analyzing the 'sports' documents in a collection. The language in this domain is more specialized and the distribution over word-context pairs is unlikely to be similar to $p_c(w,c)$. In fact, prior work in information retrieval suggests that documents on subtopics in a collection have very different unigram distributions compared to the whole corpus (Cronen-Townsend et al., 2002). Let $p_t(w,c)$ be the probability

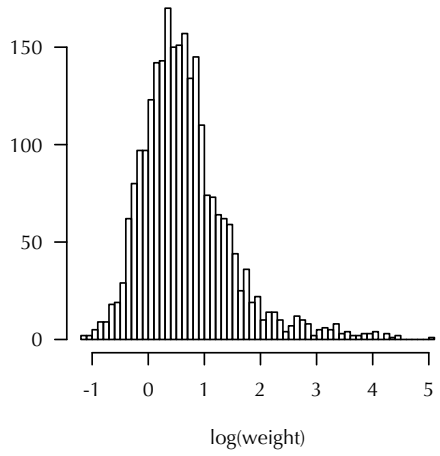


Figure 1: Importance weights for terms occurring in documents related to 'argentina pegging dollar' relative to frequency in gigaword.

of observing a word-context pair conditioned on the topic t. The expected loss under this distribution is (Shimodaira, 2000),

$$\mathcal{L}_t = \mathbb{E}_{w,c \sim p_c} \left[\frac{p_t(w,c)}{p_c(w,c)} \ell(w,c)\right] \quad (2)$$

In general, if our corpus consists of sufficiently diverse data (e.g. Wikipedia), the support of $p_t(w,c)$ is much smaller than and contained in that of $p_c(w,c)$. The loss, ℓ, of a context that occurs more frequently in the topic, will be amplified by the importance weight $\omega = \frac{p_t(w,c)}{p_c(w,c)}$. Because topics require specialized language, this is likely to occur; at the same time, these contexts are likely to be underemphasized in training a model according to Equation 1.

In order to quantify this, we took a topic from a TREC *ad hoc* retrieval collection (see Section 5 for details) and computed the importance weight for each term occurring in the set of on-topic documents. The histogram of weights ω is presented in Figure 1. While larger probabilities are expected since the size of a topic-constrained vocabulary is smaller, there are a non-trivial number of terms with much larger importance weights. If the loss, $\ell(w)$, of a word2vec embedding is worse for these words with low $p_c(w)$, then we expect these errors to be exacerbated for the topic.

Of course, these highly weighted terms may have a low value for $p_t(w)$ but a very high value relative to the corpus. We can adjust the

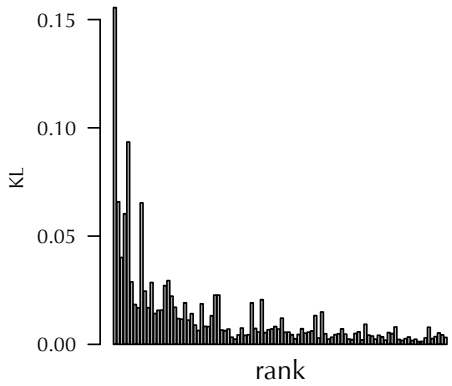


Figure 2: Pointwise Kullback-Leibler divergence for terms occurring in documents related to 'argentina pegging dollar' relative to frequency in gigaword.

global	local
cutting	tax
squeeze	deficit
reduce	vote
slash	budget
reduction	reduction
spend	house
lower	bill
halve	plan
soften	spend
freeze	billion

Figure 3: Terms similar to 'cut' for a word2vec model trained on a general news corpus and another trained only on documents related to 'gasoline tax'.

weights by considering the pointwise Kullback-Leibler divergence for each word w,

$$D_w(p_t \| p_c) = p_t(w) \log \frac{p_t(w)}{p_c(w)} \quad (3)$$

Words which have a much higher value of $p_t(w)$ than $p_c(w)$ *and* have a high absolute value of $p_t(w)$ will have high pointwise KL divergence. Figure 2 shows the divergences for the top 100 most frequent terms in $p_t(w)$. The higher ranked terms (i.e. good query expansion candidates) tend to have much higher probabilities than found in $p_c(w)$. If the loss on those words is large, this may result in poor embeddings for the most important words for the topic.

A dramatic change in distribution between the corpus and the topic has implications for performance precisely because of the objective used by word2vec (i.e. Equation 1). The training emphasizes word-context pairs occurring with high frequency in the corpus. We will demonstrate that, even with heuristic downsampling of frequent terms in word2vec, these techniques result in inferior performance for specific topics.

Thus far, we have sketched out why using the corpus distribution for a specific topic may result in undesirable outcomes. However, it is even unclear that $p_t(w|c) = p_c(w|c)$. In fact, we suspect that $p_t(w|c) \neq p_c(w|c)$ because of the 'one sense per discourse' claim (Gale et al., 1992). We can qualitatively observe the difference in $p_c(w|c)$ and $p_t(w|c)$ by training two word2vec models: the first on the large, generic Gigaword corpus and the second on a topically-constrained subset of the gigaword. We present the most similar terms to 'cut' using both a global embedding and a topic-specific embedding in Figure 3. In this case, the topic is 'gasoline tax'. As we can see, the 'tax cut' sense of 'cut' is emphasized in the topic-specific embedding.

3 Local Word Embeddings

The previous section described several reasons why a global embedding may result in over-general word embeddings. In order to perform topic-specific training, we need a set of topic-specific documents. In information retrieval scenarios users rarely provide the system with examples of topic-specific documents, instead providing a small set of keywords.

Fortunately, we can use information retrieval techniques to generate a query-specific set of topical documents. Specifically, we adopt a language modeling approach to do so (Croft and Lafferty, 2003). In this retrieval model, each document is represented as a maximum likelihood language model estimated from document term frequencies. Query language models are estimated similarly, using term frequency in the query. A document score then, is the Kullback-Leibler divergence between the query and document language

models,

$$D(p_q \| p_d) = \sum_{w \in \mathcal{V}} p_q(w) \log \frac{p_q(w)}{p_d(w)} \quad (4)$$

Documents whose language models are more similar to the query language model will have a *lower* KL divergence score. For consistency with prior work, we will refer to this as the query likelihood score of a document.

The scores in Equation 4 can be passed through a softmax function to derive a multinomial over the entire corpus (Lavrenko and Croft, 2001),

$$p(d) = \frac{\exp(-D(p_q \| p_d))}{\sum_{d'} \exp(-D(p_q \| p_{d'}))} \quad (5)$$

Recall in Section 2 that training a word2vec model weights word-context pairs according to the corpus frequency. Our query-based multinomial, $p(d)$, provides a weighting function capturing the documents relevant to this topic. Although an estimation of the topic-specific documents from a query will be imprecise (i.e. some nonrelevant documents will be scored highly), the language use tends to be consistent with that found in the known relevant documents.

We can train a local word embedding using an arbitrary optimization method by sampling documents from $p(d)$ instead of uniformly from the corpus. In this work, we use word2vec, although any method that operates on a sample of documents can be used.

4 Query Expansion with Word Embeddings

When using language models for retrieval, query expansion involves estimating an alternative to p_q. Specifically, when each expansion term is associated with a weight, we normalize these weights to derive the expansion language model, p_{q^+}. This language model is then interpolated with the original query model,

$$p_q^1(w) = \lambda p_q(w) + (1 - \lambda) p_{q^+}(w) \quad (6)$$

This interpolated language model can then be used with Equation 4 to rank documents (Abdul-Jaleel et al., 2004). We will refer to this as the expanded query score of a document.

Now we turn to using word embeddings for query expansion. Let $\mathbf{U}$ be an $|\mathcal{V}| \times k$ term embedding matrix. If $\mathbf{q}$ is a $|\mathcal{V}| \times 1$ column term vector for a query, then the expansion term weights are $\mathbf{U}\mathbf{U}^\top\mathbf{q}$. We then take the top k terms, normalize their weights, and compute $p_{q^+}(w)$.

We consider the following alternatives for $\mathbf{U}$. The first approach is to use a global model trained by sampling documents uniformly. The second approach, which we propose in this paper, is to use a local model trained by sampling documents from $p(d)$.

5 Methods

5.1 Data

To evaluate the different retrieval strategies described in Section 3, we use the following datasets. Two newswire datasets, trec12 and robust, consist of the newswire documents and associated queries from TREC *ad hoc* retrieval evaluations. The trec12 corpus consists of Tipster disks 1 and 2; and the robust corpus consists of Tipster disks 4 and 5. Our third dataset, web, consists of the ClueWeb 2009 Category B Web corpus. For the Web corpus, we only retain documents with a Waterloo spam rank above 70.[1] We present corpus statistics in Table 1.

We consider several publicly available global embeddings. We use four GloVe embeddings of different dimensionality trained on the union of Wikipedia and Gigaword documents.[2] We use one publicly available word2vec embedding trained on Google News documents.[3] We also trained a global embedding for trec12 and robust using the entire corpus. Instead of training a global embedding on the large web collection, we use a GloVe embedding trained on Common Crawl data.[4]

We train local embeddings with word2vec using one of three retrieval sources. First, we consider documents retrieved from the target corpus of the query (i.e. trec12, robust, or web). We also consider training a local embed-

[1] https://plg.uwaterloo.ca/~gvcormac/clueweb09spam/

[2] http://nlp.stanford.edu/data/glove.6B.zip

[3] https://code.google.com/archive/p/word2vec/

[4] http://nlp.stanford.edu/data/glove.840B.300d.zip

	docs	words	queries
trec12	469,949	438,338	150
robust	528,155	665,128	250
web	50,220,423	90,411,624	200
news	9,875,524	2,645,367	-
wiki	3,225,743	4,726,862	-

Table 1: Corpora used for retrieval and local embedding training.

ding by performing a retrieval on large auxiliary corpora. We use the Gigaword corpus as a large auxiliary news corpus. We hypothesize that retrieving from a larger news corpus will provide substantially more local training data than a target retrieval. We also use a Wikipedia snapshot from December 2014. We hypothesize that retrieving from a large, high fidelity corpus will provide cleaner language than that found in lower fidelity target domains such as the web. Table 1 shows the relative magnitude of these auxiliary corpora compared to the target corpora.

All corpora in Table 1 were stopped using the SMART stopword list[5] and stemmed using the Krovetz algorithm (Krovetz, 1993). We used the Indri implementation for indexing and retrieval.[6]

5.2 Evaluation

We consider several standard retrieval evaluation metrics, including NDCG@10 and interpolated precision at standard recall points (Järvelin and Kekäläinen, 2002; van Rijsbergen, 1979). NDCG@10 provides insight into performance specifically at higher ranks. An interpolated precision recall graph describes system performance throughout the entire ranked list.

5.3 Training

All retrieval experiments were conducted by performing 10-fold cross-validation across queries. Specifically, we cross-validate the number of expansion terms, $k \in \{5, 10, 25, 50, 100, 250, 500\}$, and interpolation weight, $\lambda \in [0, 1]$. For local word2vec training, we cross-validate the learning rate $\alpha \in \{10^{-1}, 10^{-2}, 10^{-3}\}$.

[5] http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

[6] http://www.lemurproject.org/indri/

All word2vec training used the publicly available word2vec cbow implementation.[7] When training the local models, we sampled 1000 documents from $p(d)$ with replacement. To compensate for the much smaller corpus size, we ran word2vec training for 80 iterations. Local word2vec models use a fixed embedding dimension of 400 although other choices did not significantly affect our results. Unless otherwise noted, default parameter settings were used.

In our experiments, expanded queries rescore the top 1000 documents from an initial query likelihood retrieval. Previous results have demonstrated that this approach results in performance nearly identical with an expanded retrieval at a much lower cost (Diaz, 2015). Because publicly available embeddings may have tokenization inconsistent with our target corpora, we restricted the vocabulary of candidate expansion terms to those occurring in the initial retrieval. If a candidate term was not found in the vocabulary of the embedding matrix, we searched for the candidate in a stemmed version of the embedding vocabulary. In the event that the candidate term was still not found after this process, we removed it from consideration.

6 Results

We present results for retrieval experiments in Table 2. We find that embedding-based query expansion outperforms our query likelihood baseline across all conditions. When using the global embedding, the news corpora benefit from the various embeddings in different situations. Interestingly, for trec12, using an embedding trained on the target corpus significantly outperforms all other global embeddings, despite using substantially less data to estimate the model. While this performance may be due to the embedding having a tokenization consistent with the target corpus, it may also come from the fact that the corpus is more representative of the target documents than other embeddings which rely on online news or are mixed with non-news content. To some extent this supports our desire to move training closer to the target distribution.

Across all conditions, local embeddings sig-

[7] https://code.google.com/p/word2vec/

Table 2: Retrieval results comparing query expansion based on various global and local embeddings. Bolded numbers indicate the best expansion in that class of embeddings. Wilcoxon signed rank test between bolded numbers indicates statistically significant improvements ($p < 0.05$) for all collections.

		global						local		
		wiki+giga				gnews	target	target	giga	wiki
	QL	50	100	200	300	300	400	400	400	400
trec12	0.514	0.518	0.518	0.530	0.531	0.530	**0.545**	0.535	**0.563***	0.523
robust	0.467	0.470	0.463	0.469	0.468	**0.472**	0.465	0.475	**0.517***	0.476
web	0.216	0.227	0.229	0.230	**0.232**	0.218	0.216	0.234	0.236	**0.258***

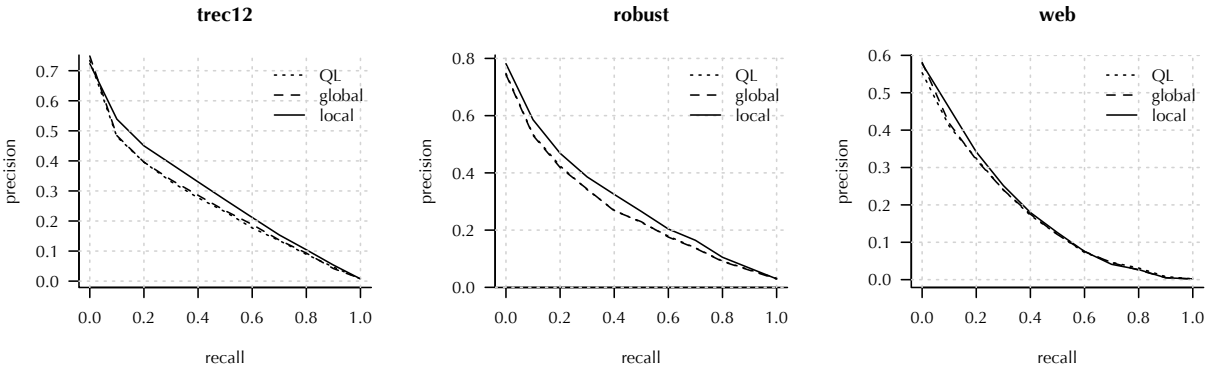


Figure 4: Interpolated precision-recall curves for query likelihood, the best global embedding, and the best local embedding from Table 2.

nificantly outperform global embeddings for query expansion. For our two news collections, estimating the local model using a retrieval from the larger Gigaword corpus led to substantial improvements. This effect is almost certainly due to the Gigaword corpus being similar in writing style to the target corpus but, at the same time, providing significantly more relevant content (Diaz and Metzler, 2006). As a result, the local embedding is trained using a larger variety of topical material than if it were to use a retrieval from the smaller target corpus. An embedding trained with a retrieval from Wikipedia tended to perform worse most likely because the language is dissimilar from news content. Our web collection, on the other hand, benefitted more from embeddings trained using retrievals from the general Wikipedia corpus. The Gigaword corpus was less useful here because news-style language is almost certainly not representative of general web documents.

Figure 4 presents interpolated precision-recall curves comparing the baseline, the best global query expansion method, and the best local query expansion method. Interestingly, although global methods achieve strong performance for NDCG@10, these improvements over the baseline are not reflected in our precision-recall curves. Local methods, on the other hand, almost always strictly dominate both the baseline and global expansion across all recall levels.

The results support the hypothesis that local embeddings provide better similarity measures than global embeddings for query expansion. In order to understand why, we first compare the performance differences between local and global embeddings. Figure 2 suggests that we should adopt a local embedding when the local unigram language model deviates from the corpus language model. To test this, we computed the KL divergence between the local unigram distribution, $\sum_d p(w|d)p(d)$, and the corpus unigram language model (Cronen-Townsend et al., 2002). We hypothesize that, when this value is high, the topic language is different from the corpus language and the

Table 3: Kendall's τ and Spearman's ρ between improvement in NDCG@10 and local KL divergence with the corpus language model. The improvement is measured for the best local embedding over the best global embedding.

	τ	ρ
trec12	0.0585	0.0798
robust	0.0545	0.0792
web	0.0204	0.0283

global embedding will be inferior to the local embedding. We tested the rank correlation between this KL divergence and the relative performance of the local embedding with respect to the global embedding. These correlations are presented in Table 3. Unfortunately, we find that the correlation is low, although it is positive across collections.

We can also qualitatively analyze the differences in the behavior of the embeddings. If we have access to the set of documents labeled relevant to a query, then we can compute the frequency of terms in this set and consider those terms with high frequency (after stopping and stemming) to be good query expansion candidates. We can then visualize where these terms lie in the global and local embeddings. In Figure 5, we present a two-dimensional projection (van der Maaten and Hinton, 2008) of terms for the query 'ocean remote sensing', with those good candidates highlighted. Our projection includes the top 50 candidates by frequency and a sample of terms occurring in the query likelihood retrieval. We notice that, in the global embedding, the good candidates are spread out amongst poorer candidates. By contrast, the local embedding clusters the candidates in general but also situates them closely around the query. As a result, we suspect that the similar terms extracted from the local embedding are more likely to include these good candidates.

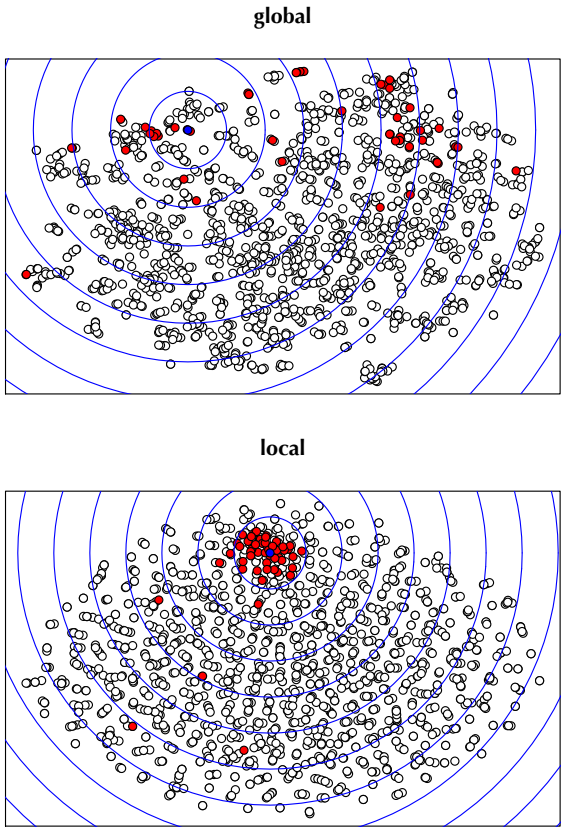


Figure 5: Global versus local embedding of highly relevant terms. Each point represents a candidate expansion term. Red points have high frequency in the relevant set of documents. White points have low or no frequency in the relevant set of documents. The blue point represents the query. Contours indicate distance from the query.

7 Discussion

The success of local embeddings on this task should alarm natural language processing researchers using global embeddings as a representational tool. For one, the approach of learning from vast amounts of data is only effective if the data is appropriate for the task at hand. And, when provided, much smaller high-quality data can provide much better performance. Beyond this, our results suggest that the approach of estimating global representations, while computationally convenient, may overlook insights possible at query time, or evaluation time in general. A similar local embedding approach can be adopted for any natural language processing task where topical locality is expected and can be estimated. Although we used a query to re-weight the corpus in our experiments, we could just as easily use alternative contextual information (e.g. a sentence, paragraph, or document) in other tasks.

Despite these strong results, we believe that

there are still some open questions in this work. First, although local embeddings provide effectiveness gains, they can be quite inefficient compared to global embeddings. We believe that there is opportunity to improve the efficiency by considering offline computation of local embeddings at a coarser level than queries but more specialized than the corpus. If the retrieval algorithm is able to select the appropriate embedding at query time, we can avoid training the local embedding. Second, although our supporting experiments (Table 3, Figure 5) add some insight into our intuition, the results are not strong enough to provide a solid explanation. Further theoretical and empirical analysis is necessary.

8 Related Work

Topical adaptation of models The shortcomings of learning a single global vector representation, especially for polysemic words, have been pointed out before (Reisinger and Mooney, 2010b). The problem can be addressed by training a global model with multiple vector embeddings per word (Reisinger and Mooney, 2010a; Huang et al., 2012) or topic-specific embeddings (Liu et al., 2015). The number of senses for each word may be fixed (Neelakantan et al., 2015), or determined using class labels (Trask et al., 2015). However, to the best of our knowledge, this is the first time that training topic-specific word embeddings has been explored.

Several methods exist in the language modeling community for topic-dependent adaptation of language models (Bellegarda, 2004). These can lead to performance improvements in tasks such as machine translation (Zhao et al., 2004) and speech recognition (Nanjo and Kawahara, 2004). Topic-specific data may be gathered in advance, by identifying corpus of topic-specific documents. It may also be gathered during the discourse, using multiple hypotheses from N-best lists as a source of topic-specific language. Then a topic-specific language model is trained (or the global model is adapted) online using the topic-specific training data. A topic-dependent model may be combined with the global model using linear interpolation (Iyer and Ostendorf, 1999) or other more sophisticated approaches (Federico, 1996; Kuhn and De Mori, 1990). Similarly to the adaptation work, we use topic-specific documents to train a topic-specific model. In our case the documents come from a first round of retrieval for the user's current query, and the word embedding model is trained based on sentences from the topic-specific document set. Unlike the past work, we do not focus on interpolating the local and global models, although this is a promising area for future work. In the current study we focus on a direct comparison between the local-only and global-only approach, for improving retrieval performance.

Word embeddings for IR Information Retrieval has a long history of learning representations of words that are low-dimensional dense vectors. These approaches can be broadly classified into two families based on whether they are learnt based on a *term-document* matrix or *term co-occurence* data. Using the *term-document* matrix for embedding leads to several well-studied approaches such as LSA (Deerwester et al., 1990), PLSA (Hofmann, 1999), and LDA (Blei et al., 2003; Wei and Croft, 2006). The performance of these models varies depending on the task, for example they are known to perform poorly for retrieval tasks unless combined with lexical features (Atreya and Elkan, 2011a). *Term-cooccurence* based embeddings, such as word2vec (Mikolov et al., 2013b; Mikolov et al., 2013a) and (Pennington et al., 2014b), have recently been remarkably popular for many natural language processing and logical reasoning tasks. However, there are relatively less known successful applications of these models in IR. Ganguly *et. al.* (Ganguly et al., 2015) used the word similarity in the word2vec embedding space as a way to estimate term transformation probabilities in a language modelling setting for retrieval. More recently, Nalisnick *et. al.* (Nalisnick et al., 2016) proposed to model document about-ness by computing the similarity between all pairs of query and document terms using dual embedding spaces. Both these approaches estimate the semantic relatedness between two terms as the cosine distance between them in the embedding space(s). We adopt a similar notion of term relatedness but focus on demon-

strating improved retrieval performance using locally trained embeddings.

Local latent semantic analysis Despite the mathematical appeal of latent semantic analysis, several experiments suggest that its empirical performance may be no better than that of ranking using standard term vectors (Deerwester et al., 1990; Dumais, 1995; Atreya and Elkan, 2011b). In order to address the coarseness of corpus-level latent semantic analysis, Hull proposed restricting analysis to the documents relevant to a query (Hull, 1994). This approach significantly improved over corpus-level analysis for routing tasks, a result that has been reproduced in consequent research (Schütze et al., 1995; Singhal et al., 1997). Our work can be seen as an extension of these results to more recent techniques such as word2vec.

9 Conclusion

We have demonstrated a simple and effective method for performing query expansion with word embeddings. Importantly, our results highlight the value of locally-training word embeddings in a query-specific manner. The strength of these results suggests that other research adopting global embedding vectors should consider local embeddings as a potentially superior representation. Instead of using a "Sriracha sauce of deep learning," as embedding techniques like word2vec have been called, we contend that the situation sometimes requires, say, that we make a béchamel or a mole verde or a sambal—or otherwise learn to cook.

References

Nasreen Abdul-Jaleel, James Allan, W. Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Donald Metzler, Mark D. Smucker, Trevor Strohman, Howard Turtle, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard. In *Online Proceedings of 2004 Text REtrieval Conference*.

Mohannad al Masri, Catherine Berrut, and Jean-Pierre Chevallet. 2016. A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information. In Nicola Ferro, Fabio Crestani, Marie-Francine Moens, Josiane Mothe, Fabrizio Silvestri, Maria Giorgio Di Nunzio, Claudia Hauff, and Gianmaria Silvello, editors, *Proceedings of the 38th European Conference on IR Research (ECIR 2016)*, pages 709–715, Cham. Springer International Publishing.

Avinash Atreya and Charles Elkan. 2011a. Latent semantic indexing (lsi) fails for trec collections. *ACM SIGKDD Explorations Newsletter*, 12(2):5–10.

Avinash Atreya and Charles Elkan. 2011b. Latent semantic indexing (lsi) fails for trec collections. *SIGKDD Explor. Newsl.*, 12(2):5–10, March.

R. Attar and A. S. Fraenkel. 1977. Local feedback in full-text retrieval systems. *J. ACM*, 24(3):397–417, July.

Jerome R Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech communication*, 42(1):93–108.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

W. Bruce Croft and John Lafferty. 2003. *Language Modeling for Information Retrieval*. Kluwer Academic Publishing.

Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2002. Predicting query performance. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306, New York, NY, USA. ACM Press.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.

Fernando Diaz and Donald Metzler. 2006. Improving the estimation of relevance models using large external corpora. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161, New York, NY, USA. ACM Press.

Fernando Diaz. 2015. Condensed list relevance models. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 313–316, New York, NY, USA, May. ACM.

Susan T. Dumais. 1995. Latent semantic indexing (LSI): TREC-3 report. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 219–230.

Marcello Federico. 1996. Bayesian estimation methods for n-gram language model adaptation. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 1, pages 240–243. IEEE.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 233–237, Stroudsburg, PA, USA. Association for Computational Linguistics.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J.F. Jones. 2015. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 795–798, New York, NY, USA. ACM.

Zellig S. Harris. 1954. Distributional structure. *WORD*, 10(2-3):146–162.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, New York, NY, USA. ACM Press.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

David Hull. 1994. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '94, pages 282–291, New York, NY, USA. Springer-Verlag New York, Inc.

R.M. Iyer and M. Ostendorf. 1999. Modeling long distance dependence in language: topic mixtures versus dynamic cache models. *Speech and Audio Processing, IEEE Transactions on*, 7(1):30–39, Jan.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *TOIS*, 20(4):422–446.

Robert Krovetz. 1993. Viewing morphology as an inference process. In *SIGIR '93: Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–202, New York, NY, USA. ACM Press.

Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12(6):570–583.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM Press.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2418–2424. AAAI Press.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Eric Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. 2016. Improving document ranking with dual word embeddings. In *Proc. WWW. International World Wide Web Conferences Steering Committee*.

Hiroaki Nanjo and Tatsuya Kawahara. 2004. Language model and speaking rate adaptation for spontaneous presentation speech recognition. *Speech and Audio Processing, IEEE Transactions on*, 12(4):391–400.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2015. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014a. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014b. Glove: Global vectors for word representation. *Proc. EMNLP*, 12:1532–1543.

Joseph Reisinger and Raymond Mooney. 2010a. A mixture model with sharing for lexical semantics. In *Proceedings of the 2010 Conference*

on Empirical Methods in Natural Language Processing, pages 1173–1182. Association for Computational Linguistics.

Joseph Reisinger and Raymond J Mooney. 2010b. Multi-prototype vector-space models of word meaning. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 109–117. Association for Computational Linguistics.

Hinrich Schütze, David A. Hull, and Jan O. Pedersen. 1995. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, pages 229–237, New York, NY, USA. ACM.

Hidetoshi Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227 – 244.

Amit Singhal, Mandar Mitra, and Chris Buckley. 1997. Learning routing queries in a query zone. *SIGIR Forum*, 31(SI):25–32, July.

Alessandro Sordoni, Yoshua Bengio, and Jian-Yun Nie. 2014. Learning concept embeddings for query expansion by quantum entropy minimization. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1586–1592. AAAI Press.

Anastasios Tombros and C. J. van Rijsbergen. 2001. Query-sensitive similarity measures for the calculation of interdocument relationships. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 17–24, New York, NY, USA. ACM Press.

Anastasios Tombros, Robert Villa, and C. J. Van Rijsbergen. 2002. The effectiveness of query-specific hierarchic clustering in information retrieval. *Inf. Process. Manage.*, 38(4):559–582, July.

Andrew Trask, Phil Michalak, and John Liu. 2015. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388*.

Laurens van der Maaten and Geoffrey E. Hinton. 2008. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605.

C. J. van Rijsbergen. 1979. *Information Retrieval*. Butterworths.

Xing Wei and W. Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 178–185, New York, NY, USA. ACM Press.

Peter Willett. 1985. Query-specific automatic document classification. In *International Forum on Information and Documentation*, volume 10, pages 28–32.

Jinxi Xu and W. Bruce Croft. 1996. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 4–11, New York, NY, USA. ACM.

David Yarowsky. 1993. One sense per collocation. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 266–271, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics*, ACL '95, pages 189–196, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bing Zhao, Matthias Eck, and Stephan Vogel. 2004. Language model adaptation for statistical machine translation with structured query models. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Together We Stand: Siamese Networks for Similar Question Retrieval

Arpita Das[1] Harish Yenala[1] Manoj Chinnakotla[2,1] Manish Shrivastava[1]

[1]IIIT Hyderabad, Hyderabad, India
{arpita.das,harish.yenala}@research.iiit.ac.in
m.shrivastava@iiit.ac.in

[2]Microsoft, Hyderabad, India
manojc@microsoft.com

Abstract

Community Question Answering (cQA) services like Yahoo! Answers[1], Baidu Zhidao[2], Quora[3], StackOverflow[4] *etc.* provide a platform for interaction with experts and help users to obtain precise and accurate answers to their questions. The time lag between the user posting a question and receiving its answer could be reduced by retrieving similar historic questions from the cQA archives. The main challenge in this task is the "lexico-syntactic" gap between the current and the previous questions. In this paper, we propose a novel approach called *"Siamese Convolutional Neural Network for cQA (SCQA)"* to find the semantic similarity between the current and the archived questions. SCQA consist of twin convolutional neural networks with shared parameters and a contrastive loss function joining them.

SCQA learns the similarity metric for question-question pairs by leveraging the question-answer pairs available in cQA forum archives. The model projects semantically similar question pairs nearer to each other and dissimilar question pairs farther away from each other in the semantic space. Experiments on large scale real-life "Yahoo! Answers" dataset reveals that SCQA outperforms current state-of-the-art approaches based on translation models, topic models and deep neural network based models which use non-shared parameters.

1 Introduction

The cQA forums have emerged as popular and effective means of information exchange on the Web. Users post queries in these forums and receive precise and compact answers in stead of a list of documents. Unlike in Web search, opinion based queries are also answered here by experts and users based on their personal experiences. The question and answers are also enhanced with rich metadata like categories, subcategories, user expert level, user votes to answers *etc.*

One of the serious concerns in cQA is *"question-starvation"* (Li and King, 2010) where a question does not get immediate answer from any user. When this happens, the question may take several hours and sometimes days to get satisfactory answers or may not get answered at all. This delay in response may be avoided by retrieving semantically related questions from the cQA archives. If a similar question is found in the database of previous questions, then the corresponding best answer can be provided without any delay. However, the major challenge associated with retrieval of similar questions is the *lexico-syntactic gap* between them. Two questions may mean the same thing but they may differ lexically and syntactically. For example the queries *"Why are yawns contagious?"* and *"Why do we yawn when we see somebody else yawning?"* convey the same meaning but differ drastically from each other in terms of words and syntax.

Several techniques have been proposed in the literature for similar question retrieval and they could be broadly classified as follows:

1. **Classic Term Weighting Based Approaches:** Classical IR based retrieval

[1]https://answers.yahoo.com/
[2]http://zhidao.baidu.com/
[3]http://www.quora.com/
[4]http://stackoverflow.com/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 378–387,
Berlin, Germany, August 7-12, 2016.

models like $BM25$ (Robertson et al., 1994) and Language modeling for Information Retrieval ($LMIR$) (Zhai and Lafferty, 2004) score the similarity based on the weights of the matching text terms between the questions.

2. **Translation Models:** Learning word or phrase level translation models from question-answer pairs in parallel corpora of same language (Jeon et al., 2005; Xue et al., 2008; Zhou et al., 2011). The similarity function between questions is then defined as the probability of translating a given question into another.

3. **Topic Models:** Learning topic models from question-answer pairs (Ji et al., 2012; Cai et al., 2011; Zhang et al., 2014). Here, the similarity between questions, is defined in the latent topic space discovered by the topic model.

4. **Deep Learning Based Approaches:** Deep Learning based models like (Zhou et al., 2016),(Qiu and Huang, 2015), (Das et al., 2016) use variants of neural network architectures to model question-question pair similarity.

Retrieving semantically similar questions can be thought of as a classification problem with large number of categories. Here, each category contains a set of related questions and the number of questions per category is small. It is possible that given a test question, we find that there are no questions semantically related to it in the archives, it will belong to a entirely new unseen category. Thus, only a subset of the categories is known during the time of training. The intuitive approach to solve this kind of problem would to learn a similarity metric between the question to be classified and the archive of previous questions. Siamese networks have shown promising results in such distance based learning methods (Bromley et al., 1993; Chopra et al., 2005). These networks possess the capability of learning the similarity metric from the available data, without requiring specific information about the categories.

In this paper, we propose a novel unified model called *Siamese Convolutional Neural Network for cQA*. $SCQA$ architecture contain deep convolutional neural networks as twin networks with a contrastive energy function at the top. These twin networks share the weights with each other (parameter sharing). The energy function used is suitable for discriminative training for Energy-Based models (LeCun and Huang, 2005). $SCQA$ learns the shared model parameters and the similarity metric by minimizing the energy function connecting the twin networks. Parameter sharing guarantees that question and its relevant answer are nearer to each other in the semantic space while the question and any answer irrelevant to it are far away from each other. For example, the representations of *"President of USA"* and *"Barack Obama"* should be nearer to each other than those of *"President of USA"* and *"Tom Cruise lives in USA"*. The learnt similarity metric is used to retrieve semantically similar questions from the archives given a new posted question.

Similar question pairs are required to train SCQA which is usually hard to obtain in large numbers. Hence, $SCQA$ overcomes this limitation by leveraging Question-Answer pairs (Q, A) from the cQA archives. This also has additional advantages such as:

- The knowledge and expertise of the answerers and askers usually differ in a cQA forum. The askers, who are novices or non-experts, usually use less technical terminology whereas the answerers, who are typically experts, are more likely to use terms which are technically appropriate in the given realm of knowledge. Due to this, a model which learns from Question-Answer (Q, A) training data has the advantage of learning mappings from non-technical and simple terms to technical terms used by experts such as $shortsight => myopia$ etc. This advantage will be lost if we learn from (Q, Q) pairs where both the questions are posed by non-experts only.

- Experts usually include additional topics that are correlated to the question topic which the original askers may not even be aware of. For example, for the question *"how can I overcome short sight?"*, an expert may give an answer containing the concepts *"laser surgery"*, *"contact lens"*, *"LASIK surgery"* etc. Due to this, the concept *short sight* gets associated with these expanded concepts as well. Since, the askers are non-experts, such

rich concept associations are hard to learn from (Q, Q) training archives even if they are available in large scale. Thus, leveraging (Q, A) training data leads to learning richer concept/term associations in SCQA.

In summary, the following are our main contributions in this paper:

- We propose a novel model $SCQA$ based on Siamese Convolutional Neural Network which use shared parameters to learn the similarity metric between question-answer pairs in a cQA dataset.
- In SCQA, we overcome the non-availability of training data in the form of question-question pairs by leveraging existing question-answer pairs from the cQA archives which also helps in improving the effectiveness of the model.
- We reduce the computational complexity by directly using character-level representations of question-answer pairs in stead of using sentence modeling based representations which also helps in handling spelling errors and out-of-vocabulary (OOV) words in documents.

The rest of the paper is organized as follows. Section 2 presents the previous approaches to conquer the problem. Section 3 describes the architecture of $SCQA$. Sections 4 and 5 explain the training and testing phase of $SCQA$ respectively. Section 6 introduces a variant of $SCQA$ by adding textual similarity to it. Section 7 describes the experimental set-up, details of the evaluation dataset and evaluation metrics. In Section 8, quantitative and qualitative results are presented. Finally, Section 9 concludes the paper.

2 Related Work

The classical retrieval models $BM25$ (Robertson et al., 1994), $LMIR$ (Zhai and Lafferty, 2004) do not help much to capture semantic relatedness because they mainly consider textual similarity between queries. Researchers have used translation based models to solve the problem of question retrieval. Jeon *et al.* (2005) leveraged the similarity between the archived answers to estimate the translation probabilities. Xue *et al.* (2008) enhanced the performance of word based translation model by combining query likelihood language model to it. Zhou *et al.* (2011) used phrase based translation model where they considered question answer pairs as parallel corpus. However, Zhang *et al.* (2014) stated that questions and answers cannot be considered parallel because they are heterogeneous in lexical level and in terms of user behaviors. To overcome these vulnerabilities topic modeling was introduced by (Ji et al., 2012; Cai et al., 2011; Zhang et al., 2014). The approach assumes that questions and answers share some common latent topics. These techniques match questions not only on a term level but also on a topic level.

Zhou *et al.* (2015) used a fisher kernel to model the fixed size representation of the variable length questions. The model enhances the embedding of the questions with the metadata "category" involved with them. Zhang *et al.* (2016) learnt representations of words and question categories simultaneously and incorporated the learnt representations into traditional language models.

Following the recent trends, deep learning is also employed to solve this problem. Qiu *et al.* (2015) introduced convolutional neural tensor network (CNTN), which combines sentence modeling and semantic matching. CNTN transforms the word tokens into vectors by a lookup layer, then encode the questions and answers to fixed-length vectors with convolutional and pooling layers, and finally model their interactions with a tensor layer. Das *et al.* (2016) used deep structured topic modeling that combined topic model and paired convolutional networks to retrieve related questions. Zhou *et al.* (2016) used a deep neural network (DNN) to map the question answer pairs to a common semantic space and calculated the relevance of each answer given the query using cosine similarity between their vectors in that semantic space. Finally they fed the learnt semantic vectors into a learning to rank (LTR) framework to learn the relative importance of each feature.

On a different line of research, several Textual-based Question Answering (QA) systems (Qanda[5], QANUS[6], QSQA[7] *etc.*) are developed that retrieve answers from the Web and other textual sources. Similarly, structured QA systems

[5] http://www.openchannelfoundation.org/projects/Qanda/

[6] http://www.qanus.com/

[7] http://www.dzonesoftware.com/products/open-source-question-answer-software/

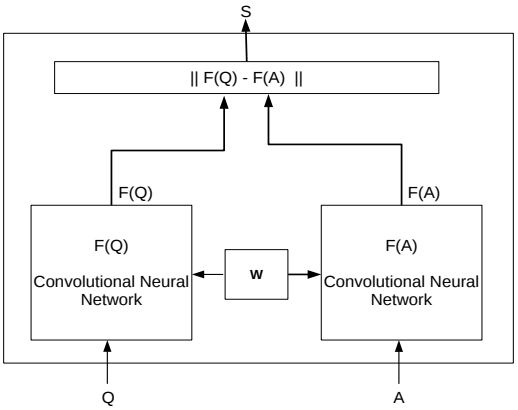


Figure 1: Architecture of Siamese network.

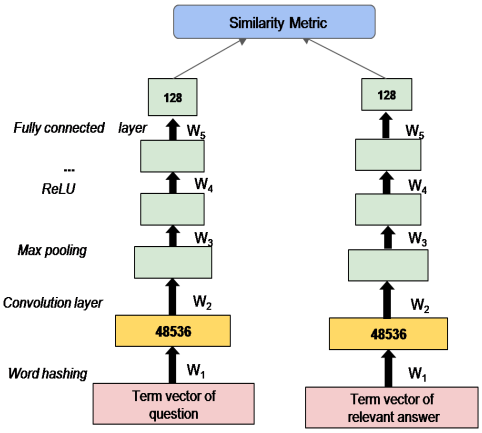


Figure 2: Architecture of SCQA. The network consists of repeating convolution, max pooling and ReLU layers and a fully connected layer. Also the weights W_1 to W_5 are shared between the sub-networks.

(Aqualog[8], NLBean[9] *etc.*) obtain answers from structured information sources with predefined ontologies. QALL-ME Framework (Ferrandez et al., 2011) is a reusable multilingual QA architecture built using structured data modeled by an ontology. The reusable architecture of the system may be utilized later to incorporate multilingual question retrieval in $SCQA$.

2.1 Siamese Neural Network

Siamese Neural Networks (shown in Figure 1) were introduced by Bromley *et al.* (1993) to solve the problem of signature verification. Later, Chopra *et al.* (2005) used the architecture with discriminative loss function for face verification. Recently these networks are used extensively to enhance the quality of visual search (Liu et al., 2008; Ding et al., 2008).

Let, $F(X)$ be the family of functions with set of parameters W. $F(X)$ is assumed to be differentiable with respect to W. Siamese network seeks a value of the parameter W such that the symmetric similarity metric is small if X_1 and X_2 belong to the same category, and large if they belong to different categories. The scalar energy function $S(Q, A)$ that measures the semantic relatedness between question answer pair (Q,A) can be defined as:

$$S(Q, A) = \|F(Q) - F(A)\| \quad (1)$$

In $SCQA$ question and relevant answer pairs are fed to train the network. The loss function is minimized so that $S(Q, A)$ is small if the answer A is relevant to the question Q and large otherwise.

[8] http://technologies.kmi.open.ac.uk/aqualog/

[9] http://www.markwatson.com/opensource/

3 Architecture of SCQA

As shown in Figure 2, $SCQA$ consists of a pair of deep convolutional neural networks (CNN) with convolution, max pooling and rectified linear (ReLU) layers and a fully connected layer at the top. CNN gives a non linear projection of the question and answer term vectors in the semantic space. The semantic vectors yielded are connected to a layer that measures distance or similarity between them. The contrastive loss function combines the distance measure and the label. The gradient of the loss function with respect to the weights and biases shared by the sub-networks, is computed using back-propagation. Stochastic Gradient Descent method is used to update the parameters of the sub-networks.

3.1 Inputs to SCQA

The size of training data used is in millions, thus representing every word with one hot vector would be practically infeasible. Word hashing introduced by Mcnamee *et al.* (2004) involves letter n-gram to reduce the dimensionality of term vectors. For a word, say, "table" represented as (#table#) where # is used as delimiter, letter 3-grams would be #ta, tab, abl, ble, le#. Thus word hashing is character level representation of documents which takes care of OOV words and words with minor spelling errors. It represents a query using a lower dimensional vector with dimension equal to number of unique letter trigrams in the training dataset (48,536 in our case).

The input to the twin networks of $SCQA$ are word hashed term vectors of the question and

answer pair and a label. The label indicates whether the sample should be placed nearer or farther in the semantic space. For positive samples (which are expected to be nearer in the semantic space), twin networks are fed with word hashed vectors of question and relevant answers which are marked as "best-answer" or "most voted answers" in the cQA dataset of Yahoo! Answers (question-relevant answer pair). For negative samples (which are expected to be far away from each other in the semantic space), twin networks are fed with word hashed vectors of question and answer of any other random question from the dataset (question-irrelevant answer pair).

3.2 Convolution

Each question-answer pair is word hashed into (q_i-a_i) such that $q_i \in \mathbb{R}^{n_t}$ and $a_i \in \mathbb{R}^{n_t}$ where n_t is the total number of unique letter trigrams in the training data. Convolution layer is applied on the word hashed question answer vectors by convolving a filter with weights $\mathbf{c} \in \mathbb{R}^{hxw}$ where h is the filter height and w is the filter width. A filter consisting of a layer of weights is applied to a small patch of word hashed vector to get a single unit as output. The filter is slided across the length of vector such that the resulting connectivity looks like a series of overlapping receptive fields which output of width w.

3.3 Max Pooling

Max pooling performs a kind of non-linear down-sampling. It splits the filter outputs into small non-overlapping grids (larger grids result to greater the signal reduction), and take the maximum value in each grid as the value in the output of reduced size. Max pooling layer is applied on top of the output given by convolutional network to extract the crucial local features to form a fixed-length feature vector.

3.4 ReLU

Non-linear function Rectified linear unit (ReLU) is applied element-wise to the output of max pooling layer. ReLU is defined as $f(x) = max(0, x)$. ReLU is preferred because it simplifies backpropagation, makes learning faster and also avoids saturation.

3.5 Fully Connected layer

The terminal layer of the convolutional neural sub-networks is a fully connected layer. It converts the output of the last ReLU layer into a fixed-length semantic vector $\mathbf{s} \in \mathbb{R}^{n_s}$ of the input to the sub-network. We have empirically set the value of n_s to 128 in $SCQA$.

4 Training

We train $SCQA$ for a question while looking for semantic similarity with the answers relevant to it. $SCQA$ is different from the other deep learning counterparts due to its property of parameter sharing. Training the network with a shared set of parameters not only reduces number of parameters (thus, save lot of computations) but also ensures consistency of the representation of questions and answers in semantic space. The shared parameters of the network are learnt with the aim to minimize the semantic distance between the question and the relevant answers and maximize the semantic distance between the question and the irrelevant answers.

Given an input $\{q_i, a_i\}$ where q_i and a_i are the i^{th} question answer pair, and a label y_i with $y_i \in \{1,\text{-}1\}$, the loss function is defined as:

$$loss(q_i, a_i) = \begin{cases} 1 - \cos(q_i, a_i), & \text{if } y = 1; \\ \max(0, \cos(q_i, a_i) - m), & \text{if } y = -1; \end{cases}$$

where m is the margin which decides by how much distance dissimilar pairs should be moved away from each other. It generally varies between 0 to 1. The loss function is minimized such that question answer pairs with label 1 (question-relevant answer pair) are projected nearer to each other and that with label -1 (question-irrelevant answer pair) are projected far away from each other in the semantic space. The model is trained by minimizing the overall loss function in a batch. The objective is to minimize :

$$L(\Lambda) = \sum_{(q_i,a_i) \in C \cup C'} loss(q_i, a_i) \qquad (2)$$

where C contains batch of question-relevant answer pairs and C' contain batch of question-irrelevant answer pairs. The parameters shared by the convolutional sub-networks are updated using Stochastic Gradient escent (SGD).

5 Testing

While testing, we need to retrieve similar questions given a query. During testing we make pairs of all the questions with the query and feed them

to $SCQA$. The term vectors of the question pairs are word hashed and fed to the twin sub-networks. The trained shared weights of the $SCQA$ projects the question vector in the semantic space. The similarity between the pairs is calculated using the similarity metric learnt during the training. Thus $SCQA$ outputs a value of distance measure (score) for each pair of questions. The threshold is dynamically set to the average similarity score across questions and we output only those which have a similarity greater than the average similarity score.

6 Siamese Neural Network with Textual Similarity

$SCQA$ is trained using question-relevant answer pairs as positive samples and question-irrelevant answer pairs as negative samples. It poorly models the basic text similarity because in the (Q, A) training pairs, the answerers often do not repeat the question words while providing the answer. For example: for the question "Who is the President of the US?", the answerer would just provide "Barrack Obama". Due to this, although the model learns that $president\, the\, US \implies Barrack\, Obama$, the similarity for $president \implies president$ wouldn't be much and hence needs to be augmented through $BM25$ or some such similarity function.

Though $SCQA$ can strongly model semantic relations between documents, it needs boosting in the area of textual similarity. The sense of word based similarity is infused to $SCQA$ by using BM25 ranking algorithm. Lucene[10] is used to calculate the BM25 scores for question pairs. The score from similarity metric of $SCQA$ is combined with the BM25 score. A new similarity score is calculated by the weighted combination of the $SCQA$ and $BM25$ score as:

$$score = \alpha * SCQA_{score} + (1 - \alpha) * BM25_{score} \quad (3)$$

where α control the weights given to $SCQA$ and $BM25$ models. It range from 0 to 1. $SCQA$ with this improved similarity metric is called Siamese Convolutional Neural Network for cQA with Textual Similartity (*T-SCQA*). Figure 4 depicts the testing phase of *T-SCQA*. This model will give better performance in datasets with good mix of questions that are lexically and semantically similar. The value of α can be tuned according to the nature of dataset.

[10] https://lucene.apache.org/

Hyperparameter	Value
Batch Size	100
Depth of CNN	3
Learning rate	0.01
Momentum	0.05
Kernel width of Convolution	10
Kernel width of MaxPooling	100
Length of semantic vector	128

Table 1: Hyperparameters of $SCQA$.

7 Experiments

We collected Yahoo! Answers dataset from Yahoo! Labs Webscope[11]. Each question in the dataset contains title, description, best answer, most voted answers and meta-data like categories, sub categories etc. For training dataset, we randomly selected 2 million data and extracted question-relevant answer pairs and question-irrelevant answer pairs from them to train $SCQA$. Similarly, our validation dataset contains 400,000 question answer pairs. The hyperparameters of the network are tuned on the validation dataset. The values of the hyperparameters for which we obtained the best results is shown in Table 1.

We used the annotated survey dataset of 1018 questions released by Zhang *et al.* (2014) as testset for all the models. On this gold data, we evaluated the performance of the models with three evaluation criteria: Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) and Precision at K (P@K).

Each question and answer was pre-processed by lower-casing, stemming, stopword and special character removal.

7.1 Parameter Sharing

In order to find out whether parameter sharing helps in the present task we build two models named Deep Structured Neural Network for Community Question Answering($DSQA$) and Deep Structured Neural Network for Community Question Answering with Textual Similarity *T-DSQA*. $DSQA$ and *T-DSQA* have the same architecture as $SCQA$ and *T-SCQA* with the exception that in

[11] http://webscope.sandbox.yahoo.com/catalog.php?datatype=l

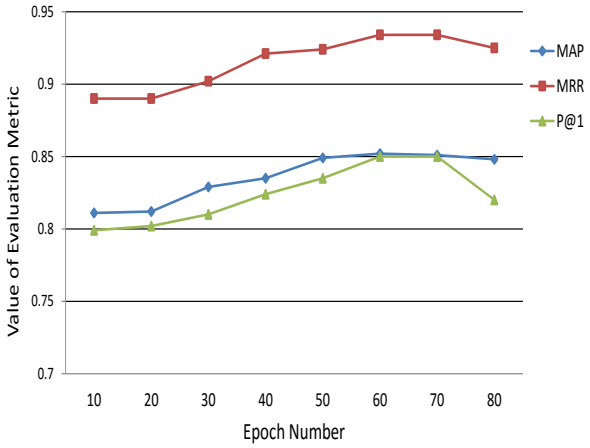


Figure 3: Variation of evaluation metrics with the epochs.

Method	MAP	MRR	P@1
LMIR	0.762	0.844	0.717
translation(word)	0.786	0.870	0.807
translation+LM	0.787	0.869	0.804
translation(phrase)	0.789	0.875	0.817
Q-A topic model	0.787	0.879	0.810
Q-A topic model(s)	0.800	0.888	0.820
DSQA	0.755	0.921	0.751
T-DSQA	0.801	0.932	0.822
SCQA	0.811	0.895	0.830
T-SCQA	**0.852***	**0.934***	**0.849***

Table 2: Results on Yahoo! Answers dataset. The best results are obtained by *T-SCQA* (bold faced). The difference between the results marked(*) and other methods are statistically significant with p <0.001.

the former models weights are not shared by the convolutional sub-networks. The weightage α for controlling corresponding scores of $SCQA$ and $BM25$ for the model *T-SCQA* was tuned on the validation set.

8 Results

We did a comparative study of the results of the previous methods with respect to $SCQA$ and *T-SCQA*. The baseline performance is shown by query likelihood language model (LM). For the translation based methods $translation(word)$, $translation+LM$ and $translation(phrase)$ we implemented the papers by Jeon *et al.* (2005), Xue *et al.* (2008), Zhou *et al.* (2011) respectively. The first paper deals with word based translation, the second enhanced the first by adding language model to it and the last paper implements phrase based translation method to bridge lexical gap. As seen from Table 2, the translation based methods outperforms the baseline significantly. The models are trained using GIZA++[12] tool with the question and best answer pair as the parallel corpus. For the topic based *Q-A topic model* and $Q\text{-}A\ topic\ model(s)$, we implemented the models $QATM\text{-}PR$ (Question-Answer Topic Model) Ji *et al.*(2012) and $TBLM_{SQATM-V}$ (Supervised Question-Answer Topic Model with user votes as supervision) Zhang *et al.* (2014) respectively. Again it is visible from the Table 2 that topic based approaches show slight improvement over translation based methods but they show significant improvement over baseline. The models $DSQA$ and *T-DSQA* were built using convolutional neural sub-networks joined by a distance measure at the top. There is no sharing of parameters involved between the sub-networks of these models. It is clear from the comparison of results between *T-DSQA* and *T-SCQA* that parameter sharing definitely helps in the task of similar question retrieval in cQA forums. *T-SCQA* outperforms all the previous approaches significantly.

8.1 Quantitative Analysis

$SCQA$ and *T-SCQA* learns the semantic relationship between the question and their best and most voted answers. It is observed that by varying the weights of $SCQA$ and $BM25$ scores, the value of MAP changes significantly (Figure 5). The weight is tuned in the validation dataset. We trained our model for several epochs and observed how the results varied with the epochs. We found that the evaluation metrics changed with increasing the number of epochs but became saturated after epoch 60. The comparison of evaluation metrics with epochs can be visualised in Figure 3.

The comparisons $SCQA$ and *T-SCQA* with the previously proposed models is shown in Table 2. For baseline we considered the traditional language model $LMIR$. The results in the table are consistent with the literature which says translation based models outperform the baseline methods and topic based approaches outperform the translational methods. Also, it is observed that deep learning based solution with parameter sharing is more helpful for this task than without parameter sharing. Note, that the results of previous models stated in Table 2 differ from the original

[12] http://www.statmt.org/moses/giza/GIZA++.html

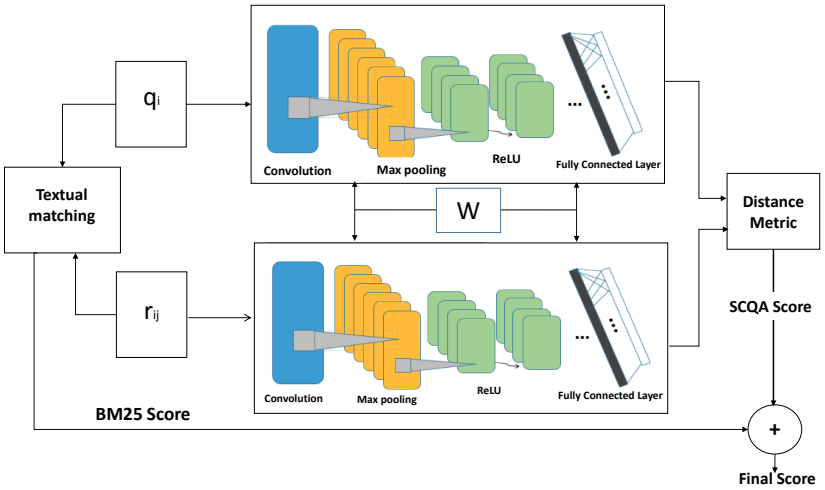


Figure 4: Testing phase of *T-SCQA*. Here the q_i is the i^{th} query and r_{ij} is the j^{th} question retrieved by q_i. The twin CNN networks share the parameters ($\mathbf{W}$) with each other. The connecting distance metric layer outputs the $SCQA$ score and the textual matching module outputs the $BM25$ score. The weighted combination of these scores give the final score. r_{ij} is stated similar to the query q_i if the final score of the pair exceeds an appropriate threshold.

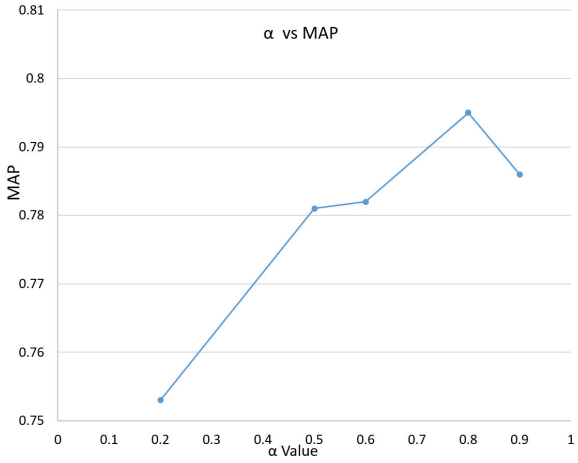


Figure 5: The variation of MAP with α.

papers since we tried to re-implement those models with our training data (to the best of our capability). Though we use the test data released by Zhang *et al.* (2014) we do not report their results in Table 2 due to the difference in training data used to train the models.

In the test dataset released by Zhang *et al.* (2014), there are fair amount of questions that possess similarity in the word level hence *T-SCQA* performed better than $SCQA$ for this dataset. *T-SCQA* gives the best performance in all evaluation measures. The results of *T-SCQA* in Table 2 uses the trained model at epoch 60 with the value of α as 0.8.

8.2 Qualitative Analysis

In Table 3 few examples are shown to depict how results of *T-SCQA* reflect strong semantic information when compared to other baseline methods. For Q1 we compare performance of *LMIR* and *T-SCQA*. *LMIR* outputs the question by considering word based similarity. It focuses on matching the words "how", "become", "naturally" *etc*, hence it outputs "How can I be naturally funny?" which is irrelevant to the query. On the other hand, $T-SCQA$ retrieves the questions that are semantically relevant to the query. For Q2 we compare the performance of *T-SCQA* with phrase based translation model (Zhou et al., 2011). The outputs of *translation(phrase)* model shows that the translation of "nursery" and "pre-school" to "day-care", "going to university" to "qualifications" are highly probable. The questions retrieved are semantically related, however asking craft ideas for pre-school kids for the event of mother's day is irrelevant in this context. The results of our model solely focuses on the qualifications, degrees and skills one needs to work in a nursery. For Q3 we compare the performance of *T-SCQA* with supervised topic model (Zhang et al., 2014). The questions retrieved by both the models revolve around the topic "effect of smoking on children". While the topic model retrieve questions which deal with smoking by mother and its effect on child, *T-SCQA* retrieve questions which deals not only with the affects of a mother smoking but also the effect of passive smoking on the child. For Q4 we com-

Query		Comment
Q1: How can I become naturally happy?		
$LMIR$	1.How can I be naturally happy? 2.How can I become naturally funny?	$LMIR$ performs word based matching using "how","become", "naturally" *etc.*
T-SCQA	1.Are some of us naturally born happy or do we learn how to become happy? 2.How can I become prettier and feel happier with myself?	
Q2: Do you need to go to university to work in a nursery or pre-school?		For *translation* (phrase) university->degree nursery->daycare are highly probable translations but craft ideas for daycare is irrelevant.
$translation$ $(phrase)$	1.What degree do you need to work in a nursery? 2. I work at a daycare with pre-school kids(3-5). Any ideas on crafts for mother's day?	
T-SCQA	1.Will my B.A hons in childhood studies put me in as an unqualified nursery nurse? 2.What skills are needed to work in a nursery, or learned from working in a nursery?	
Q3: Does smoking affect an unborn child?		Both models retrieve questions on topic "effect of smoking on children" but *T-SCQA* could retrieve based on passive smoking through father.
Q-A topic model(s)	1.How do smoking cigarettes and drinking affect an unborn child? 2.How badly will smoking affect an unborn child?	
T-SCQA	1.How does cigarette smoking and alcohol consumption by mothers affect unborn child? 2.Does smoking by a father affect the unborn child? If there is no passive smoking, then is it still harmful?	
Q4: How do I put a video on YouTube?		*T-DSQA* could not decipher "put". It relates "put" to download and transfer of videos while *T-SCQA* relates it to uploading videos.
T-DSQA	1.How can I download video from YouTube and put them on my Ipod? 2.I really want to put videos from YouTube to my Ipod..how?	
T-SCQA	1.How do I post a video on YouTube? 2.How can I make a channel on YouTube and upload videos on it? plz help me...	

Table 3: This table compares the qualitative performance of *T-SCQA* with *LMIR*, phrase based translation model *translation(phrase)*, supervised topic model *Q-A topic model(s)* and deep semantic model without parameter sharing *T-DSQA*. For queries Q1-4 *T-SCQA* show better performance than the previous models .

pare the performance of *T-SCQA* with *T-DSQA*. *T-DSQA* retrieves the questions that are related to downloading and transferring YouTube videos to other devices. Thus, *T-DSQA* cannot clearly clarify the meaning of "put" in Q4. However, the retrieved questions of *T-SCQA* are more aligned towards the ways to record videos and upload them in YouTube. The questions retrieved by *T-SCQA* are semantically more relevant to the query Q4.

9 Conclusions

In this paper, we proposed $SCQA$ for similar question retrieval which tries to bridge the lexico-syntactic gap between the question posed by the user and the archived questions. $SCQA$ employs twin convolutional neural networks with shared parameters to learn the semantic similarity between the question and answer pairs. Interpolating $BM25$ scores into the model *T-SCQA* results in improved matching performance for both textual and semantic matching. Experiments on large scale real-life "Yahoo! Answers" dataset revealed that T-SCQA outperforms current state-of-the-art approaches based on translation models, topic models and deep neural network based models which use non-shared parameters.

As part of future work, we would like to enhance SCQA with the meta-data information like categories, user votes, ratings, user reputation of the questions and answer pairs. Also, we would like to experiment with other deep neural architectures such as Recurrent Neural Networks, Long Short Term Memory Networks, *etc.* to form the sub-networks.

References

Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a siamese time delay neural network. *IJPRAI*.

Li Cai, Guangyou Zhou, Kang Liu, and Jun Zhao. 2011. Learning the latent topics for question retrieval in community QA. *IJCNLP*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. *CVPR*.

Arpita Das, Manish Shrivastava, and Manoj Chinnakotla. 2016. Mirror on the wall: Finding similar questions with deep structured topic modeling. Springer.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. Using conditional random fields to extract contexts and answers of questions from online forums. *ACL*.

Oscar Ferrandez, Christian Spurk, Milen Kouylekov, Iustin Dornescu, Sergio Ferrandez, Matteo Negri, Ruben Izquierdo, David Tomas, Constantin Orasan, Guenter Neumann, et al. 2011. The qall-me framework: A specifiable-domain multilingual question answering architecture. *Web semantics*.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. *CIKM*.

Zongcheng Ji, Fei Xu, Bin Wang, and Ben He. 2012. Question-answer topic model for question retrieval in community question answering. *CIKM*.

Yann LeCun and Fu Jie Huang. 2005. Loss functions for discriminative training of energy-based models. *AISTATS*.

Baichuan Li and Irwin King. 2010. Routing questions to appropriate answerers in community question answering services. *CIKM*.

Yuanjie Liu, Shasha Li, Yunbo Cao, Chin-Yew Lin, Dingyi Han, and Yong Yu. 2008. Understanding and summarizing answers in community-based question answering services. *ICCL*.

Paul Mcnamee and James Mayfield. 2004. Character n-gram tokenization for european language text retrieval. *Information retrieval*.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. *IJCAI*.

Stephen E Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, Mike Gatford, et al. 1994. Okapi at trec-3. *NIST Special Publication*.

Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. 2008. Retrieval models for question and answer archives. *SIGIR*.

Chengxiang Zhai and John Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*

Kai Zhang, Wei Wu, Haocheng Wu, Zhoujun Li, and Ming Zhou. 2014. Question retrieval with high quality answers in community question answering. *CIKM*.

Kai Zhang, Wei Wu, Fang Wang, Ming Zhou, and Zhoujun Li. 2016. Learning distributed representations of data in community question answering for question retrieval. *ICWSDM*.

Guangyou Zhou, Li Cai, Jun Zhao, and Kang Liu. 2011. Phrase-based translation model for question retrieval in community question answer archives. *ACL:HLT*.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. *ACL*.

Guangyou Zhou, Yin Zhou, Tingting He, and Wensheng Wu. 2016. Learning semantic representation with neural networks for community question answering retrieval. *Knowledge-Based Systems*.

News Citation Recommendation with Implicit and Explicit Semantics

Hao Peng,[*1] **Jing Liu,**[2] **Chin-Yew Lin**[2]

[1]School of EECS, Peking University, Beijing, 100871, China
[2]Microsoft Research, Beijing, 100080, China
penghao.pku@gmail.com {liudani, cyl}@microsoft.com

Abstract

In this work, we focus on the problem of news citation recommendation. The task aims to recommend news citations for both authors and readers to create and search news references. Due to the sparsity issue of news citations and the engineering difficulty in obtaining information on authors, we focus on content similarity-based methods instead of collaborative filtering-based approaches. In this paper, we explore word embedding (i.e., implicit semantics) and grounded entities (i.e., explicit semantics) to address the variety and ambiguity issues of language. We formulate the problem as a reranking task and integrate different similarity measures under the learning to rank framework. We evaluate our approach on a real-world dataset. The experimental results show the efficacy of our method.

1 Introduction

When an author writes an online news article, s/he often cites previously published news reports to elaborate a mentioned event or support his/her point of view. For the convenience of the readers, the editor usually associates the words with hyperlinks. Through the links the readers can directly access the referenced articles to know more details about the events. If there is no reference for a mentioned event, the readers may search the related news reports for further reading. Hence, it is valuable to have automatic news citation recommendations for authors and readers to create or search news references.

In this paper, we focus on the problem of *news citation recommendation*. As shown in Table 1, given a snippet of citing context (left), the task aims to retrieve a list of news articles (right) as references. This task differs from traditional recommendation tasks, e.g., citation recommendation for scientific papers, in that: (a) based on the statistics from our dataset, the number of references per news article is 4.56 on average, much less than the number of citations per academic paper (typically dozens); (b) the author-topic information is usually unavailable, since it is technically difficult to obtain author information from news articles. These differences make the collaborative filtering-based methods, which have been widely applied to paper citation recommendation, less available in our scenario. Therefore, in this paper we focus on content similarity-based methods to deal with the task of news citation recommendation.

Previous studies use string-based overlap (Xu et al., 2014), machine translation measures (Madnani et al., 2012), and dependency syntax (Wan et al., 2006; Wang et al., 2015) to model text similarity. More recent work focuses on neural network methods (Yin and Schütze, 2015; He et al., 2015; Hu et al., 2014; dos Santos et al., 2015; Lei et al., 2016). There are two major challenges rendering these approaches not suitable for this task: (i) the variety and (ii) the ambiguity of language. By variety, we mean that the same meaning may be expressed with different phrases. Taking the first row in Table 1 for example, *Vlaar* in the citing context refers to *Ron Vlaar*, a Dutch football player, who is referred to as *Dutch star* and *Netherlands international* in the cited article. By ambiguity, we mean that the same expression may have different meanings in different contexts. In the second example in Table 1, the mention *tiger* refers to *tiger the mammal*. By contrast, in "Detroit Tigers links: The Tigers are in trouble" for example, the word *Tiger* is the name of a team. In this paper, we explore both implicit and explicit semantics to ad-

[*] Work done during internship at Microsoft Research.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 388–398,
Berlin, Germany, August 7-12, 2016.

Citing Context	Cited Article
...	**Man United and Arsenal on red alert as top Dutch star officially joins free agent list**
Manchester United and Arsenal have both been interested in Vlaar in the past, suggesting Southampton will have to fight hard to land him.	The Netherlands international has joined the free agent list today and is no longer contractually obliged to remain at Villa Park.
...	...
...	**Bangladesh 's abundant tiger population has collapsed to just 100**
Conservationists want the Bangladeshi government to step up and help save the tigers through greater administration and enforcement of anti-poaching laws, as Bangladesh does not legally protect tigers to the extent that other governments do, according to Inhabitat.	In Bangladesh, a new census shows that tiger populations in the Sundarbans mangrove forest are more endangered than ever. The study, which used hidden cameras to track and record tigers, provides a more accurate update than previous surveys that used other methods. The year-long census, which ended this April, revealed only around 100 of the big cats remain in what was once home to the largest population of tigers on earth.
...	...

Table 1: Two pair of news snippets. For readability concerns, we keep only the sentence associated with an anchor link in the citing part, and the **title** and lead paragraph of the cited part.

dress the above issues. Specifically, the implicit semantics can be obtained from the word embedding trained on large scale corpus, and the explicit semantics through linking entity mentions to the grounded entities in a knowledge base.

In this paper, we explore using both word embedding and grounded knowledge to model the relatedness between citing context and articles. We formulate the problem as a re-ranking task. We use learning to rank to integrate different similarity measures and evaluate the models on a real world dataset constructed from Bing News[1]. We further give quantitative analysis of the effects of word embedding and grounded entities in the task.

In summary, the main contributions of this paper are three-fold:

- We propose the task of *news citation recommendation* and construct a real-world dataset for this task.
- We utilize both word embedding based similarity measures and knowledge-based methods to tackle the problem. We formulate the problem as a re-ranking task and leverage learning to rank algorithm to integrate different similarity measures.
- We conduct extensive experiments on a large dataset. The results show the effectiveness of word embedding and grounded entities. We further quantitatively analyze how the implicit semantics from word embedding and explicit semantics from grounded knowledge benefit the task of interest.

[1] https://www.bing.com/news

2 Problem Formulation

In this section, we introduce the *news citation recommendation* problem and formulate it as a re-ranking task. We first introduce definitions that will be used through the rest of the paper:

Citing Context. *Citing context* is a sentence which contains an anchor text associated with a hyperlink. As shown in Table 1, the underlined words are associated with a hyperlink pointing to another news article, and the sentence (left) which contains the anchor is the citing context.

Cited Article. Given a piece of citing context, the article that the hyperlink links to is defined as its *cited article*. It is expected that a news article is well-structured, and its headline together with its lead paragraph gives a good brief description of the whole story (Kianmehr et al., 2009). In this paper, a news article can either be represented by its title and lead paragraph or by the passage as a whole. We conduct experiments under both of the two different settings.

Candidate Article Set. Considering efficiency, we follow the procedure adopted by many recommendation systems (Lei et al., 2016; Tan et al., 2015) and formulate the problem as a re-ranking task. In another word, given a citing context, we first use efficient retrieval methods with high recall to generate a list of articles as the *candidate article set*, and then run the system to get a re-ranked list.

News Citation Recommendation. Given a citing context, the task aims to construct an ordered list of news articles, top of which are most relevant to the context, and can serve as the cited articles.

3 Method

In this section, we first explain the similarity measures based on word embedding (implicit semantics) and grounded knowledge (explicit semantics) to deal with variety and ambiguity problems. Then we briefly introduce the baselines and the learning to rank framework.

3.1 Implicit Semantics for Variety

The distributed word representation by *word2vec* factors word distance and captures semantic similarities through vector arithmetic (Mikolov et al., 2013). In this work, we train a skip-gram model to bridge the vocabulary gap between context-article pairs. Previous work represents the documents with averaged vectors of words (Tang et al., 2014; Tan et al., 2015). However, this may lead to the loss of detailed information of the documents. In this paper, we adopt a different approach, explained below.

Word Mover's Distance (WMD). Kusner et al. (2015) combine distributed word representations with the earth mover's distance (EMD) (Rubner et al., 1998; Wan, 2007) to measure the distance between documents. They use the Euclidean distance between words' low dimensional representations as building blocks, and optimize a special case of the EMD to obtain the cumulative distance. More formally, let $X = \{(x_1, w_{x_1}), (x_2, w_{x_2}), \cdots, (x_m, w_{x_m})\}$ be the normalized bag-of-words representation for a citing context after removing stop-words, where word x_i appears w_{x_i} times (then normalized by the total count of words in X), $i = 1, 2, \cdots, m$. Similarly, we have the representation for a candidate article, $Y = \{(y_1, w_{y_1}), (y_2, w_{y_2}), \cdots, (y_n, w_{y_n})\}$. The WMD calculates the minimum cumulative cost by solving the linear programming problem below:

$$\min_{\mathbf{T}} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{T}_{ij} c_{ij}$$

$$s.t. \quad \sum_{j=1}^{n} \mathbf{T}_{ij} = w_{x_i}, \quad i = 1, 2, \cdots, m,$$

$$\sum_{i=1}^{m} \mathbf{T}_{ij} = w_{y_j}, \quad j = 1, 2, \cdots, n,$$

where $\mathbf{T} \in \mathbb{R}^{m \times n}$ is the transportation flow matrix, and c_{ij} indicates the distance between x_i and y_j. Here $c_{ij} = \|\mathbf{vector}(x_i) - \mathbf{vector}(y_j)\|$, where function $\mathbf{vector}(w)$ returns the word vector of w. Then the distance is normalized by the total flow:

$$WMD(X, Y) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{T}_{ij} c_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{T}_{ij}}$$

3.2 Explicit Semantics for Ambiguity

News articles tend to be well written, and contain many named entity mentions. Making use of this property, we deal with the ambiguity problem by using grounded entities (explicit semantics). Given a context-article pair, we first recognize all named entity mentions on both sides and link them to knowledge bases (e.g., Wikipedia and Freebase), then use the following measures to model the similarity.

- **Entity Overlap.** Given a context–article pair, we consider two metrics, namely, $precision$ and $recall$, to measure their entity overlap. The $precision$ is defined as:

$$precision = \frac{entity\text{-}overlap(citing, cited)}{entity\text{-}count(citing)}$$

and $recall$ as:

$$recall = \frac{entity\text{-}overlap(citing, cited)}{entity\text{-}count(cited)}$$

- **Embedding Based Matching.** We build two separate information networks for Wikipedia entities using (a) the anchor links on Wikipedia pages and (b) the Freebase entity graph (Bollacker et al., 2008). Then we apply Large-scale Information Network Embedding (LINE) (Tang et al., 2015) system[2] to the networks to embed the entities into low-dimensional spaces. We then measure the similarity by the minimized cosine distance between entities' on the citing and the cited side:

$$minDIS_{citing} = \frac{1}{|citing|} \sum_{i=1}^{|citing|} \min_{y_j \in Y} (1 - cos(x_i, y_j)),$$

and vice versa:

$$minDIS_{cited} = \frac{1}{|citied|} \sum_{j=1}^{|cited|} \min_{x_i \in X} (1 - cos(x_i, y_j)),$$

where X refers to the citing context, and $x_i \in X$ are the grounded entities in the citing part. Similar with Y and y_j.

[2] https://github.com/tangjianpku/LINE

- **Wikipedia Evidence.** Given a context-article pair, we refer to world knowledge for supporting evidence. In particular, we first apply an entity linking system to detect the entity mentions on both sides and ground them into Wikipedia entries, each of which has its own description page. Second, we collect the descriptions for entities from the candidate article and extract as evidence those sentences containing entities from citing context. We refer to this evidence as *cited evidence*. For instance, the article in Table 8 contains grounded entity *Scottish National Party*. And in the description for it, there is a sentence containing the entity *Scotland* from the citing context: "The Scottish National Party (SNP) is a Scottish nationalist and social-democratic political party in *Scotland*." Thus we extract this sentence as cited evidence supporting this pair.
  We count the overlapping nouns between the citing context and the cited evidence to calculate $precision$ and $recall$,

$$precision = \frac{noun\text{-}overlap(context, cited\ evidence)}{noun\text{-}count(context)}$$

$$recall = \frac{noun\text{-}overlap(article, citing\ evidence)}{noun\text{-}count(article)}$$

3.3 Baselines

We design several baseline features for the two groups of features mentioned above:

- **TF-IDF Distance.** We use TF-IDF distance as a basic measure. The similarity is calculated with cosine distance based on TF-IDF vector representations for the text.
- **Ungrounded Mentions.** Note that entity overlap features also adapt to ungrounded mentions. The embedding-based matching features for ungrounded mentions are similar to those for grounded entities. The only difference is that here each mention is represented by the averaged vectors of all the words it contains. Wikipedia evidence is not feasible for ungrounded mentions.

Table 2 summarizes all the features we use. A cited article can either be represented by its headline+lead paragraph or as a whole. Therefore, we extract features under two different settings: (a) headlines and lead paragraphs only; (b) the full articles. Most of the features are extracted under both of the settings. However, feature 2 is much too computation-intensive and feature 7 needs POS-tagging as the preprocessing. Thus these two are only extracted under setting (a).

3.4 Learning to Rank Framework

Many different learning to rank algorithms have been proposed to deal with the ranking problem, including pointwise, pairwise, and listwise approaches (Xia et al., 2008). Listwise methods receive ranked lists as training samples, and directly optimize the metric of interest by minimizing the respective cost. And it has been reported that the listwise method usually achieves better performance compared to others (Qin et al., 2008; Cao et al., 2007). In this work, we use the linear model and apply coordinate ascent for parameter optimization.

4 Experiments

4.1 Data Collection

We collect one month's news articles from Bing News. The citing context set consists of all the sentences associated with anchor link(s). For each piece of citing context, its cited article is extracted through its hyperlink. If there are multiple links associated with the context, only the first one is considered. We pair each citing context and its cited article as a ground truth sample. We further label as ground truths those articles sharing the same title as the cited article. This is rather reasonable since a single passage may have multiple reprints by difference sources. On average, there are 2.20 ground truth cited articles for each citing context in the dataset.

In order to focus only on news events, we filter out those pairs whose hyperlinks are associated with three words or less (usually names for persons or places, and lead to definition pages). We also discard those samples whose citing contexts contain or are exactly the same as the titles of the cited articles. For example, "READ MORE: The stories you need to read, in one handy email" links to an article titled "The stories you need to read, in one handy email".

The dataset is preprocessed with Stanford CoreNLP toolkit (Manning et al., 2014), including sentence splitting, tokenizing for whole passages, and POS-tagging for titles and lead paragraphs. We use the JERL system by Luo et al. (2015) for entity detection and grounding. It recognizes entity mentions and links them to Wikipedia entries.

	Feature	Full Article?	# of features	Description
Dealing with Variety				
1	WMD	n	1	Word vector based earth mover's distance.
Dealing with Ambiguity				
2	Grounded Entity Overlap	y	4	Precision and recall for grounded named entities.
3	Embedding-based Matching	y	16	Minimized matching distance with LINE vectors.
4	Wikipedia Evidence	n	2	Precision and recall for evidence from Wikipedia.
Baselines				
5	TF-IDF	y	2	The cosine distance with TF-IDF.
6	Ungrounded Mention Overlap	y	4	Precision and recall for ungrounded mentions.
7	Embedding-based Matching	y	4	Minimized matching distance with averaged vectors.
Total			33	

Table 2: A list of all features used in the experiments. The third column indicates whether the corresponding feature is extracted from the full articles. If not, it's extracted only from the headlines and lead paragraphs.

We use each mention's text span as an ungrounded mention, and its corresponding Wikipedia ID as a grounded entity. For instance, in Table 8, the detected text span *Westminster* is an ungrounded mention, and it's grounded to the entry *Parliament of the United Kindom*.

4.2 Selecting Candidates

Given a citing context, we construct its candidate article set with the top 200 articles retrieved by TF-IDF distance. In the experiments, approximately 92.61% of the ground truth cited articles appear in the candidate sets. We discard those that do not. We further randomly split the remaining 33318 pairs into training/validation/test sets with the proportion of 3:1:1.

For each training pair, we randomly sample 5 articles from its candidate article set (excluding ground truth) and pair them with the citing context as negative samples. According to Tan et al. (2015), the number of negative samples does not significantly affect the linear learning to rank model's performance. During validation and testing, all of the 200 candidates are taken into account.

4.3 Experimental Setup

In the experiments, we set the TF-IDF as the baseline, and incrementally add different groups of features to the system.

The word embedding is pretrained with skip-gram model (Mikolov et al., 2013) on Wikipedia corpus and then fine-tuned using the method proposed in Wieting et al. (2015) on PPDB (Ganitkevitch et al., 2013). The embedding fine-tuned with paraphrase pairs can better capture the semantic relatedness of different phrase. In the experiments, we observe a $1\% - 2\%$ improvement by the fine-tuned word representations compared to vanilla skip-gram vectors.

We use the linear model in RankLib[3] for the learning to rank implementation. Coordinate ascent is used for parameter optimization. The model is trained to directly optimize the evaluation metrics, Precision@1, Precision@5, NDCG@5 and MAP, respectively.

For NDCG@5 measure, we set a binary relevance score, i.e., the scores equal to 1 for ground truths, 0 for negative samples.

4.4 Experimental Results

Table 3 gives the performance of the baselines and the systems using different groups of features on test and validation sets. The results show that WMD brings a consistent improvement over its TF-IDF baseline, and so do grounded entities compared to ungrounded mentions.

Individually added to the TF-IDF baseline, WMD has the largest performance boost, followed by grounded entity features. Besides, the additional information from grounded entity knowledge helps the model outperform the ungrounded mentions, with a consistent margin of 1.0%-2.0% NDCG@5.

We further compare the performance of the models when using features from headlines+lead paragraphs only and those from full passages. As shown in Table 3, the former brings much better performance on each metric compared to the latter.

It's worth noting that there are ground truths mis-labeled as irrelevant in the dataset. A primary

[3] https://sourceforge.net/p/lemur/wiki/RankLib/

id	Features	Precision@1		Precision@5		NDCG@5		MAP	
		Test	Dev	Test	Dev	Test	Dev	Test	Dev
	Headline + Lead Para								
1	TF-IDF	42.61	42.21	19.84	19.78	52.72	52.22	53.50	53.06
2	+ Ungrounded Mentions	43.67	43.04	19.45	19.30	53.84	53.26	54.46	54.12
3	+ Grounded Entities	44.52	44.02	20.84	20.51	55.99	55.0	56.55	56.09
4	+ Ungrounded+Grounded	43.93	44.05	20.2	19.66	55.99	55.17	56.52	56.13
5	TF-IDF + WMD	45.94	45.84	21.11	21.62	57.20	57.50	58.12	58.34
6	+ Ungrounded Mentions	46.44	46.63	21.05	21.56	57.61	57.80	58.55	58.78
7	+ Grounded Entities	**47.63**	**47.5**	**21.96**	**22.07**	58.52	58.41	**60.01**	**59.83**
8	+ Ungrounded+Grounded	47.23	46.84	21.58	21.56	**59.01**	**58.91**	59.88	59.66
	Full Article								
9	TF-IDF	49.3	48.11	23.33	23.06	60.51	59.54	60.71	59.73
10	+ Ungrounded Mentions	50.46	50.42	23.81	23.67	61.97	61.73	62.6	61.94
11	+ Grounded Entities	51.42	50.27	23.91	23.78	63.26	62.09	63.23	62.15
12	+ Ungrounded+Grounded	51.46	50.23	23.85	23.74	62.94	62.48	63.15	63.02
13	TF-IDF + WMD	52.31	51.82	23.87	24.04	63.71	63.99	64.08	63.62
14	+ Ungrounded Mentions	53.26	53.3	23.98	24.16	64.57	64.29	64.52	64.37
15	+ Grounded Entities	**54.12**	**53.29**	24.37	24.05	65.29	64.48	65.32	64.53
16	+ Ungrounded+Grounded	54.04	53.21	**24.52**	**24.33**	**65.56**	**65.11**	**65.35**	**64.56**

Table 3: Experimental results in percentage on the dataset collected from Bing News.

id	Features	NDCG@5 on $\mathbf{S}$	NDCG@5 on $\tilde{\mathbf{S}}$
	Headline + Lead Para		
1	TF-IDF	52.77	56.28
2	+ Ungrounded Mentions	53.34	56.86
3	+ Grounded Entities	55.03	58.57
4	+ Ungrounded+Grounded	55.18	58.86
5	TF-IDF + WMD	56.51	60.13
6	+ Ungrounded Mentions	57.04	60.82
7	+ Grounded Entities	57.4	61.47
8	+ Ungrounded+Grounded	**58.05**	**61.78**

Table 4: Experimental results in percentage on $\mathbf{S}$ and $\tilde{\mathbf{S}}$. $\mathbf{S}$ is a randomly constructed subset of the test set, and $\tilde{\mathbf{S}}$ is obtained by manually labeling samples in $\mathbf{S}$.

reason is that news sites sometimes individually publish different reports on a certain event. And the articles don't necessarily share the same title. To see how this affects the model, we randomly build a subset $\mathbf{S}$ of the test set and manually label the selected samples, which gives $\tilde{\mathbf{S}}$[4]. Table 4 compares the model's performance on $\mathbf{S}$ and $\tilde{\mathbf{S}}$ under Headline+Lead paragraph setting. There is a consistent improvement of NDCG@5 score on $\tilde{\mathbf{S}}$ compared to that on $\mathbf{S}$. Besides that, on manually labeled data, the model's performance across different feature settings is almost in accord with that on the full test set. These results show that there are indeed mis-labeled ground truths in the dataset, but they have little influence when comparing different groups of features.

[4]Manually labeling all of the dev and test samples would be too time consuming, and we leave it to future work.

5 Analysis

In this section, we give detailed win-loss analysis for the models trained with NDCG@5 metric under headlines+lead paragraphs setting. Specifically, given two systems with different feature configurations, we compare their performance on each test sample. The results are shown as a heatmap in Figure 1. X and Y axises indicate the identifiers for each feature group, following those in Table 3. For example, the data point at $(5, 1)$ indicates that the inclusion of WMD brings better ranking scores to TF-IDF on 18.4% of the test samples; and as a trade off, it lowers the scores on 11.4% of the samples. We also observe that grounded entities brings gain to 15.9% of the samples, and loss for 9.6% of them. On average, two different groups of features disagree on 26.4% of the test samples.

We further give several mis-predictions by the model using certain groups of features, and illustrate how they are corrected by the inclusion of others (or the other way round). By mis-prediction, we mean that no ground truth cited article appears in the top 5 predictions of the returned list.

5.1 Dealing with Variety

Table 5 shows a mis-prediction by TF-IDF, but corrected after including WMD.

TF-IDF distance favors the high-score match-

ing keywords *approval* and *rating* between the citing context and mis-predicted article. On the other hand, distributed word representations factor the distances between word pairs, which helps to capture their semantic closeness, e.g., (*Argentines*, *Argentina*, Cosince distance: 0.210), (*poll*, *election*, 0.020), and (*increasingly*, *growing*, 0.286). WMD helps to bridge the vocabulary gap between the citing context and the cited article.

On the other hand, though not often, the use of distributed representation can also create mistakes. Table 6 gives an example where the inclusion of the WMD feature changes a correct prediction by TF-IDF into a mistake. By analyzing the WMD's transportation flow matrix **T**, we find that the used word embedding relates *MP* to *minister*, and *publicly* to *government*. More curiously, persons' names are very similar in its semantic space: (*Davies*, *Stephen*, 0.602), and (*Davies*, *Harper*, 0.635). A possible reason could be that both of the two names are very common, and thus the cooccurrence-based representation learning method is not able to distinguish them. This also justifies our use of grounded entities as additional information: from the Wikipedia description for entity *Stephen Harper*, the system might be able to find out that he actually serves in Canadian government, not in the UK's nor in the Welsh.

5.2 Dealing with Ambiguity

Entity grounding helps by resolving the ambiguity e.g., alias, abbreviation, of the entity mentions.

As shown in Table 7, *tiger* refers to *the mammal* in the ground truth pair. However, the same word refers to *Detroit Tigers the team* in the mis-predicted article. This ambiguity is resolved when the mention is grounded to its Wikipedia entry. In another example shown in Table 8, ungrounded mention *SNP*, though detected, contributes little to supporting the ground truth pair. However, when it's grounded to the entry *Scottish National Party*, the system leverages world knowledge and relates it to the mention of *Scotland* in the citing context.

The inclusion of grounded entity information may also lead to mistakes, many of which are due to the limited performance of the entity recognition and disambiguation system. We'd like to discuss another kind of error here, shown in Table 9. In the citing context, *The Daily Telegraph* is a newspaper published in the UK. It has little to do with the involved event except for reporting it. However, the system favors a farmers' story which

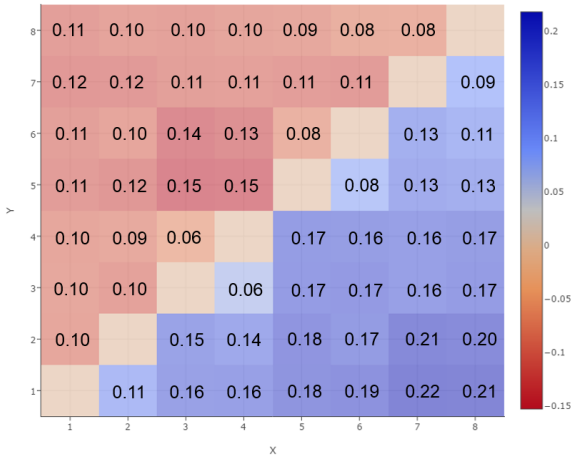


Figure 1: Heatmap for win-loss analysis results. Point (x, y) indicates how much feature x wins (loses if negative) against y.The X and Y axises indicate the identifiers for each feature group, following those in Table 3.

actually happened in the UK. We find that this contributes a lot to the system's errors when including grounded entities. We leave it to future work to figure out how to deal with this issue.

6 Related Work

This section reviews three lines of related work: (i) document recommendation, (ii) pharaphrase identification, (iii) question retrieval.

6.1 Document Recommendation

Existing literature mainly focuses on content-based methods and collaborative filtering (Adomavicius and Tuzhilin, 2005). There are studies trying to recommend documents based on citation contexts, either through identifying the motivations of the citations (Aya et al., 2005), or through the topical similarity (Ritchie, 2008; Ramachandran and Amir, 2007). On the other hand, Mcnee et al. (2002) leverage multiple information sources from authorship, paper-citation relations, and co-citations to recommend research papers.

Combining the context-based approaches and collaborative filtering, Torres et al. (2004) and Strohman et al. (2007) report better performance. Tang and Zhang (2009) use the Restricted Boltzmann Machine to model citations for placeholders, and Tan et al. (2015) integrate multiple features to recommend quotes for writings.

In the news domain, context-based approaches are presumably favorable due to the fact that the articles are relatively content-rich and citation-sparse. Previous studies manage to utilize information retrieval techniques to recommend news articles given a seed article (Yang et al., 2009; Bogers and van den Bosch, 2007).

Sides	–	Samples
citing	–	An earlier poll showed Argentines are also increasingly happy with her performance as President, putting her approval rating at almost 43%, up from 31% in September.
cited	Ground Truth	**Kirchner's Growing Popularity Could Skew Argentine Election** As Argentina gears up for a presidential election in October, the approval ratings of the current president, Cristina Kirchner, are improving and her rising reputation could affect the results of the election to replace her.
	Top-1 Prediction	**Bill Shorten's Approval Rating Falls in Wake of Royal Commission** The opposition leader gained approval from only 27% of the voters surveyed, while 52% disapproved.

Table 5: A mis-prediction by TF-IDF corrected by the inclusion of WMD.

Sides	–	Samples
citing	–	Chris Grayling was responding to a question from Gower Conservative MP Byron Davies about the regeneration investment fund for Wales "and the underselling of a large amount of publicly owned property".
cited	Ground Truth	**Wales land deal leaves taxpayers 15m short** A Welsh government spokesperson said there were conflicting valuations.
	Top-1 Prediction	**Conservative MP compares Stephen Harper government to Jesus, inspiring hilarious #CPCJesus tweets** Is it time we started referring to Prime Minister Stephen Harper as "Our Lord and Saviour"?

Table 6: A correct prediction by TF-IDF but then changes into a mistake when including WMD.

Sides	–	Samples
citing	–	Conservationists want the Bangladeshi government (Government of Bangladesh) to step up and help save the tigers through greater administration and enforcement of anti-poaching laws, as Bangladeshi (Bangladesh) does not legally protect tigers to the extent that other governments do, according to Inhabitat.
cited	Ground Truth	**Bangladeshi's (Bangladesh) abundant tiger population has collapsed to just 100** In Bangladeshi (Bangladesh), a new census shows that tiger populations in the Sundarbans (Sundarbans) mangrove forest are more endangered than ever. The study, which used hidden cameras to track and record tigers, provides a more accurate update than previous surveys that used other methods.
	Top-1 Prediction	**Detroit Tigers (Detroit Tigers) links: The Tigers (Detroit Tigers) are in trouble** After losing three straight games prior to All-Star break (Major League Baseball All-Star Game), the the Tigers (Detroit Tigers) don't have much more time to waste if they want to stay in contention.

Table 7: A mis-prediction by TF-IDF corrected by the inclusion of grounded entity features. The linked Wikipedia entries are indicated below the underlined entity mentions.

Sides	–	Samples
citing	–	With activities at Westminster (Parliament of the United Kingdom) challenging a narrow view of nationalism, and a planned charm offensive across the UK and Ireland (IrelandUnited Kingdom relations), it is that the party intends to significantly expand its reach beyond Scotland (Scotland).
cited	Ground Truth	**SNP (Scottish National Party) launches bid to extend influence beyond Scotland (Scotland)** First Minister of Scotland and SNP leader (Scottish National Party) Nicola Sturgeon (Nicola Sturgeon) worked hard to reassure voters in the election campaign.
	Top-1 Prediction	**Apple Pay (Apple Pay) UK (United Kingdom) launch confirmed for mid-July** Leaked documents from retailers suggest a launch date early next week.

Table 8: A mis-prediction by TF-IDF+ungrounded mention features corrected by the TF-IDF+grounded entity features. The linked Wikipedia entries are indicated below the underlined entity mentions.

Sides	–	Samples
citing	–	According to a UK Telegraph (United Kindom / The Daily Telegraph) report, the government is now forcing farmers and food manufacturers to sell anywhere from 30-100% of their products to the state , as opposed to stores and supermarkets.
cited	Ground Truth	**Venezuelan (Venezuela) farmers ordered to hand over produce to state** As Venezuela's (Venezuela) food shortages worsen, the president of the country's Food Industry (Food Industry) Chamber has said that authorities ordered producers of milk (milk), pasta, oil (oil), rice (rice), sugar (sugar) and flour to supply their products to the state stores.
	Top-1 Prediction	**Welsh (United Kindom) farmers launch #NoLambWeek price campaign** Fed-up Welsh (United Kindom) farmers are encouraging others to withhold their fat lambs for a week in protest at the current slump in the UK lamb trade.

Table 9: A correct prediction by TF-IDF but then changes into a mistake when including grounded entity features. The linked Wikipedia entries are indicated below the underlined entity mentions.

6.2 Paraphrase Identification

Several hand-crafted features have proven helpful in modeling sentence/phrase similarity, e.g., string-based overlap (Xu et al., 2014), machine translation measures (Madnani et al., 2012), and dependency syntax (Wan et al., 2006; Wang et al., 2015). Using the combination and discriminative re-weighting of the mentioned features, Ji and Eisenstein (2013) manage to obtain more competitive results.

More recent work has switched the focus onto neural methods. Socher et al. (2011) recursively encode the representations of sentences by the compositions of words. Convolutional neural nets (LeCun et al., 1998; Collobert and Weston, 2008) are also exploited in the tasks of paraphrase identification and sentence matching (Yin and Schütze, 2015; He et al., 2015; Hu et al., 2014).

Story link detection (SLD) is a similar task which aims to classify whether two news stories discuss the same event. Farahat et al. (2003) leverage part of speech tagging technique as well as task-specific similarity measures to boost the system's performance. Shah et al. (2006) show that entity based document representation is a better choice compared to word-based representations in SLD. In our scenario, the query is typically a piece of context sentence instead of an entire article. Therefore, we find that document level methods yield sub-optimal performance when used to model the similarity of citing context and the articles. Besides, due to the fact that there might be multiple reports for a single event, we consider it reasonable to formulate our problem into a ranking task instead of classification.

6.3 Question Retrieval

The key problem in question retrieval lies in modeling questions' similarity. Machine translation techniques (Jeon et al., 2005) and topic models (Duan et al., 2008) have been utilized by previous works. An alternative is representation learning. Zhou et al. (2015) use category-based meta-data to learn word embeddings. dos Santos et al. (2015) and Lei et al. (2016) obtain superior performance over hand-crafted features with CNN.

News articles are more well-written than most documents in QA communities, which results in the feasibility of high-quality entity detection and grounding.

7 Discussions

In this paper, we propose a novel problem of *news citation recommendation*, which aims to recommend news citations for references based on a citing context. We develop a re-ranking system leveraging implicit and explicit semantics for content similarity. We construct a real-world dataset. The experimental results show the efficacy of our approach.

8 Acknowledgments

This research is partially supported by National Basic Research Program of China under Grant No. 2015CB352201, National Natural Science Foundation of China under Grant No. 61502014, and China Post-doctoral Foundation under Grant No. 2015M580927.

References

Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.*, pages 734–749.

Selcuk Aya, Carl Lagoze, and Thorsten Joachims. 2005. Citation classification and its applications. In *Proceedings of the International Conference on Knowledge Management*, pages 287–298.

Toine Bogers and Antal van den Bosch. 2007. Comparing and evaluating information retrieval algorithms for news recommendation. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, pages 141–144.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *ICML*.

Cícero Nogueira dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *ACL-IJCNLP 2015*, pages 694–699.

Huizhong Duan, Yunbo Cao, Chin-Yew Lin, and Yong Yu. 2008. Searching questions by identifying question topic and question focus. In *ACL*, pages 156–164.

Ayman Farahat, Francine Chen, and Thorsten Brants. 2003. Optimizing story link detection is not equivalent to optimizing new event detection. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 232–239.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: the paraphrase database. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 758–764.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 84–90.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896.

Keivan Kianmehr, Shang Gao, Jawad Attari, M. Mushfiqur Rahman, Kofi Akomeah, Reda Alhajj, Jon Rokne, and Ken Barker. 2009. Text summarization techniques: Svm versus neural networks. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications Services*, pages 487–491.

Matt J. Kusner, Yu Sun, Nicholas I. Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *ICML*, pages 957–966.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Tao Lei, Hrishikesh Joshi, Regina Barzilay, Tommi S. Jaakkola, Kateryna Tymoshenko, Alessandro Moschitti, and Lluís Màrquez i Villodre. 2016. Semi-supervised question retrieval with gated convolutions. In *NAACL HLT 2016*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 879–888.

Nitin Madnani, Joel R. Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics*, pages 182–190.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Sean M. Mcnee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al M. Rashid, Joseph A. Konstan, and John Ried. 2002. On the recommending of citations for research papers. *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 116–125.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tao Qin, Xu-Dong Zhang, Ming-Feng Tsai, De-Sheng Wang, Tie-Yan Liu, and Hang Li. 2008. Query-level loss functions for information retrieval. *Inf. Process. Manage.*, 44:838–855.

Deepak Ramachandran and Eyal Amir. 2007. Bayesian Inverse Reinforcement Learning. *Proceedings of the 20th International Joint Conference on Artical Intelligence*, 51:2586–2591.

Anna Ritchie. 2008. *Citation Context Analysis for Information Retrieval*. Ph.D. thesis.

Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 1998. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*.

Chirag Shah, W. Bruce Croft, and David Jensen. 2006. Representing documents with named entities for story link detection (sld). In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 868–869.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Trevor Strohman, W. Bruce Croft, and David Jensen. 2007. Recommending citations for academic papers. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 705–706.

Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. 2015. Learning to recommend quotes for writing. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2453–2459.

Jie Tang and Jing Zhang. 2009. A discriminative approach to topic-based citation recommendation. In *Advances in Knowledge Discovery and Data Mining*, volume 5476, pages 572–579.

Xuewei Tang, Xiaojun Wan, and Xun Zhang. 2014. Cross-language context-aware citation recommendation in scientific articles. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 817–826.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. ACM.

R. Torres, S.M. McNee, M. Abel, J.A. Konstan, and J. Riedl. 2004. Enhancing digital libraries with techlens. In *Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/IEEE Conference on*, pages 228–236.

Stephen Wan, Mark Dras, Robert Dale, and Cecile Paris. 2006. Using dependency-based features to take the 'para-farce' out of paraphrase. In *Proceedings of the Australasian Language Technology Workshop 2006*, pages 131–138.

Xiaojun Wan. 2007. A novel document similarity measure based on earth mover's distance. *Inf. Sci.*, pages 3718–3730.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015. Syntax-based deep matching of short texts. In *IJCAI*, pages 1354–1361.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*, 3:345–358.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1192–1199.

Wei Xu, Alan Ritter, Chris Callison-Burch, William B. Dolan, and Yangfeng Ji. 2014. Extracting lexically divergent paraphrases from twitter. *TACL*, 2:435–448.

Yin Yang, Nilesh Bansal, Wisam Dakka, Panagiotis Ipeirotis, Nick Koudas, and Dimitris Papadias. 2009. Query by document. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 34–43.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Guangyou Zhou, Tingting He, Jun Zhao, and Po Hu. 2015. Learning continuous word embedding with metadata for question retrieval in community question answering. In *ACL-IJCNLP 2015*, pages 250–259.

Grapheme-to-Phoneme Models for (Almost) Any Language

Aliya Deri and Kevin Knight
Information Sciences Institute
Department of Computer Science
University of Southern California
{aderi, knight}@isi.edu

Abstract

Grapheme-to-phoneme (g2p) models are rarely available in low-resource languages, as the creation of training and evaluation data is expensive and time-consuming. We use Wiktionary to obtain more than 650k word-pronunciation pairs in more than 500 languages. We then develop phoneme and language distance metrics based on phonological and linguistic knowledge; applying those, we adapt g2p models for high-resource languages to create models for related low-resource languages. We provide results for models for 229 adapted languages.

1 Introduction

Grapheme-to-phoneme (g2p) models convert words into pronunciations, and are ubiquitous in speech- and text-processing systems. Due to the diversity of scripts, phoneme inventories, phonotactic constraints, and spelling conventions among the world's languages, they are typically language-specific. Thus, while most statistical g2p learning methods are language-agnostic, they are trained on language-specific data—namely, a pronunciation dictionary consisting of word-pronunciation pairs, as in Table 1.

Building such a dictionary for a new language is both time-consuming and expensive, because it requires expertise in both the language and a notation system like the International Phonetic Alphabet, applied to thousands of word-pronunciation pairs. Unsurprisingly, resources have been allocated only to the most heavily-researched languages. GlobalPhone, one of the most extensive multilingual text and speech databases, has pronunciation dictionaries in only 20 languages (Schultz et al., 2013)[1].

[1]We have been unable to obtain this dataset.

lang	word	pronunciation
eng	anybody	ẹ n iː b ɒ d iː
pol	żołądka	ʐ�o w o n̪ t̪ k a
ben	শাক	ʂ ɔ k ʈ ɔ
heb	חלומות	ʁ a l o m o t

Table 1: Examples of English, Polish, Bengali, and Hebrew pronunciation dictionary entries, with pronunciations represented with the International Phonetic Alphabet (IPA).

word	eng	deu	nld
gift	g ɪ f t^{h}	g ɪ f t	ɣ ɪ f t
class	k^{h} l æ s	k l aː s	k l ɑ s
send	s ẹ n d	z ɛ n t	s ɛ n t

Table 2: Example pronunciations of English words using English, German, and Dutch g2p models.

For most of the world's more than 7,100 languages (Lewis et al., 2009), no data exists and the many technologies enabled by g2p models are inaccessible.

Intuitively, however, pronouncing an unknown language should not necessarily require large amounts of language-specific knowledge or data. A native German or Dutch speaker, with no knowledge of English, can approximate the pronunciations of an English word, albeit with slightly different phonemes. Table 2 demonstrates that German and Dutch g2p models can do the same.

Motivated by this, we create and evaluate g2p models for low-resource languages by adapting existing g2p models for high-resource languages using linguistic and phonological information. To facilitate our experiments, we create several notable data resources, including a multilingual pronunciation dictionary with entries for more than 500 languages.

The contributions of this work are:

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 399–408,
Berlin, Germany, August 7-12, 2016.

- Using data scraped from Wiktionary, we clean and normalize pronunciation dictionaries for 531 languages. To our knowledge, this is the most comprehensive multilingual pronunciation dictionary available.
- We synthesize several named entities corpora to create a multilingual corpus covering 384 languages.
- We develop a language-independent distance metric between IPA phonemes.
- We extend previous metrics for language-language distance with additional information and metrics.
- We create two sets of g2p models for "high resource" languages: 97 simple rule-based models extracted from Wikipedia's "IPA Help" pages, and 85 data-driven models built from Wiktionary data.
- We develop methods for adapting these g2p models to related languages, and describe results for 229 adapted models.
- We release all data and models.

2 Related Work

Because of the severe lack of multilingual pronunciation dictionaries and g2p models, different methods of rapid resource generation have been proposed.

Schultz (2009) reduces the amount of expertise needed to build a pronunciation dictionary, by providing a native speaker with an intuitive rule-generation user interface. Schlippe et al. (2010) crawl web resources like Wiktionary for word-pronunciation pairs. More recently, attempts have been made to automatically extract pronunciation dictionaries directly from audio data (Stahlberg et al., 2016). However, the requirement of a native speaker, web resources, or audio data specific to the language still blocks development, and the number of g2p resources remains very low. Our method avoids these issues by relying only on text data from high-resource languages.

Instead of generating language-specific resources, we are instead inspired by research on cross-lingual automatic speech recognition (ASR) by Vu and Schultz (2013) and Vu et al. (2014), who exploit linguistic and phonetic relationships in low-resource scenarios. Although these works focus on ASR instead of g2p models and rely on audio data, they demonstrate that speech technology is portable across related languages.

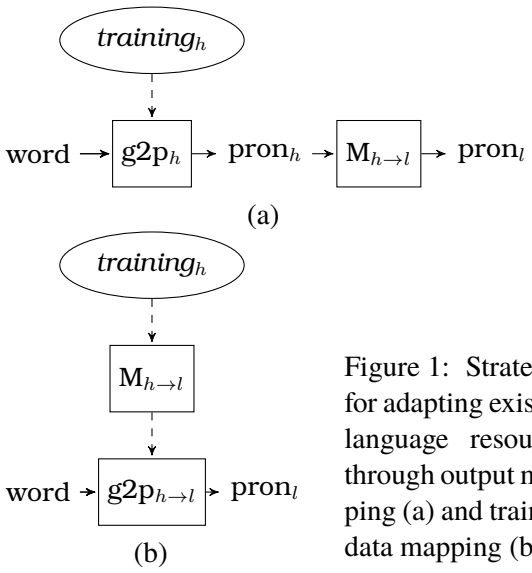


Figure 1: Strategies for adapting existing language resources through output mapping (a) and training data mapping (b).

3 Method

Given a low-resource language l without g2p rules or training data, we adapt resources (either an existing g2p model or a pronunciation dictionary) from a high-resource language h to create a g2p for l. We assume the existence of two modules: a phoneme-to-phoneme distance metric `phon2phon`, which allows us to map between the phonemes used by h to the phonemes used by l, and a closest language module `lang2lang`, which provides us with related language h.

Using these resources, we adapt resources from h to l in two different ways:

- Output mapping (Figure 1a): We use g2p$_h$ to pronounce word$_l$, then map the output to the phonemes used by l with `phon2phon`.
- Training data mapping (Figure 1b): We use `phon2phon` to map the pronunciations in h's pronunciation dictionary to the phonemes used by l, then train a g2p model using the adapted data.

The next sections describe how we collect data, create phoneme-to-phoneme and language-to-language distance metrics, and build high-resource g2p models.

4 Data

This section describes our data sources, which are summarized in Table 3.

4.1 Phoible

Phoible (Moran et al., 2014) is an online repository of cross-lingual phonological data. We use

<table>
<tr><td>Phoible</td><td>Wiki IPA Help tables</td><td colspan="2">Wiktionary</td></tr>
<tr><td>1674 languages
2155 lang. inventories
2182 phonemes
37 features</td><td rowspan="4">97 languages
24 scripts
1753 graph. segments
1534 phon. segments
3389 unique g-p rules</td><td colspan="2">531 languages
49 scripts
658k word-pron pairs</td></tr>
<tr><td>NE data</td><td>Wiktionary train</td><td>Wiktionary test</td></tr>
<tr><td>384 languages
36 scripts
9.9m NEs</td><td>85 languages
42 scripts
629k word-pron pairs</td><td>501 languages
45 scripts
26k word-pron pairs</td></tr>
</table>

Table 3: Summary of data resources obtained from Phoible, named entity resources, Wikipedia IPA Help tables, and Wiktionary. Note that, although our Wiktionary data technically covers over 500 languages, fewer than 100 include more than 250 entries (Wiktionary train).

two of its components: language phoneme inventories and phonetic features.

4.1.1 Phoneme inventories

A phoneme inventory is the set of phonemes used to pronounce a language, represented in IPA. Phoible provides 2156 phoneme inventories for 1674 languages. (Some languages have multiple inventories from different linguistic studies.)

4.1.2 Phoneme feature vectors

For each phoneme included in its phoneme inventories, Phoible provides information about 37 phonological features, such as whether the phoneme is nasal, consonantal, sonorant, or a tone. Each phoneme thus maps to a unique feature vector, with features expressed as `+`, `-`, or `0`.

4.2 Named Entity Resources

For our language-to-language distance metric, it is useful to have written text in many languages. The most easily accessible source of this data is multilingual named entity (NE) resources.

We synthesize 7 different NE corpora: Chinese-English names (Ji et al., 2009), Geonames (Vatant and Wick, 2006), JRC names (Steinberger et al., 2011), corpora from LDC[2], NEWS 2015 (Banchs et al., 2015), Wikipedia names (Irvine et al., 2010), and Wikipedia titles (Lin et al., 2011); to this, we also add multilingual Wikipedia titles for place names from an online English-language gazetteer (Everett-Heath, 2014). This yields a list of 9.9m named entities (8.9 not including English data) across 384 languages, which include the English translation, named entity type, and script information where possible.

4.3 Wikipedia IPA Help tables

To explain different languages' phonetic notations, Wikipedia users have created "IPA Help" pages,[3] which provide tables of simple grapheme examples of a language's phonemes. For example, on the English page, the phoneme z has the examples "**z**oo" and "ha**s**." We automatically scrape these tables for 97 languages to create simple grapheme-phoneme rules.

Using the `phon2phon` distance metric and mapping technique described in Section 5, we clean each table by mapping its IPA phonemes to the language's Phoible phoneme inventory, if it exists. If it does not exist, we map the phonemes to valid Phoible phonemes and create a phoneme inventory for that language.

4.4 Wiktionary pronunciation dictionaries

Ironically, to train data-driven g2p models for high-resource languages, and to evaluate our low-resource g2p models, we require pronunciation dictionaries for many languages. A common and successful technique for obtaining this data (Schlippe et al., 2010; Schlippe et al., 2012a; Yao and Kondrak, 2015) is scraping Wiktionary, an open-source multilingual dictionary maintained by Wikimedia. We extract unique word-pronunciation pairs from the English, German, Greek, Japanese, Korean, and Russian sites of Wiktionary. (Each Wiktionary site, while written in its respective language, contains word entries in multiple languages.)

[2]LDC2015E13, LDC2015E70, LDC2015E82, LDC2015E90, LDC2015E84, LDC2014E115, and LDC2015E91

[3]`https://en.wikipedia.org/wiki/Category:International_Phonetic_Alphabet_help`

Since Wiktionary data is very noisy, we apply length filtering as discussed by Schlippe et al. (2012b), as well as simple regular expression filters for HTML. We also map Wiktionary pronunciations to valid Phoible phonemes and language phoneme inventories, if they exist, as discussed in Section 5. This yields 658k word-pronunciation pairs for 531 languages. However, this data is not uniformly distributed across languages—German, English, and French account for 51% of the data.

We extract test and training data as follows: For each language with at least 1 word-pron pair with a valid word (at least 3 letters and alphabetic), we extract a test set of a maximum of 200 valid words. From the remaining data, for every language with 50 or more entries, we create a training set with the available data.

Ultimately, this yields a training set with 629k word-pronunciation pairs in 85 languages, and a test set with 26k pairs in 501 languages.

5 Phonetic Distance Metric

Automatically comparing pronunciations across languages is especially difficult in text form. Although two versions of the "sh" sound, "ʃ" and "ɕ," sound very similar to most people and very different from "m," to a machine all three characters seem equidistant.

Previous research (Özbal and Strapparava, 2012; Vu and Schultz, 2013; Vu et al., 2014) has addressed this issue by matching exact phonemes by character or manually selecting comparison features; however, we are interested in an automatic metric covering all possible IPA phoneme pairs.

We handle this problem by using Phoible's phoneme feature vectors to create `phon2phon`, a distance metric between IPA phonemes. In this section we also describe how we use this metric to clean open-source data and build phoneme-mapping models between languages.

5.1 `phon2phon`

As described in Section 4.1.2, each phoneme in Phoible maps to a unique feature vector; each feature value is `+`, `-`, or `0`, representing whether a feature is present, not present, or not applicable. (Tones, for example, can never be syllabic or stressed.)

We convert each feature vector into a bit representation by mapping each value to 3 bits. `+` to `110`, `-` to `101`, and `0` to `000`. This captures the idea that the features `+` and `-` are more similar than `0`.

lang	word	scraped	cleaned
ces	jód	ˈjoːd	j o d
pus	څلور	t͡saˈlor	t s a l o r
kan	ಭಾರತ	bhārata	b h a ɾ a ṭ a
hye	օդապարր	otʰaˈpɑɾ	o̤ t̤ʰ a p a l̤
ukr	тарган	tɑrˈɦɑn	t̪ a ɾ̪ h a n̪

Table 4: Examples of scraped and cleaned Wiktionary pronunciation data in Czech, Pashto, Kannada, Armenian, and Ukrainian.

Data: all phonemes P, scraped phoneme set S, language inventory T
Result: Mapping table M
initialize empty table M;
for p_s *in* S **do**
  if $p_s \notin P$ *and* ASCII$(p_s) \in P$ **then**
    $p_s = \text{ASCII}(p_s)$;
  end
  $p_p = \min_{\forall_{p_t \in T}} (\texttt{phon2phon}(p_s, p_t))$;
  add $p_s \rightarrow p_p$ to M;
end

Algorithm 1: A condensed version of our procedure for mapping scraped phoneme sets from Wikipedia and Wiktionary to Phoible language inventories. The full algorithm handles segmentation of the scraped pronunciation and heuristically promotes coverage of the Phoible inventory.

We then compute the normalized Hamming distance between every phoneme pair $p_{1,2}$ with feature vectors $f_{1,2}$ and feature vector length n as follows:

$$\texttt{phon2phon}(p_1, p_2) = \frac{\sum_{i=1}^{n} 1, \text{if} f_1^i \neq f_2^i}{n}$$

5.2 Data cleaning

We now combine `phon2phon` distances and Phoible phoneme inventories to map phonemes from scraped Wikipedia IPA help tables and Wiktionary pronunciation dictionaries to Phoible phonemes and inventories. We describe a condensed version of our procedure in Algorithm 1, and provide examples of cleaned Wiktionary output in Table 4.

5.3 Phoneme mapping models

Another application of `phon2phon` is to transform pronunciations in one language to another language's phoneme inventory. We can do this by

lang	avg	phon	script
English	German	Latin	French
Hindi	Gujarati	Bengali	Sanskrit
Vietnamese	Indonesian	Sindhi	Polish

Table 5: Closest languages with Wikipedia versions, based on `lang2lang` averaged metrics, phonetic inventory distance, and script distance.

creating a single-state weighted finite-state transducer (wFST) W for input language inventory I and output language inventory O:

$$\forall_{p_i \in I, p_o \in O} W.\text{add}(p_i, p_o, 1 - \texttt{phon2phon}(p_i, p_o))$$

W can then be used to map a pronunciation to a new language; this has the interesting effect of modeling accents by foreign-language speakers: *think* in English (pronounced "θ ɪ ŋ k^h") becomes "s̪ ɛ ŋ k" in German; the capital city *Dhaka* (pronounced in Bengali with a voiced aspirated "d̪ʱ") becomes the unaspirated "d æ k^h æ" in English.

6 Language Distance Metric

Since we are interested in mapping high-resource languages to low-resource related languages, an important subtask is finding the related languages of a given language.

The URIEL Typological Compendium (Littell et al., 2016) is an invaluable resource for this task. By using features from linguistic databases (including Phoible), URIEL provides 5 distance metrics between languages: genetic, geographic, composite (a weighted composite of genetic and geographic), syntactic, and phonetic. We extend URIEL by adding two additional metrics, providing averaged distances over all metrics, and adding additional information about resources. This creates `lang2lang`, a table which provides distances between and information about 2,790 languages.

6.1 Phoneme inventory distance

Although URIEL provides a distance metric between languages based on Phoible features, it only takes into account broad phonetic features, such as whether each language has voiced plosives. This can result in some non-intuitive results: based on this metric, there are almost 100 languages phonetically equivalent to the South Asian language Gujarati, among them Arawak and Chechen.

To provide a more fine-grained phonetic distance metric, we create a phoneme inventory distance metric using `phon2phon`. For each pair of language phoneme inventories $L_{1,2}$ in Phoible, we compute the following:

$$d(L_1, L_2) = \sum_{p_1 \in L_1} \min_{p_2 \in L_2} (\texttt{phon2phon}(p_1, p_2))$$

and normalize by dividing by $\sum_i d(L_1, L_i)$.

6.2 Script distance

Although Urdu is very similar to Hindi, its different alphabet and writing conventions would make it difficult to transfer an Urdu g2p model to Hindi. A better candidate language would be Nepali, which shares the Devanagari script, or even Bengali, which uses a similar South Asian script. A metric comparing the character sets used by two languages is very useful for capturing this relationship.

We first use our multilingual named entity data to extract character sets for the 232 languages with more than 500 NE pairs; then, we note that Unicode character names are similar for linguistically related scripts. This is most notable in South Asian scripts: for example, the Bengali ক, Gujarati ક, and Hindi क have Unicode names `BENGALI LETTER KA`, `GUJARATI LETTER KA`, and `DEVANAGARI LETTER KA`, respectively.

We remove script, accent, and form identifiers from the Unicode names of all characters in our character sets, to create a set of reduced character names used across languages. Then we create a binary feature vector f for every language, with each feature indicating the language's use of a reduced character (like `LETTER KA`). The distance between two languages $L_{1,2}$ can then be computed with a spatial cosine distance:

$$d(L_1, L_2) = 1 - \frac{f_1 \cdot f_2}{\|f_1\|_2 \, \|f_2\|_2}$$

6.3 Resource information

Each entry in our `lang2lang` distance table also includes the following features for the second language: the number of named entities, whether it is in Europarl (Koehn, 2005), whether it has its own Wikipedia, whether it is primarily written in the same script as the first language, whether it has an IPA Help page, whether it is in our Wiktionary test set, and whether it is in our Wiktionary training set.

Table 5 shows examples of the closest languages to English, Hindi, and Vietnamese, according to different lang2lang metrics.

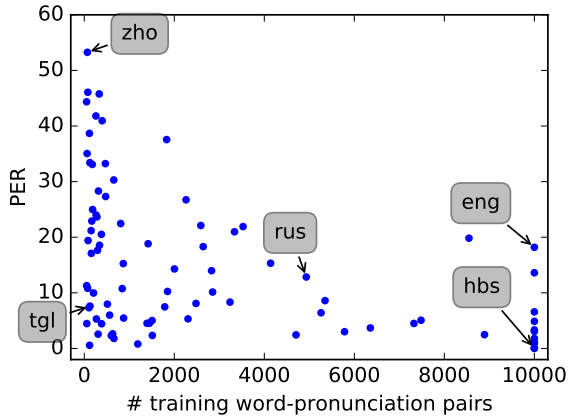


Figure 2: Training data size vs. PER for 85 models trained from Wiktionary. Labeled languages: English (eng), Serbo-Croatian (hbs), Russian (rus), Tagalog (tgl), and Chinese macrolanguage (zho).

7 Evaluation Metrics

The next two sections describe our high-resource and adapted g2p models. To evaluate these models, we compute the following metrics:

- % of words skipped: This shows the coverage of the g2p model. Some g2p models do not cover all character sequences. All other metrics are computed over non-skipped words.
- word error rate (WER): The percent of incorrect 1-best pronunciations.
- word error rate 100-best (WER 100): The percent of 100-best lists without the correct pronunciation.
- phoneme error rate (PER): The percent of errors per phoneme. A PER of 15.0 indicates that, on average, a linguist would have to edit 15 out of 100 phonemes of the output.

We then average these metrics across all languages (weighting each language equally).

8 High Resource g2p Models

We now build and evaluate g2p models for the "high-resource" languages for which we have either IPA Help tables or sufficient training data from Wiktionary. Table 6 shows our evaluation of these models on Wiktionary test data, and Table 7 shows results for individual languages.

8.1 IPA Help models

We first use the rules scraped from Wikipedia's IPA Help pages to build rule-based g2p models. We build a wFST for each language, with a path for each rule $g \rightarrow p$ and weight $w = 1/\text{count}(g)$. This method prefers rules with longer grapheme segments; for example, for the word *tin*, the output "ʃ n" is preferred over the correct "t^h ɪ n" because of the rule *ti*→ʃ. We build 97 IPA Help models, but have test data for only 91—some languages, like Mayan, do not have any Wiktionary entries.

As shown in Table 6, these rule-based models do not perform very well, suffering especially from a high percentage of skipped words. This is because IPA Help tables explain phonemes' relationships to graphemes, rather than vice versa. Thus, the English letter *x* is omitted, since its composite phonemes are better explained by other letters.

8.2 Wiktionary-trained models

We next build models for the 85 languages in our Wiktionary train data set, using the wFST-based Phonetisaurus (Novak et al., 2011) and MITLM (Hsu and Glass, 2008), as described by Novak et al (2012). We use a maximum of 10k pairs of training data, a 7-gram language model, and 50 iterations of EM.

These data-driven models outperform IPA Help models by a considerable amount, achieving a WER of 44.69 and PER of 15.06 averaged across all 85 languages. Restricting data to 2.5k or more training examples boosts results to a WER of 28.02 and PER of 7.20, but creates models for only 29 languages.

However, in some languages good results are obtained with very limited data; Figure 2 shows the varying quality across languages and data availability.

8.3 Unioned models

We also use our rule-based IPA Help tables to improve Wiktionary model performance. We accomplish this very simply, by prepending IPA help rules like the German *sch*→ʃ to the Wiktionary training data as word-pronunciation pairs, then running the Phonetisaurus pipeline.

Overall, the unioned g2p models outperform both the IPA help and Wiktionary models; however, as shown in Table 7, the effects vary across different languages. It is unclear what effect language characteristics, quality of IPA Help rules, and training data size have on unioned model improvement.

model	# langs	% skip	WER	WER 100	PER
ipa-help	91	21.49	78.13	59.18	35.36
wiktionary	85	4.78	44.69	23.15	15.06
unioned	85	3.98	44.17	21.97	14.70
ipa-help	56	22.95	82.61	61.57	35.51
wiktionary	56	3.52	40.28	20.30	13.06
unioned	56	2.31	39.49	18.51	12.52

Table 6: Results for high-resource models. The top portion of the table shows results for all models; the bottom shows results only for languages with both IPA Help and Wiktionary models.

lang	ben	tgl	tur	deu
# train	114	126	2.5k	10k
ipa-help	100.0	64.8	69.0	40.2
wikt	85.6	**34.2**	**39.0**	32.5
unioned	**66.2**	36.2	**39.0**	**24.5**

Table 7: WER scores for Bengali, Tagalog, Turkish, and German models. Unioned models with IPA Help rules tend to perform better than Wiktionary-only models, but not consistently.

9 Adapted g2p Models

Having created a set of high-resource models and our `phon2phon` and `lang2lang` metrics, we now explore different methods for adapting high-resource models and data for related low-resource languages. For comparable results, we restrict the set of high-resource languages to those covered by both our IPA Help and Wiktionary data.

9.1 No mapping

The simplest experiment is to run our g2p models on related low-resource languages, without adaptation. For each language l in our test set, we determine the top high-resource related languages $h_{1,2,...}$ according to the `lang2lang` averaged metric that have both IPA Help and Wiktionary data and the same script, not including the language itself. For IPA Help models, we choose the 3 most related languages $h_{1,2,3}$ and build a g2p model from their combined g-p rules. For Wiktionary and unioned models, we compile 5k words from the closest languages $h_{1,2,...}$ such that each h contributes no more than one third of the data (adding IPA Help rules for unioned models) and train a model from the combined data.

For each test word-pronunciation pair, we trivially map the word's letters to the characters used in $h_{1,2,...}$ by removing accents where necessary; we then use the high-resource g2p model to produce a pronunciation for the word. For example, our Czech IPA Help model uses a model built from g-p rules from Serbo-Croatian, Polish, and Slovenian; the Wiktionary and unioned models use data and rules from these languages and Latin as well.

This expands 56 g2p models (the languages covered by both IPA Help and Wiktionary models) to models for 211 languages. However, as shown in Table 8, results are very poor, with a very high WER of 92% using the unioned models and a PER of more than 50%. Interestingly, IPA Help models perform better than the unioned models, but this is primarily due to their high skip rate.

9.2 Output mapping

We next attempt to improve these results by creating a wFST that maps phonemes from the inventories of $h_{1,2...}$ to l (as described in Section 5.3). As shown in Figure 1a, by chaining this wFST to $h_{1,2...}$'s g2p model, we map the g2p model's output phonemes to the phonemes used by l. In each base model type, this process considerably improves accuracy over the no mapping approach; however, the IPA Help skip rate increases (Table 8).

9.3 Training data mapping

We now build g2p models for l by creating synthetic data for the Wiktionary and unioned models, as in Figure 1b. After compiling word-pronunciation pairs and IPA Help g-p rules from closest languages $h_{1,2,...}$, we then map the pronunciations to l and use the new pronunciations as training data. We again create unioned models by adding the related languages' IPA Help rules to the training data.

This method performs slightly worse in accuracy than output mapping, a WER of 87%, but has a much lower skip rate of 7%.

method	base model	# langs	% skip	WER	WER 100	PER
no mapping	ipa-help	211	12.46	**91.57**	**78.96**	**54.84**
	wikt	211	8.99	93.15	80.36	57.07
	unioned	211	**8.54**	92.38	79.26	57.21
output mapping	ipa-help	211	12.68	85.45	67.07	47.94
	wikt	211	15.00	86.48	66.20	46.84
	unioned	211	**11.72**	**84.82**	**63.63**	**46.25**
training data mapping	wikt	211	8.55	87.40	70.94	48.89
	unioned	211	**7.19**	**87.36**	**70.75**	**47.48**
rescripted	wikt	+10	15.94	**93.66**	81.76	**56.37**
	unioned	+10	**14.97**	94.45	**80.68**	57.35
final	wikt/unioned	229	6.77	88.04	69.80	48.01

Table 8: Results for adapted g2p models. Final adapted results (using the 85 languages covered by Wiktionary and unioned high-resource models, as well as rescripting) cover 229 languages.

lang	method	base model	rel langs	word	gold	hyp
eng	no mapping	ipa-help	deu, nld, swe	fuse	f j uː z	f ʏ s ɛ
arz	output mapping	unioned	fas, urd	بانجو	b æː n̪ g uː	b a n̪ d̪ ʃ uː
afr	training mapping	unioned	nld, lat, isl	dood	d ɔ t	d uː t
sah	training mapping	unioned	rus, bul, ukr	хатырык	k a t̪ ɯ r̪ ɯ k	k a t̪ i r̪ i k
kan	rescripted	unioned	hin, ben	ಡುಷ್ಟ	d̪ u ʂ ʈʰ a	d̪ uː ʂ ʈʰ
guj	rescripted	unioned	san, ben, hin	ક્રોએશિઆ	k ɾ o e ç ɪ a	k ɾ õː ə ʂ ɪ a

Table 9: Sample words, gold pronunciations, and hypothesis pronunciations for English, Egyptian Arabic, Afrikaans, Yakut, Kannada, and Gujarati.

9.4 Rescripting

Adaptation methods thus far have required that h and l share a script. However, this excludes languages with related scripts, like Hindi and Bengali.

We replicate our data mapping experiment, but now allow related languages $h_{1,2,...}$ with different scripts from l but a script distance of less than 0.2. We then build a simple "rescripting" table based on matching Unicode character names; we can then map not only h's pronunciations to l's phoneme set, but also h's word to l's script.

Although performance is relatively poor, rescripting adds 10 new languages, including Telugu, Gujarati, and Marwari.

9.5 Discussion

Table 8 shows evaluation metrics for all adaptation methods. We also show results using all 85 Wiktionary models (using unioned where IPA Help is available) and rescripting, which increases the total number of languages to 229. Table 9 provides examples of output with different languages.

In general, mapping combined with IPA Help rules in unioned models provides the best results. Training data mapping achieves similar scores as output mapping as well as a lower skip rate. Word skipping is problematic, but could be lowered by collecting g-p rules for the low-resource language.

Although the adapted g2p models make many individual phonetic errors, they nevertheless capture overall pronunciation conventions, without requiring language-specific data or rules. Specific points of failure include rules that do not exist in related languages (e.g., the silent "e" at the end of "fuse" and the conversion of "d̪ʃ" to "g" in Egyptian Arabic), mistakes in phoneme mapping, and overall "pronounceability" of the output.

9.6 Limitations

Although our adaptation strategies are flexible, several limitations prevent us from building a g2p model for any language. If there is not enough information about the language, our `lang2lang` table will not be able to provide related high-resource languages. Additionally, if the language's script is not closely related to another language's and thus cannot be rescripted (as with Thai and Armenian), we are not able to adapt related g2p data or models.

10 Conclusion

Using a large multilingual pronunciation dictionary from Wiktionary and rule tables from Wikipedia, we build high-resource g2p models and show that adding g-p rules as training data can improve g2p performance. We then leverage `lang2lang` distance metrics and `phon2phon` phoneme distances to adapt g2p resources for high-resource languages for 229 related low-resource languages. Our experiments show that adapting training data for low-resource languages outperforms adapting output. To our knowledge, these are the most broadly multilingual g2p experiments to date.

With this publication, we release a number of resources to the NLP community: a large multilingual Wiktionary pronunciation dictionary, scraped Wikipedia IPA Help tables, compiled named entity resources (including a multilingual gazetteer), and our `phon2phon` and `lang2lang` distance tables.[4]

Future directions for this work include further improving the number and quality of g2p models, as well as performing external evaluations of the models in speech- and text-processing tasks. We plan to use the presented data and methods for other areas of multilingual natural language processing.

11 Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments, as well as our colleagues Marjan Ghazvininejad, Jonathan May, Nima Pourdamghani, Xing Shi, and Ashish Vaswani for their advice. We would also like to thank Deniz Yuret for his invaluable help with data collection. This work was supported in part by DARPA (HR0011-15-C-0115) and ARL/ARO (W911NF-10-1-0533). Computation for the work described in this paper was supported by the University of Southern California's Center for High-Performance Computing.

References

Rafael E Banchs, Min Zhang, Xiangyu Duan, Haizhou Li, and A Kumaran. 2015. Report of NEWS 2015 machine transliteration shared task. In *Proc. NEWS Workshop*.

John Everett-Heath. 2014. *The Concise Dictionary of World Place-Names*. Oxford University Press, 2nd edition.

Bo-June Paul Hsu and James R Glass. 2008. Iterative language model estimation: efficient data structure & algorithms. In *Proc. Interspeech*.

Ann Irvine, Chris Callison-Burch, and Alexandre Klementiev. 2010. Transliterating from all languages. In *Proc. AMTA*.

Heng Ji, Ralph Grishman, Dayne Freitag, Matthias Blume, John Wang, Shahram Khadivi, Richard Zens, and Hermann Ney. 2009. Name extraction and translation for distillation. *Handbook of Natural Language Processing and Machine Translation: DARPA Global Autonomous Language Exploitation*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit*.

M Paul Lewis, Gary F Simons, and Charles D Fennig. 2009. *Ethnologue: Languages of the world*. SIL international, Dallas.

Wen-Pin Lin, Matthew Snover, and Heng Ji. 2011. Unsupervised language-independent name translation mining from Wikipedia infoboxes. In *Proc. Workshop on Unsupervised Learning in NLP*.

Patrick Littell, David Mortensen, and Lori Levin. 2016. URIEL. Pittsburgh: Carnegie Mellon University. `http://www.cs.cmu.edu/~dmortens/uriel.html`. Accessed: 2016-03-19.

Steven Moran, Daniel McCloy, and Richard Wright, editors. 2014. *PHOIBLE Online*. Max Planck Institute for Evolutionary Anthropology, Leipzig.

Josef R Novak, D Yang, N Minematsu, and K Hirose. 2011. Phonetisaurus: A WFST-driven phoneticizer. *The University of Tokyo, Tokyo Institute of Technology*.

Josef R Novak, Nobuaki Minematsu, and Keikichi Hirose. 2012. WFST-based grapheme-to-phoneme conversion: open source tools for alignment, model-building and decoding. In *Proc. International Workshop on Finite State Methods and Natural Language Processing*.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proc. ACL*.

Tim Schlippe, Sebastian Ochs, and Tanja Schultz. 2010. Wiktionary as a source for automatic pronunciation extraction. In *Proc. Interspeech*.

Tim Schlippe, Sebastian Ochs, and Tanja Schultz. 2012a. Grapheme-to-phoneme model generation for Indo-European languages. In *Proc. ICASSP*.

Tim Schlippe, Sebastian Ochs, Ngoc Thang Vu, and Tanja Schultz. 2012b. Automatic error recovery for pronunciation dictionaries. In *Proc. Interspeech*.

[4] Instructions for obtaining this data are available at the authors' websites.

Tanja Schultz, Ngoc Thang Vu, and Tim Schlippe. 2013. GlobalPhone: A multilingual text & speech database in 20 languages. In *Proc. ICASSP*.

Tanja Schultz. 2009. Rapid language adaptation tools and technologies for multilingual speech processing systems. In *Proc. IEEE Workshop on Automatic Speech Recognition*.

Felix Stahlberg, Tim Schlippe, Stephan Vogel, and Tanja Schultz. 2016. Word segmentation and pronunciation extraction from phoneme sequences through cross-lingual word-to-phoneme alignment. *Computer Speech & Language*, 35:234 – 261.

Ralf Steinberger, Bruno Pouliquen, Mijail Kabadjov, and Erik Van der Goot. 2011. JRC-Names: A freely available, highly multilingual named entity resource. In *Proc. Recent Advances in Natural Language Processing*.

Bernard Vatant and Marc Wick. 2006. Geonames ontology.
Online at http://www.geonames.org/ontology.

Ngoc Thang Vu and Tanja Schultz. 2013. Multilingual multilayer perceptron for rapid language adaptation between and across language families. In *Proc. Interspeech*.

Ngoc Thang Vu, David Imseng, Daniel Povey, Petr Motlicek Motlicek, Tanja Schultz, and Hervé Bourlard. 2014. Multilingual deep neural network based acoustic modeling for rapid language adaptation. In *Proc. ICASSP*.

Lei Yao and Grzegorz Kondrak. 2015. Joint generation of transliterations from multiple representations. In *Proc. NAACL HLT*.

Neural Word Segmentation Learning for Chinese

Deng Cai and **Hai Zhao***
Department of Computer Science and Engineering
Key Lab of Shanghai Education Commission
for Intelligent Interaction and Cognitive Engineering
Shanghai Jiao Tong University, Shanghai, China
thisisjcykcd@gmail.com, zhaohai@cs.sjtu.edu.cn

Abstract

Most previous approaches to Chinese word segmentation formalize this problem as a character-based sequence labeling task so that only contextual information within fixed sized local windows and simple interactions between adjacent tags can be captured. In this paper, we propose a novel neural framework which thoroughly eliminates context windows and can utilize complete segmentation history. Our model employs a gated combination neural network over characters to produce distributed representations of word candidates, which are then given to a long short-term memory (LSTM) language scoring model. Experiments on the benchmark datasets show that without the help of feature engineering as most existing approaches, our models achieve competitive or better performances with previous state-of-the-art methods.

1 Introduction

Most east Asian languages including Chinese are written without explicit word delimiters, therefore, word segmentation is a preliminary step for processing those languages. Since Xue (2003), most methods formalize the Chinese word segmentation (CWS) as a sequence labeling problem with character position tags, which can be handled with supervised learning methods such as Maximum Entropy (Berger et al., 1996; Low et al., 2005) and Conditional Random Fields (Lafferty et al., 2001; Peng et al., 2004; Zhao et al., 2006a). However, those methods heavily depend on the choice of handcrafted features.

Recently, neural models have been widely used for NLP tasks for their ability to minimize the effort in feature engineering. For the task of CWS, Zheng et al. (2013) adapted the general neural network architecture for sequence labeling proposed in (Collobert et al., 2011), and used character embeddings as input to a two-layer network. Pei et al. (2014) improved upon (Zheng et al., 2013) by explicitly modeling the interactions between local context and previous tag. Chen et al. (2015a) proposed a gated recursive neural network to model the feature combinations of context characters. Chen et al. (2015b) used an LSTM architecture to capture potential long-distance dependencies, which alleviates the limitation of the size of context window but introduced another window for hidden states.

Despite the differences, all these models are designed to solve CWS by assigning labels to the characters in the sequence one by one. At each time step of inference, these models compute the tag scores of character based on (i) context features within a fixed sized local window and (ii) tagging history of previous *one*.

Nevertheless, the tag-tag transition is insufficient to model the complicated influence from previous segmentation decisions, though it could sometimes be a crucial clue to later segmentation decisions. The fixed context window size, which is broadly adopted by these methods for feature engineering, also restricts the flexibility of modeling diverse distances. Moreover, word-level information, which is being the greater granularity unit as suggested in (Huang and Zhao, 2006), remains

*Corresponding author. This paper was partially supported by Cai Yuanpei Program (CSC No. 201304490199 and No. 201304490171), National Natural Science Foundation of China (No. 61170114 and No. 61272248), National Basic Research Program of China (No. 2013CB329401), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 409–420,
Berlin, Germany, August 7-12, 2016.

Models		Characters	Words	Tags
character based	(Zheng et al., 2013), ...	$c_{i-2}, c_{i-1}, c_i, c_{i+1}, c_{i+2}$	-	$t_{i-1}t_i$
	(Chen et al., 2015b)	$c_0, c_1, \ldots, c_i, c_{i+1}, c_{i+2}$	-	$t_{i-1}t_i$
word based	(Zhang and Clark, 2007), ...	c in w_{j-1}, w_j, w_{j+1}	w_{j-1}, w_j, w_{j+1}	-
	Ours	$c_0, c_1, \ldots, c_i$	$w_0, w_1, \ldots, w_j$	-

Table 1: Feature windows of different models. $i(j)$ indexes the current character(word) that is under scoring.

unemployed.

To alleviate the drawbacks inside previous methods and release those inconvenient constrains such as the fixed sized context window, this paper makes a latest attempt to re-formalize CWS as a direct segmentation learning task. Our method does not make tagging decisions on individual characters, but directly evaluates the relative likelihood of different segmented sentences and then search for a segmentation with the highest score. To feature a segmented sentence, a series of distributed vector representations (Bengio et al., 2003) are generated to characterize the corresponding word candidates. Such a representation setting makes the decoding quite different from previous methods and indeed much more challenging, however, more discriminative features can be captured.

Though the vector building is word centered, our proposed scoring model covers all three processing levels from character, word until sentence. First, the distributed representation starts from character embedding, as in the context of word segmentation, the n-gram data sparsity issue makes it impractical to use word vectors immediately. Second, as the word candidate representation is derived from its characters, the inside character structure will also be encoded, thus it can be used to determine the word likelihood of its own. Third, to evaluate how a segmented sentence makes sense through word interacting, an LSTM (Hochreiter and Schmidhuber, 1997) is used to chain together word candidates incrementally and construct the representation of partially segmented sentence at each decoding step, so that the coherence between next word candidate and previous segmentation history can be depicted.

To our best knowledge, our proposed approach to CWS is the first attempt which explicitly models the entire contents of the segmenter's state, including the complete history of both segmentation decisions and input characters. The comparisons of feature windows used in different models are shown in Table 1. Compared to both sequence labeling schemes and word-based models in the past, our model thoroughly eliminates context windows and can capture the complete history of segmentation decisions, which offers more possibilities to effectively and accurately model segmentation context.

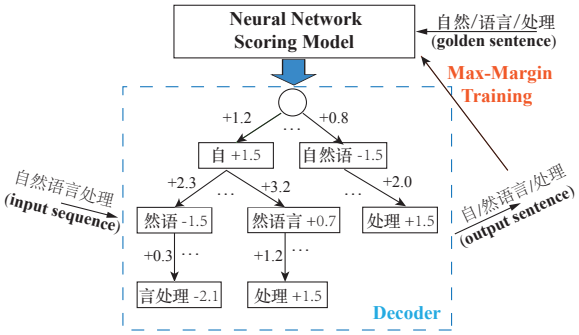


Figure 1: Our framework.

2 Overview

We formulate the CWS problem as finding a mapping from an input character sequence x to a word sequence y, and the output sentence y^* satisfies:

$$y^* = \arg\max_{y \in \text{GEN}(x)} \left(\sum_{i=1}^{n} \text{score}(y_i | y_1, \cdots, y_{i-1}) \right)$$

where n is the number of word candidates in y, and GEN(x) denotes the set of possible segmentations for an input sequence x. Unlike all previous works, our scoring function is sensitive to the complete contents of partially segmented sentence.

As shown in Figure 1, to solve CWS in this way, a neural network scoring model is designed to evaluate the likelihood of a segmented sentence. Based on the proposed model, a decoder is developed to find the segmented sentence with the highest score. Meanwhile, a max-margin method is utilized to perform the training by comparing

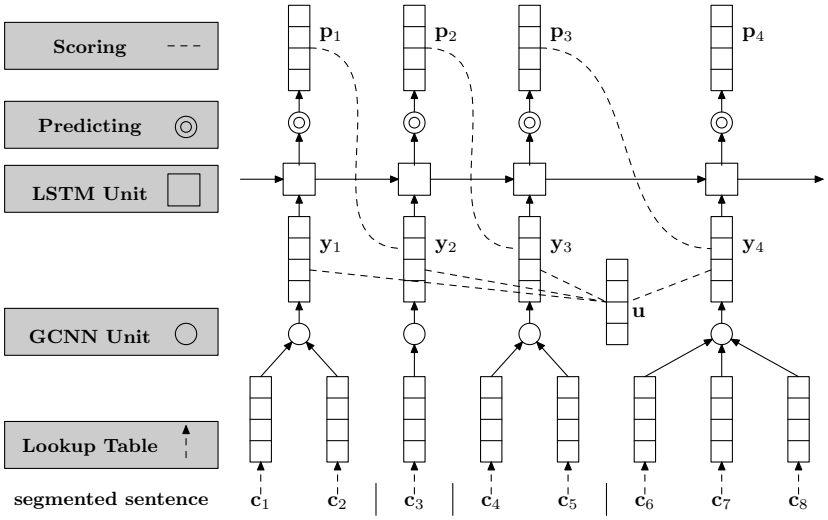


Figure 2: Architecture of our proposed neural network scoring model, where $\mathbf{c}_i$ denotes the i-th input character, $\mathbf{y}_j$ denotes the learned representation of the j-th word candidate, $\mathbf{p}_k$ denotes the prediction for the $(k+1)$-th word candidate and $\mathbf{u}$ is the trainable parameter vector for scoring the likelihood of individual word candidates.

the structured difference of decoder output and the golden segmentation. The following sections will introduce each of these components in detail.

3 Neural Network Scoring Model

The score for a segmented sentence is computed by first mapping it into a sequence of word candidate vectors, then the scoring model takes the vector sequence as input, scoring on each word candidate from two perspectives: (1) how likely the word candidate itself can be recognized as a legal word; (2) how reasonable the link is for the word candidate to follow previous segmentation history immediately. After that, the word candidate is appended to the segmentation history, updating the state of the scoring system for subsequent judgements. Figure 2 illustrates the entire scoring neural network.

3.1 Word Score

Character Embedding. While the scores are decided at the word-level, using word embedding (Bengio et al., 2003; Wang et al., 2016) immediately will lead to a remarkable issue that rare words and out-of-vocabulary words will be poorly estimated (Kim et al., 2015). In addition, the character-level information inside an n-gram can be helpful to judge whether it is a true word. Therefore, a lookup table of character embeddings is used as the bottom layer.

Formally, we have a character dictionary D of size $|D|$. Then each character $c \in D$ is represented as a real-valued vector (character embedding) $\mathbf{c} \in \mathbb{R}^d$, where d is the dimensionality of the vector space. The character embeddings are then stacked into an embedding matrix $\mathbf{M} \in \mathbb{R}^{d \times |D|}$. For a character $c \in D$, its character embedding $\mathbf{c} \in \mathbb{R}^d$ is retrieved by the embedding layer according to its index.

Gated Combination Neural Network. In order to obtain word representation through its characters, in the simplest strategy, character vectors are integrated into their word representation using a weight matrix $\mathbf{W}^{(L)}$ that is shared across all words with the same length L, followed by a non-linear function $g(\cdot)$. Specifically, $\mathbf{c}_i$ $(1 \leq i \leq L)$ are d-dimensional character vector representations respectively, the corresponding word vector $\mathbf{w}$ will be d-dimensional as well:

$$\mathbf{w} = g(\mathbf{W}^{(L)} \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_L \end{bmatrix}) \quad (1)$$

where $\mathbf{W}^{(L)} \in \mathbb{R}^{d \times Ld}$ and g is a non-linear function as mentioned above.

Although the mechanism above seems to work well, it can not sufficiently model the complicated combination features in practice, yet.

Gated structure in neural network can be useful for hybrid feature extraction according to (Chen et al., 2015a; Chung et al., 2014; Cho et al., 2014),

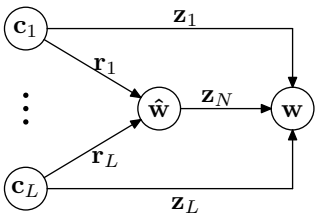


Figure 3: Gated combination neural network.

we therefore propose a gated combination neural network (GCNN) especially for character compositionality which contains two types of gates, namely *reset gate* and *update gate*. Intuitively, the reset gates decide which part of the character vectors should be mixed while the update gates decide what to preserve when combining the characters information. Concretely, for words with length L, the word vector $\mathbf{w} \in \mathbb{R}^d$ is computed as follows:

$$\mathbf{w} = \mathbf{z}_N \odot \hat{\mathbf{w}} + \sum_{i=1}^{L} \mathbf{z}_i \odot \mathbf{c}_i$$

where $\mathbf{z}_N, \mathbf{z}_i\ (1 \leq i \leq L)$ are update gates for new activation $\hat{\mathbf{w}}$ and governed characters respectively, and $\odot$ indicates element-wise multiplication.

The new activation $\hat{\mathbf{w}}$ is computed as:

$$\hat{\mathbf{w}} = \tanh(\mathbf{W}^{(L)} \begin{bmatrix} \mathbf{r}_1 \odot \mathbf{c}_1 \\ \vdots \\ \mathbf{r}_L \odot \mathbf{c}_L \end{bmatrix})$$

where $\mathbf{W}^{(L)} \in \mathbb{R}^{d \times Ld}$ and $\mathbf{r}_i \in \mathbb{R}^d\ (1 \leq i \leq L)$ are the reset gates for governed characters respectively, which can be formalized as:

$$\begin{bmatrix} \mathbf{r}_1 \\ \vdots \\ \mathbf{r}_L \end{bmatrix} = \sigma(\mathbf{R}^{(L)} \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_L \end{bmatrix})$$

where $\mathbf{R}^{(L)} \in \mathbb{R}^{Ld \times Ld}$ is the coefficient matrix of reset gates and σ denotes the sigmoid function.

The update gates can be formalized as:

$$\begin{bmatrix} \mathbf{z}_N \\ \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_L \end{bmatrix} = \exp(\mathbf{U}^{(L)} \begin{bmatrix} \hat{\mathbf{w}} \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_L \end{bmatrix}) \odot \begin{bmatrix} 1/\mathbf{Z} \\ 1/\mathbf{Z} \\ \vdots \\ 1/\mathbf{Z} \end{bmatrix}$$

where $\mathbf{U}^{(L)} \in \mathbb{R}^{(L+1)d \times (L+1)d}$ is the coefficient matrix of update gates, and $\mathbf{Z} \in \mathbb{R}^d$ is the normalization vector,

$$\mathbf{Z}_k = \sum_{i=1}^{L} [\exp(\mathbf{U}^{(L)} \begin{bmatrix} \hat{\mathbf{w}} \\ \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_L \end{bmatrix})]_{d \times i + k}$$

where $0 \leq k < d$.

According to the normalization condition, the update gates are constrained by:

$$\mathbf{z}_N + \sum_{i=1}^{L} \mathbf{z}_i = \mathbf{1}$$

The gated mechanism is capable of capturing both character and character interaction characteristics to give an efficient word representation (See Section 6.3).

Word Score. Denote the learned vector representations for a segmented sentence y with $[\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n]$, where n is the number of word candidates in the sentence. *word score* will be computed by the dot products of vector $\mathbf{y}_i (1 \leq i \leq n)$ and a trainable parameter vector $\mathbf{u} \in \mathbb{R}^d$.

$$\text{Word\_Score}(\mathbf{y}_i) = \mathbf{u} \cdot \mathbf{y}_i \tag{2}$$

It indicates how likely a word candidate by itself is to be a true word.

3.2 Link Score

Inspired by the recurrent neural network language model (RNN-LM) (Mikolov et al., 2010; Sundermeyer et al., 2012), we utilize an LSTM system to capture the coherence in a segmented sentence.

Long Short-Term Memory Networks. The LSTM neural network (Hochreiter and Schmidhuber, 1997) is an extension of the recurrent neural network (RNN), which is an effective tool for sequence modeling tasks using its hidden states for history information preservation. At each time step t, an RNN takes the input $\mathbf{x}_t$ and updates its recurrent hidden state $\mathbf{h}_t$ by

$$\mathbf{h}_t = g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b})$$

where g is a non-linear function.

Although RNN is capable, in principle, to process arbitrary-length sequences, it can be difficult to train an RNN to learn long-range dependencies due to the vanishing gradients. LSTM addresses

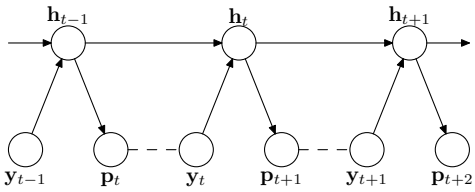


Figure 4: Link scores (dashed lines).

this problem by introducing a memory cell to preserve states over long periods of time, and controls the update of hidden state and memory cell by three types of gates, namely *input gate*, *forget gate* and *output gate*. Concretely, each step of LSTM takes input $\mathbf{x}_t, \mathbf{h}_{t-1}, \mathbf{c}_{t-1}$ and produces $\mathbf{h}_t, \mathbf{c}_t$ via the following calculations:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i\mathbf{x}_t + \mathbf{U}^i\mathbf{h}_{t-1} + \mathbf{b}^i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f\mathbf{x}_t + \mathbf{U}^f\mathbf{h}_{t-1} + \mathbf{b}^f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}^o\mathbf{x}_t + \mathbf{U}^o\mathbf{h}_{t-1} + \mathbf{b}^o) \\
\hat{\mathbf{c}}_t &= \tanh(\mathbf{W}^c\mathbf{x}_t + \mathbf{U}^c\mathbf{h}_{t-1} + \mathbf{b}^c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where $\sigma, \odot$ are respectively the element-wise sigmoid function and multiplication, $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t, \mathbf{c}_t$ are respectively the input gate, forget gate, output gate and memory cell activation vector at time t, all of which have the same size as hidden state vector $\mathbf{h}_t \in \mathbb{R}^H$.

Link Score. LSTMs have been shown to outperform RNNs on many NLP tasks, notably language modeling (Sundermeyer et al., 2012).

In our model, LSTM is utilized to chain together word candidates in a left-to-right, incremental manner. At time step t, a prediction $\mathbf{p}_{t+1} \in \mathbb{R}^d$ about next word $\mathbf{y}_{t+1}$ is made based on the hidden state $\mathbf{h}_t$:

$$\mathbf{p}_{t+1} = \tanh(\mathbf{W}^p\mathbf{h}_t + \mathbf{b}^p)$$

link score for next word $\mathbf{y}_{t+1}$ is then computed as:

$$\text{Link\_Score}(\mathbf{y}_{t+1}) = \mathbf{p}_{t+1} \cdot \mathbf{y}_{t+1} \qquad (3)$$

Due to the structure of LSTM, the prediction vector $\mathbf{p}_{t+1}$ carries useful information detected from the entire segmentation history, including previous segmentation decisions. In this way, our model gains the ability of sequence-level discrimination rather than local optimization.

3.3 Sentence score

Sentence score for a segmented sentence y with n word candidates is computed by summing up word scores (2) and link scores (3) as follow:

$$s(y_{[1:n]}, \theta) = \sum_{t=1}^{n}(\mathbf{u} \cdot \mathbf{y}_t + \mathbf{p}_t \cdot \mathbf{y}_t) \qquad (4)$$

where θ is the parameter set used in our model.

4 Decoding

The total number of possible segmented sentences grows exponentially with the length of character sequence, which makes it impractical to compute the scores of every possible segmentation. In order to get exact inference, most sequence-labeling systems address this problem with a Viterbi search, which takes the advantage of their hypothesis that the tag interactions only exist within adjacent characters (Markov assumption). However, since our model is intended to capture complete history of segmentation decisions, such dynamic programming algorithms can not be adopted in this situation.

Algorithm 1 Beam Search.

Input: model parameters θ
beam size k
maximum word length w
input character sequence $c[1:n]$
Output: Approx. k best segmentations

$\pi[0] \leftarrow \{(score = 0, \mathbf{h} = \mathbf{h}_0, \mathbf{c} = \mathbf{c}_0)\}$
for $i = 1$ to n **do**
  ▷ Generate Candidate Word Vectors
  $X \leftarrow \emptyset$
  for $j = \max(1, i - w)$ to i **do**
    $\mathbf{w} =$ GCNN-Procedure$(c[j:i])$
    X.add$((index = j - 1, word = \mathbf{w}))$
  end for
  ▷ Join Segmentation
  $Y \leftarrow \{\ y$.append$(x) \mid y \in \pi[x.index]$ and $x \in X\}$
  ▷ Filter k-Max
  $\pi[i] \leftarrow k\text{-}\arg\max_{y \in Y} y.score$
end for
return $\pi[n]$

To make our model efficient in practical use, we propose a beam-search algorithm with dynamic programming motivations as shown in Algorithm 1. The main idea is that any segmentation of the

first i characters can be separated as two parts, the first part consists of characters with indexes from 0 to j that is denoted as y, the rest part is the word composed by $c[j+1:i]$. The influence from previous segmentation y can be represented as a triple ($y.score$, $y.\mathbf{h}$, $y.\mathbf{c}$), where $y.score$, $y.\mathbf{h}$, $y.\mathbf{c}$ indicate the current score, current hidden state vector and current memory cell vector respectively. Beam search ensures that the total time for segmenting a sentence of n characters is $w \times k \times n$, where w, k are maximum word length and beam size respectively.

5 Training

We use the max-margin criterion (Taskar et al., 2005) to train our model. As reported in (Kummerfeld et al., 2015), the margin methods generally outperform both likelihood and perception methods. For a given character sequence $x^{(i)}$, denote the correct segmented sentence for $x^{(i)}$ as $y^{(i)}$. We define a structured margin loss $\Delta(y^{(i)}, \hat{y})$ for predicting a segmented sentence $\hat{y}$:

$$\Delta(y^{(i)}, \hat{y}) = \sum_{t=1}^{m} \mu \mathbf{1}\{y^{(i),t} \neq \hat{y}^t\}$$

where m is the length of sequence $x^{(i)}$ and μ is the discount parameter. The calculation of margin loss could be regarded as to count the number of incorrectly segmented characters and then multiple it with a fixed discount parameter for smoothing. Therefore, the loss is proportional to the number of incorrectly segmented characters.

Given a set of training set Ω, the regularized objective function is the loss function $J(\theta)$ including an ℓ_2 norm term:

$$J(\theta) = \frac{1}{|\Omega|} \sum_{(x^{(i)}, y^{(i)}) \in \Omega} l_i(\theta) + \frac{\lambda}{2} ||\theta||_2^2$$

$$l_i(\theta) = \max_{\hat{y} \in \text{GEN}(x^{(i)})} (s(\hat{y}, \theta) + \Delta(y^{(i)}, \hat{y}) - s(y^{(i)}, \theta))$$

where the function $s(\cdot)$ is the sentence score defined in equation (4).

Due to the hinge loss, the objective function is not differentiable, we use a subgradient method (Ratliff et al., 2007) which computes a gradient-like direction. Following (Socher et al., 2013), we use the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs to minimize the objective.

Character embedding size	$d = 50$
Hidden unit number	$H = 50$
Initial learning rate	$\alpha = 0.2$
Margin loss discount	$\mu = 0.2$
Regularization	$\lambda = 10^{-6}$
Dropout rate on input layer	$p = 0.2$
Maximum word length	$w = 4$

Table 2: Hyper-parameter settings.

The update for the i-th parameter at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}$$

where α is the initial learning rate and $g_{\tau,i} \in \mathbb{R}^{|\theta_i|}$ is the subgradient at time step τ for parameter θ_i.

6 Experiments

6.1 Datasets

To evaluate the proposed segmenter, we use two popular datasets, PKU and MSR, from the second International Chinese Word Segmentation Bakeoff (Emerson, 2005). These datasets are commonly used by previous state-of-the-art models and neural network models.

Both datasets are preprocessed by replacing the continuous English characters and digits with a unique token. All experiments are conducted with standard Bakeoff scoring program[1] calculating precision, recall, and F_1-score.

6.2 Hyper-parameters

Hyper-parameters of neural network model significantly impact on its performance. To determine a set of suitable hyper-parameters, we divide the training data into two sets, the first 90% sentences as training set and the rest 10% sentences as development set. We choose the hyper-parameters as shown in Table 2.

We found that the character embedding size has a limited impact on the performance as long as it is large enough. The size 50 is chosen as a good trade-off between speed and performance. The number of hidden units is set to be the same as the character embedding. Maximum word length determines the number of parameters in GCNN part and the time consuming of beam search, since the words with a length $l > 4$ are relatively rare,

[1] http://www.sighan.org/bakeoff2003/score

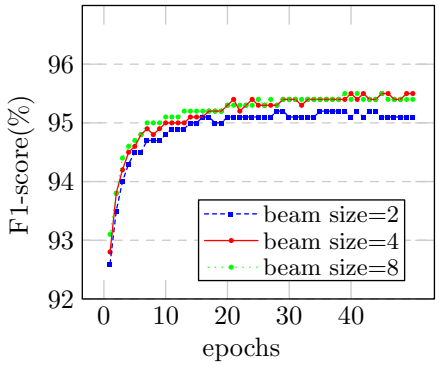


Figure 5: Performances of different beam sizes on PKU dataset.

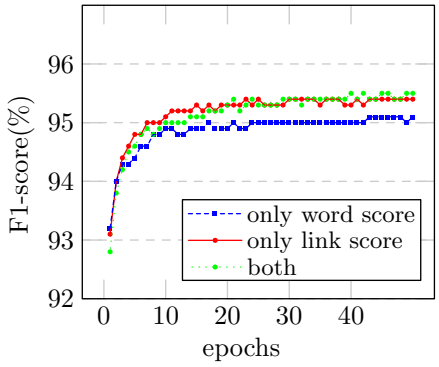


Figure 6: Performances of different score strategies on PKU dataset.

0.29% in PKU training data and 1.25% in MSR training data, we set the maximum word length to 4 in our experiments.[2]

Dropout is a popular technique for improving the performance of neural networks by reducing overfitting (Srivastava et al., 2014). We also drop the input layer of our model with dropout rate 20% to avoid overfitting.

6.3 Model Analysis

Beam Size. We first investigated the impact of beam size over segmentation performance. Figure 5 shows that a segmenter with beam size 4 is enough to get the best performance, which makes our model find a good balance between accuracy and efficiency.

GCNN. We then studied the role of GCNN in our model. To reveal the impact of GCNN, we re-implemented a simplified version of our model, which replaces the GCNN part with a single nonlinear layer as in equation (1). The results are listed in Table 3, which demonstrate that the performance is significantly boosted by exploiting the GCNN architecture (94.0% to 95.5% on F_1-score), while the best performance that the simplified version can achieve is 94.7%, but using a much larger character embedding size.

models	P	R	F
Single layer ($d = 50$)	94.3	93.7	94.0
GCNN ($d = 50$)	95.8	95.2	95.5
Single layer ($d = 100$)	94.9	94.4	94.7

Table 3: Performances of different models on PKU dataset.

	PKU		MSR	
+Dictionary	ours	theirs	ours	theirs
(Chen et al., 2015a)	94.9	**95.9**	95.8	96.2
(Chen et al., 2015b)	94.6	95.7	95.7	**96.4**
This work	**95.7**	-	**96.4**	-

Table 4: Comparison of using different Chinese idiom dictionaries.[3]

Link Score & Word Score. We conducted several experiments to investigate the individual effect of link score and word score, since these two types of scores are intended to estimate the sentence likelihood from two different perspectives: the semantic coherence between words and the existence of individual words. The learning curves of models with different scoring strategies are shown in Figure 6.

The model with only word score can be regarded as the situation that the segmentation decisions are made only based on local window information. The comparisons show that such a model gives moderate performance. By contrast, the model with only link score gives a much better performance close to the joint model, which demonstrates that the complete segmentation history, which can not be effectively modeled in previous schemes, possesses huge appliance value for word segmentation.

6.4 Results

[2]This 4-character limitation is just for consistence for both datasets. We are aware that it is a too strict setting, especially which makes additional performance loss in a dataset with larger average word length, i.e., MSR.

[3]The dictionary used in (Chen et al., 2015a; Chen et al., 2015b) is neither publicly released nor specified the exact source until now. We have to re-run their code using our selected dictionary to make a fair comparison. Our dictionary has been submitted along with this submission.

Models	PKU			MSR		
	P	R	F	P	R	F
(Zheng et al., 2013)	92.8	92.0	92.4	92.9	93.6	93.3
(Pei et al., 2014)	93.7	93.4	93.5	94.6	94.2	94.4
(Chen et al., 2015a)*	94.6	94.2	94.4	94.6	95.6	95.1
(Chen et al., 2015b) *	94.6	94.0	94.3	94.5	95.5	95.0
This work	**95.5**	**94.9**	**95.2**	**96.1**	**96.7**	**96.4**
+Pre-trained character embedding						
(Zheng et al., 2013)	93.5	92.2	92.8	94.2	93.7	93.9
(Pei et al., 2014)	94.4	93.6	94.0	95.2	94.6	94.9
(Chen et al., 2015a)*	94.8	94.1	94.5	94.9	95.9	95.4
(Chen et al., 2015b)*	95.1	94.4	94.8	95.1	96.2	95.6
This work	**95.8**	**95.2**	**95.5**	**96.3**	**96.8**	**96.5**

Table 5: Comparison with previous neural network models. Results with * are from our runs on their released implementations.[5]

Models	PKU	MSR	PKU	MSR
(Tseng et al., 2005)	95.0	96.4	-	-
(Zhang and Clark, 2007)	94.5	97.2	-	-
(Zhao and Kit, 2008b)	95.4	97.6	-	-
(Sun et al., 2009)	95.2	97.3	-	-
(Sun et al., 2012)	95.4	97.4	-	-
(Zhang et al., 2013)	-	-	96.1*	97.4*
(Chen et al., 2015a)	94.5	95.4	96.4*	97.6*
(Chen et al., 2015b)	94.8	95.6	96.5*	97.4*
This work	95.5	96.5	-	-

Table 6: Comparison with previous state-of-the-art models. Results with * used external dictionary or corpus.

We first compare our model with the latest neural network methods as shown in Table 4. However, (Chen et al., 2015a; Chen et al., 2015b) used an extra preprocess to filter out Chinese idioms according to an external dictionary.[4] Table 4 lists the results (F_1-scores) with different dictionaries, which show that our models perform better when under the same settings.

Table 5 gives comparisons among previous neural network models. In the first block of Table 5, the character embedding matrix $\mathbf{M}$ is randomly initialized. The results show that our proposed novel model outperforms previous neural network methods.

Previous works have found that the performance can be improved by pre-training the character embeddings on large unlabeled data. Therefore, we use word2vec (Mikolov et al., 2013) toolkit[6] to pre-train the character embeddings on the Chinese Wikipedia corpus and use them for initialization. Table 5 also shows the results with additional pre-trained character embeddings. Again, our model achieves better performance than previous neural network models do.

Table 6 compares our models with previous state-of-the-art systems. Recent systems such as (Zhang et al., 2013), (Chen et al., 2015b) and (Chen et al., 2015a) rely on both extensive feature engineering and external corpora to boost performance. Such systems are not directly comparable with our models. In the closed-set setting, our models can achieve state-of-the-art performance

[4]In detail, when a dictionary is used, a preprocess is performed before training and test, which scans original text to find out Chinese idioms included in the dictionary and replace them with a unique token. This treatment does not strictly follow the convention of closed-set setting defined by SIGHAN Bakeoff, as no linguistic resources, either dictionary or corpus, other than the training corpus, should be adopted.

[5]To make comparisons fair, we re-run their code (https://github.com/dalstonChen) without their unspecified Chinese idiom dictionary.

[6]http://code.google.com/p/word2vec/

Max. word length	F_1 score	Time (Days)
4	96.5	4
5	96.7	5
6	96.8	6

Table 7: Results on MSR dataset with different maximum decoding word length settings.

on PKU dataset but a competitive result on MSR dataset, which can attribute to too strict maximum word length setting for consistence as it is well known that MSR corpus has a much longer average word length (Zhao et al., 2010).

Table 7 demonstrates the results on MSR corpus with different maximum decoding word lengths, in which both F_1 scores and training time are given. The results show that the segmentation performance can indeed further be improved by allowing longer words during decoding, though longer training time are also needed. As 6-character words are allowed, F_1 score on MSR can be furthermore improved 0.3%.

For the running cost, we roughly report the current computation consuming on PKU dataset.[7] It takes about two days to finish 50 training epochs (for results in Figure 6 and the last row of Table 6) only with two cores of an Intel i7-5960X CPU. The requirement for RAM during training is less than 800MB. The trained model can be saved within 4MB on the hard disk.

7 Related Work

Neural Network Models. Most modern CWS methods followed (Xue, 2003) treated CWS as a sequence labeling problems (Zhao et al., 2006b). Recently, researchers have tended to explore neural network based approaches (Collobert et al., 2011) to reduce efforts of feature engineering (Zheng et al., 2013; Qi et al., 2014; Chen et al., 2015a; Chen et al., 2015b). They modeled CWS as tagging problem as well, scoring tags on individual characters. In those models, tag scores are decided by context information within local windows and the sentence-level score is obtained via *context-independently* tag transitions. Pei et al. (2014) introduced the tag embedding as input to capture the combinations of context and tag history. However, in previous works, only the tag of previous *one* character was taken into consideration though theoretically the complete history of actions taken by the segmenter should be considered.

Alternatives to Sequence Labeling. Besides sequence labeling schemes, Zhang and Clark (2007) proposed a word-based perceptron method. Zhang et al. (2012) used a linear-time incremental model which can also benefits from various kinds of features including word-based features. But both of them rely heavily on massive handcrafted features. Contemporary to this work, some neural models (Zhang et al., 2016a; Liu et al., 2016) also leverage word-level information. Specifically, Liu et al. (2016) use a semi-CRF taking segment-level embeddings as input and Zhang et al. (2016a) use a transition-based framework.

Another notable exception is (Ma and Hinrichs, 2015), which is also an embedding-based model, but models CWS as configuration-action matching. However, again, this method only uses the context information within limited sized windows.

Other Techniques. The proposed model might furthermore benefit from some techniques in recent state-of-the-art systems, such as semi-supervised learning (Zhao and Kit, 2008b; Zhao and Kit, 2008a; Sun and Xu, 2011; Zhao and Kit, 2011; Zeng et al., 2013; Zhang et al., 2013), incorporating global information (Zhao and Kit, 2007; Zhang et al., 2016b), and joint models (Qian and Liu, 2012; Li and Zhou, 2012).

8 Conclusion

This paper presents a novel neural framework for the task of Chinese word segmentation, which contains three main components: (1) a factory to produce word representation when given its governed characters; (2) a sentence-level likelihood evaluation system for segmented sentence; (3) an efficient and effective algorithm to find the best segmentation.

The proposed framework makes a latest attempt to formalize word segmentation as a direct structured learning procedure in terms of the recent distributed representation framework.

Though our system outputs results that are better than the latest neural network segmenters but comparable to all previous state-of-the-art systems, the framework remains a great of potential that can be further investigated and improved in the future.

[7] Our code is released at https://github.com/jcyk/CWS.

References

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

Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1744–1753.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1197–1206.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 133.

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

Chang-Ning Huang and Hai Zhao. 2006. Which is essential for chinese word segmentation: Character versus word. In *The 20th Pacific Asia Conference on Language, Information and Computation*, pages 1–12.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. An empirical analysis of optimization for max-margin nlp. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 273–279.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth Interntional Conference on Machine Learning*.

Zhongguo Li and Guodong Zhou. 2012. Unified dependency parsing of chinese morphological and syntactic structures. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1445–1454.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. *arXiv preprint arXiv:1604.05499*.

Jin Kiat Low, Hwee Tou Ng, and Wenyuan Guo. 2005. A maximum entropy approach to chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, volume 1612164, pages 448–455.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1733–1743.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *11th Annual Conference of the International Speech Communication Association*, pages 1045–1048.

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

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 293–303.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of the 20th international conference on Computational Linguistics*, page 562.

Yanjun Qi, Sujatha G Das, Ronan Collobert, and Jason Weston. 2014. Deep learning for character-based information extraction. In *Advances in Information Retrieval*, pages 668–674.

Xian Qian and Yang Liu. 2012. Joint chinese word segmentation, pos tagging and parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 501–511.

Nathan D Ratliff, J Andrew Bagnell, and Martin Zinkevich. 2007. (approximate) subgradient methods for structured prediction. In *International Conference on Artificial Intelligence and Statistics*, pages 380–387.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 455–465.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 970–979.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 56–64.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 253–262.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *13th Annual Conference of the International Speech Communication Association*.

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

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop on Chinese language Processing*, volume 171.

Peilu Wang, Yao Qian, Hai Zhao, Frank K. Soong, Lei He, and Ke Wu. 2016. Learning distributed word representations for bidirectional lstm recurrent neural network. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Xiaodong Zeng, Derek F. Wong, Lidia S. Chao, and Isabel Trancoso. 2013. Graph-based semi-supervised model for joint chinese word segmentation and part-of-speech tagging. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 770–779.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 840–847.

Kaixu Zhang, Maosong Sun, and Changle Zhou. 2012. Word segmentation on chinese mirco-blog data with a linear-time incremental model. In *Second CIPS-SIGHAN Joint Conference on Chinese Language Processing*, pages 41–46.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 311–321.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016a. Transition-based neural word segmentation. In *Proceedings of the 54nd Annual Meeting of the Association for Computational Linguistics*.

Zhiong Zhang, Hai Zhao, and Lianhui Qin. 2016b. Probabilistic graph-based dependency parsing with convolutional neural network. In *Proceedings of the 54nd Annual Meeting of the Association for Computational Linguistics*.

Hai Zhao and Chunyu Kit. 2007. Incorporating global information into supervised learning for chinese word segmentation. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 66–74.

Hai Zhao and Chunyu Kit. 2008a. Exploiting unlabeled text with different unsupervised segmentation criteria for chinese word segmentation. *Research in Computing Science*, 33:93–104.

Hai Zhao and Chunyu Kit. 2008b. Unsupervised segmentation helps supervised learning of character tagging for word segmentation and named entity recognition. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 106–111.

Hai Zhao and Chunyu Kit. 2011. Integrating unsupervised and supervised word segmentation: The role of goodness measures. *Information Sciences*, 181(1):163–183.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006a. An improved chinese word segmentation system with conditional random field. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, volume 1082117.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006b. Effective tag set selection in chinese word segmentation via conditional random field modeling. In *Proceedings of the 9th Pacific Association for Computational Linguistics*, volume 20, pages 87–94.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2010. A unified character-based tagging framework for chinese word segmentation. *ACM Transactions on Asian Language Information Processing*, 9(2):5.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657.

Transition-Based Neural Word Segmentation

Meishan Zhang[1] and **Yue Zhang**[2] and **Guohong Fu**[1]
1. School of Computer Science and Technology, Heilongjiang University, Harbin, China
2. Singapore University of Technology and Design
mason.zms@gmail.com,
yue_zhang@sutd.edu.sg,
ghfu@hotmail.com

Abstract

Character-based and word-based methods are two main types of statistical models for Chinese word segmentation, the former exploiting sequence labeling models over characters and the latter typically exploiting a transition-based model, with the advantages that word-level features can be easily utilized. Neural models have been exploited for character-based Chinese word segmentation, giving high accuracies by making use of external character embeddings, yet requiring less feature engineering. In this paper, we study a neural model for word-based Chinese word segmentation, by replacing the manually-designed discrete features with neural features in a word-based segmentation framework. Experimental results demonstrate that word features lead to comparable performances to the best systems in the literature, and a further combination of discrete and neural features gives top accuracies.

1 Introduction

Statistical word segmentation methods can be categorized character-based (Xue, 2003; Tseng et al., 2005) and word-based (Andrew, 2006; Zhang and Clark, 2007) approaches. The former casts word segmentation as a sequence labeling problem, using segmentation tags on characters to mark their relative positions inside words. The latter, in contrast, ranks candidate segmented outputs directly, extracting both character and full-word features.

An influential character-based word segmentation model (Peng et al., 2004; Tseng et al., 2005) uses B/I/E/S labels to mark a character as the beginning, internal (neither beginning nor end), end and only-character (both beginning and end) of a word, respectively, employing conditional random field (CRF) to model the correspondence between the input character sequence and output label sequence. For each character, features are extracted from a five-character context window and a two-label history window. Subsequent work explores different label sets (Zhao et al., 2006), feature sets (Shi and Wang, 2007) and semi-supervised learning (Sun and Xu, 2011), reporting state-of-the-art accuracies.

	character-based	word-based
discrete	Peng et al. (2004) Tseng et al. (2005)	Andrew (2006) Zhang and Clark (2007)
neural	Zheng et al. (2013) Chen et al. (2015b)	**this work**

Figure 1: Word segmentation methods.

Recently, neural network models have been investigated for the character tagging approach. The main idea is to replace manual discrete features with automatic real-valued features, which are derived automatically from distributed character representations using neural networks. In particular, convolution neural network[1] (Zheng et al., 2013), tensor neural network (Pei et al., 2014), recursive neural network (Chen et al., 2015a) and long-short-term-memory (LSTM) (Chen et al., 2015b) have been used to derive neural feature representations from input word sequences, which are fed into a CRF inference layer.

In this paper, we investigate the effectiveness of word embedding features for neural network segmentation using transition-based models. Since it is challenging to integrate word features to the CRF inference framework of the existing

[1]The term in this paper is used to denote the neural network structure with convolutional layers, which is different from the typical convolution neural network that has a pooling layer upon convolutional layers (Krizhevsky et al., 2012).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 421–431,
Berlin, Germany, August 7-12, 2016.

step	action	buffer($\cdots w_{-1}w_0$)	queue($c_0c_1\cdots$)
0	-	ϕ	中 国 ···
1	*SEP*	中	国 外 ···
2	*APP*	中国	外 企 ···
3	*SEP*	中国 外	企 业 ···
4	*APP*	中国 外企	业 务 ···
5	*SEP*	中国 外企 业	务 发 ···
6	*APP*	中国 外企 业务	发 展 ···
7	*SEP*	··· 业务 发	展 迅 速
8	*APP*	··· 业务 发展	迅 速
9	*SEP*	··· 发展 迅	速
10	*APP*	··· 发展 迅速	ϕ

Figure 2: Segmentation process of "中国 (Chinese) 外企 (foreign company) 业务 (business) 发展 (develop) 迅速 (quickly)".

character-based methods, we take inspiration from word-based discrete segmentation instead. In particular, we follow Zhang and Clark (2007), using the transition-based framework to decode a sentence from left-to-right incrementally, scoring partially segmented results using both character-level and word-level features. Beam-search is applied to reduce error propagation and large-margin training with early-update (Collins and Roark, 2004) is used for learning from inexact search.

We replace the discrete word and character features of Zhang and Clark (2007) with word and character embeddings, respectively, and change their linear model into a deep neural network. Following Zheng et al. (2013) and Chen et al. (2015b), we use convolution neural networks to achieve local feature combination and LSTM to learn global sentence-level features, respectively. The resulting model is a word-based neural segmenter that can leverage rich embedding features. Its correlation with existing work on Chinese segmentation is shown in Figure 1.

Results on standard benchmark datasets show the effectiveness of word embedding features for neural segmentation. Our method achieves state-of-the-art results without any preprocess based on external knowledge such as Chinese idioms of Chen et al. (2015a) and Chen et al. (2015b). We release our code under GPL for research reference.[2]

2 Baseline Discrete Model

We exploit the word-based segmentor of Zhang and Clark (2011) as the baseline system. It incrementally segments a sentence using a transition system, with a state holding a partially-segmented sentence in a buffer s and ordering the next incoming characters in a queue q. Given an input Chinese sentence, the buffer is initially empty and the queue contains all characters of the sentence, a sequence of transition actions are used to consume characters in the queue and build the output sentence in the buffer. The actions include:

- `Append` (APP), which removes the first character from the queue, and appends it to the last word in the buffer;
- `Separate` (SEP), which moves the first character of the queue onto the buffer as a new (sub) word.

Given the input sequence of characters "中国外企业务发展迅速" (The business of foreign company in China develops quickly), the correct output can be derived using action sequence "*SEP APP SEP APP SEP APP SEP APP SEP APP*", as shown in Figure 2.

Search. Based on the transition system, the decoder searches for an optimal action sequence for a given sentence. Denote an action sequence as $A = a_1 \cdots a_n$. We define the score of A as the total score of all actions in the sequence, which is computed by:

$$score(A) = \sum_{a \in A} score(a) = \sum_{a \in A} w \cdot f(s, q, a),$$

where w is the model parameters, f is a feature extraction function, s and q are the buffer and queue of a certain state before the action a is applied.

The feature templates are shown in Table 1, which are the same as Zhang and Clark (2011). These base features include three main source of information. First, characters in the front of the queue and the end of the buffer are used for scoring both *separate* and *append* actions (e.g. c_0). Second, words that are identified are used to guide *separate* actions (e.g. w_0). Third, relevant information of identified words, such as their lengths and first/last characters are utilized for additional features (e.g. $len(w_{-1})$).

We follow Zhang and Clark (2011) in using beam-search for decoding, shown in Algorith 1, where Θ is the set of model parameters. Initially the beam contains only the initial state. At each step, each state in the beam is extended by applying both *SEP* and *APP*, resulting in a set of new states, which are scored and ranked. The top *B* are

[2]https://github.com/SUTDNLP/NNTransitionSegmentor.

Feature templates	Action
$c_{-1}c_0$	*APP, SEP*
w_{-1}, $w_{-1}w_{-2}$, $w_{-1}c_0$, $w_{-2}len(w_{-1})$ $start(w_{-1})c_0$, $end(w_{-1})c_0$ $start(w_{-1})end(w_{-1})$, $end(w_{-2})end(w_{-1})$ $w_{-2}len(w_{-1})$, $len(w_{-2})w_{-1}$ w_{-1}, where $len(w_{-1}) = 1$	*SEP*

Table 1: Feature templates for the baseline model, where w_i denotes the word in the buffer, c_i denotes the character in the queue, as shown in Figure 2, $start(.)$, $end(.)$ and $len(.)$ denote the first, last character and length of a word, respectively.

Algorithm 1 Beam-search decoding.

function DECODE($c_1 \cdots c_n$, Θ)
  agenda $\leftarrow$ { (ϕ, $c_1 \cdots c_n$, score=0.0) }
  for *i* **in** $1 \cdots n$
    beam $\leftarrow$ { }
    for *cand* **in** *agenda*
      new $\leftarrow$ SEP(*cand*, c_i, Θ)
      ADDITEM(beam, new)
      new $\leftarrow$ APP(cand, c_i, Θ)
      ADDITEM(*beam*, *new*)
    agenda $\leftarrow$ TOP-B(*beam*, *B*)
  best $\leftarrow$ BESTITEM(*agenda*)
  $w_1 \cdots w_m \leftarrow$ EXTRACTWORDS(*best*)

used as the beam states for the next step. The same process replaces until all input character are processed, and the highest-scored state in the beam is taken for output. Online leaning with max-margin is used, which is given in section 4.

3 Transition-Based Neural Model

We use a neural network model to replace the discrete linear model for scoring transition action sequences. For better comparison between discrete and neural features, the overall segmentation framework of the baseline is used, which includes the incremental segmentation process, the beam-search decoder and the training process integrated with beam-search (Zhang and Clark, 2011). In addition, the neural network scorer takes the similar feature sources as the baseline, which includes character information over the input, word information of the partially constructed output, and the history sequence of the actions that have been applied so far.

The overall architecture of the neural scorer is shown in Figure 3. Given a certain state

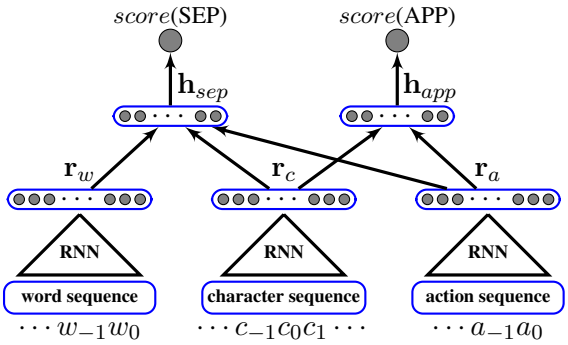


Figure 3: Scorer for the neural transition-based Chinese word segmentation model. We denote the last word in the buffer as w_0, the next incoming character as c_0 in the queue in consistent with Figure 2, and the last applied action as a_0.

configuration (s, q), we use three separate recurrent neural networks (RNN) to model the word sequence $\cdots w_{-1}w_0$, the character sequence $\cdots c_{-1}c_0c_1\cdots$, and the action sequence $\cdots a_{-1}a_0$, respectively, resulting in three dense real-valued vectors $\{\mathbf{r}_w, \mathbf{r}_c$ and $\mathbf{r}_a\}$, respectively. All the three feature vectors are used scoring the *SEP* action. For *APP*, on the other hand, we use only the character and action features $\mathbf{r}_c$ and $\mathbf{r}_a$ because the last word w_0 in the buffer is a partial word. Formally, given $\mathbf{r}_w, \mathbf{r}_c, \mathbf{r}_a$, the action scores are computed by:

$$score(\text{SEP}) = \mathbf{w}_{sep}\mathbf{h}_{sep}$$
$$score(\text{APP}) = \mathbf{w}_{app}\mathbf{h}_{app}$$

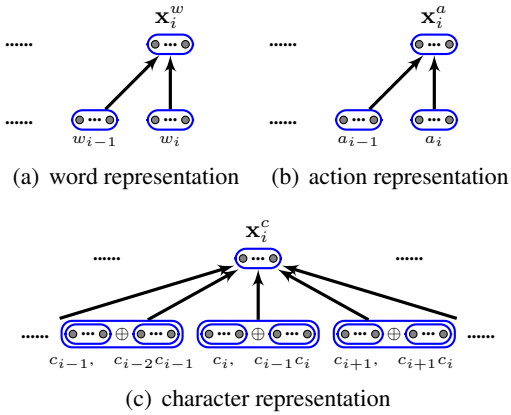
where

$$\mathbf{h}_{sep} = \tanh(W_{sep}[\mathbf{r}_w, \mathbf{r}_c, \mathbf{r}_a] + \mathbf{b}_{sep})$$
$$\mathbf{h}_{app} = \tanh(W_{app}[\mathbf{r}_c, \mathbf{r}_a] + \mathbf{b}_{app})$$

$W_{sep}, W_{app}, \mathbf{b}_{sep}, \mathbf{b}_{app}, \mathbf{w}_{sep}, \mathbf{w}_{app}$ are model parameters.

The neural networks take the embedding forms of words, characters and actions as input, for extracting $\mathbf{r}_w, \mathbf{r}_c$ and $\mathbf{r}_a$, respectively. We exploit the LSTM-RNN structure (Hochreiter and Schmidhuber, 1997), which can better capture non-local syntactic and semantic information from a sequential input, yet reducing gradient explosion or diminishing during training.

In general, given a sequence of input vectors $\mathbf{x}_0 \cdots \mathbf{x}_n$, the LSTM-RNN computes a sequence of hidden vectors $\mathbf{h}_0 \cdots \mathbf{h}_n$, respectively, with each $\mathbf{h}_i$ being determined by the input $\mathbf{x}_i$ and the previous hidden vector $\mathbf{h}_{i-1}$. A cell structure $\mathbf{ce}$ is



(a) word representation (b) action representation

(c) character representation

Figure 4: Input representations of LSTMS for $\mathbf{r}_a$ (actions) $\mathbf{r}_w$ (words) and $\mathbf{r}_c$ (characters).

used to carry long-term memory information over the history $\mathbf{h}_0 \cdots \mathbf{h}_i$ for calculating $\mathbf{h}_i$, and information flow is controlled by an input gate $\mathbf{ig}$, an output gate $\mathbf{og}$ and a forget gate $\mathbf{fg}$. Formally, the calculation of $\mathbf{h}_i$ using $\mathbf{h}_{i-1}$ and $\mathbf{x}_i$ is:

$$\begin{aligned}
\mathbf{ig}_i &= \sigma(W_{ig}\mathbf{x}_i + U_{ig}\mathbf{h}_{i-1} + V_{ig}\mathbf{ce}_{i-1} + \mathbf{b}_{ig}) \\
\mathbf{fg}_i &= \sigma(W_{fg}\mathbf{x}_i + U_{fg}\mathbf{h}_{i-1} + V_{fg}\mathbf{ce}_{i-1} + \mathbf{b}_{fg}) \\
\mathbf{ce}_i &= \mathbf{fg}_i \odot \mathbf{ce}_{i-1} + \\
&\qquad \mathbf{ig}_i \odot \tanh(W_{ce}\mathbf{x}_i + U_{ce}\mathbf{h}_{i-1} + \mathbf{b}_{ce}) \\
\mathbf{og}_i &= \sigma(W_{og}\mathbf{x}_i + U_{og}\mathbf{h}_{i-1} + V_{og}\mathbf{ce}_i + \mathbf{b}_{og}) \\
\mathbf{h}_i &= \mathbf{og}_i \odot \tanh(\mathbf{ce}_i),
\end{aligned}$$

where $U, V, W, \mathbf{b}$ are model parameters, and $\odot$ denotes Hadamard product.

When used to calculate $\mathbf{r}_w$, $\mathbf{r}_c$ and $\mathbf{r}_a$, the general LSTM structure above is given different input sequences $\mathbf{x}_0 \cdots \mathbf{x}_n$, according to the word, character and action sequences, respectively.

3.1 Input representation

Words. Given a word w, we use a looking-up matrix E_w to obtain its embedding $\mathbf{e}_w(w)$. The matrix can be obtained through pre-training on large size of auto segmented corpus. As shown in Figure 4(a), we use a convolutional neural layer upon a two-word window to obtain $\cdots \mathbf{x}_{-1}^w \mathbf{x}_0^w$ for the LSTM for $\mathbf{r}_w$, with the following formula:

$$\mathbf{x}_i^w = \tanh\left(W_w[\mathbf{e}_w(w_{i-1}), \mathbf{e}_w(w_i)] + \mathbf{b}_w\right)$$

Actions. We represent an action a with an embedding $\mathbf{e}_a(a)$ from a looking-up table E_a, and apply the similar convolutional neural network to obtain $\cdots \mathbf{x}_{-1}^a \mathbf{x}_0^a$ for $\mathbf{r}_a$, as shown in Figure 4(b). Given the input action sequence $\cdots a_{-1}a_0$, the $\mathbf{x}_i^a$ is computed by:

$$\mathbf{x}_i^a = \tanh\left(W_a[\mathbf{e}_a(a_{i-1}), \mathbf{e}_a(a_i)] + \mathbf{b}_a\right)$$

Characters. We make embeddings for both character unigrams and bigrams by looking-up matrixes E_c and E_{bc}, respectively, the latter being shown to be useful by Pei et al. (2014). For each character c_i, the unigram embedding $\mathbf{e}_c(c_i)$ and the bigram embedding $\mathbf{e}_{bc}(c_{i-1}c_i)$ are concatenated, before being given to a CNN with a convolution size of 5. For the character sequence $\cdots c_{-1}c_0c_1 \cdots$ of a given state (s, q), we compute its input vectors $\cdots \mathbf{x}_{-1}^c \mathbf{x}_0^c \mathbf{x}_1^c \cdots$ for the LSTM for $\mathbf{r}_c$ by:

$$\begin{aligned}
\mathbf{x}_i^c = \tanh\big(W_c[&\mathbf{e}_c(c_{i-2}) \oplus \mathbf{e}_{bc}(c_{i-3}c_{i-2}), \\
&\cdots, \mathbf{e}_c(c_i) \oplus \mathbf{e}_{bc}(c_{i-1}c_i), \cdots, \\
&\mathbf{e}_c(c_{i+2}) \oplus \mathbf{e}_{bc}(c_{i+1}c_{i+2})] + \mathbf{b}_c\big)
\end{aligned}$$

For all the above input representations, the looking-up tables E_w, E_a , E_c, E_{bc} and the weights W_w, W_a, W_c, $\mathbf{b}_w$, $\mathbf{b}_a$, $\mathbf{b}_c$ are model parameters. For calculating $\mathbf{r}_w$ and $\mathbf{r}_a$, we apply the LSTMs directly over the sequences $\cdots x_{-1}^w x_0^w$ and $\cdots x_{-1}^a x_0^a$ for words and actions, and use the outputs $\mathbf{h}_0^w$ and $\mathbf{h}_0^a$ for $\mathbf{r}_w$ and $\mathbf{r}_a$, respectively. For calculating $\mathbf{r}_c$, we further use a bi-directional extension of the original LSTM structure. In particular, the base LSTM is applied to the input character sequence both from left to right and from right to left, leading to two hidden node sequences $\cdots \mathbf{h}_{-1}^{cL}\mathbf{h}_0^{cL}\mathbf{h}_1^{cL} \cdots$ and $\cdots \mathbf{h}_{-1}^{cR}\mathbf{h}_0^{cR}\mathbf{h}_1^{cR} \cdots$, respectively. For the current character c_0, $\mathbf{h}_0^{cL}$ and $\mathbf{h}_0^{cR}$ are concatenated to form the final vector $\mathbf{r}_c$. This is feasible because the character sequence is input and static, and previous work has demonstrated better capability of bi-directional LSTM for modeling sequences (Yao and Zweig, 2015).

3.2 Integrating discrete features

Our model can be extended by integrating the baseline discrete features into the feature layer. In particular,

$$\begin{aligned}
score(\text{SEP}) &= \mathbf{w'}_{sep}(\mathbf{h}_{sep} \oplus f_{sep}) \\
score(\text{APP}) &= \mathbf{w'}_{app}(\mathbf{h}_{app} \oplus f_{app}),
\end{aligned}$$

where $\mathbf{f}_{sep}$ and $\mathbf{f}_{app}$ represent the baseline sparse vector for *SEP* and *APP* features, respectively, and $\oplus$ denotes the vector concatenation operation.

Algorithm 2 Max-margin training with early-update.

function TRAIN($c_1 \cdots c_n$, $a_1^g \cdots a_n^g$, Θ)
  agenda $\leftarrow$ { (ϕ, $c_1 \cdots c_n$, score=0.0) }
  for i **in** $1 \cdots n$
    beam $\leftarrow$ { }
    for *cand* **in** *agenda*
      new $\leftarrow$ SEP(*cand, c_i, Θ)*
      if $\{a_i^g \neq$ SEP$\}$ new.*score* += η
      ADDITEM*(*beam*,* new*)*
      new $\leftarrow$ APP*(*cand, c_i, Θ)
      if $\{a_i^g \neq$ APP$\}$ *new*.score += η
      ADDITEM(*beam*, *new*)
    agenda $\leftarrow$ TOP-B(*beam, B*)
    if $\{$ITEM$(a_1^g \cdots a_i^g) \notin$ agenda$\}$
      $\Theta = \Theta - f\big($BESTITEM(*agenda*)$\big)$
      $\Theta = \Theta + f\big($ITEM$((a_1^g \cdots a_i^g)\big)$
      return
  if $\{$ITEM$(a_1^g \cdots a_n^g) \neq$ BESTITEM(*agenda*)$\}$
    $\Theta = \Theta - f\big($BESTITEM(*agenda*)$\big)$
    $\Theta = \Theta + f\big($ITEM$((a_1^g \cdots a_n^g)\big)$

4 Training

To train model parameters for both the discrete and neural models, we exploit online learning with early-update as shown in Algorithm 2. A max-margin objective is exploited,[3] which is defined as:

$$L(\Theta) = \frac{1}{K}\sum_{k=1}^{K} l(A_k^g, \Theta) + \frac{\lambda}{2} \parallel \Theta \parallel^2$$

$$l(A_k^g, \Theta) = \max_{A} \big(score(A_k, \Theta) + \eta \cdot \delta(A_k, A_k^g)\big) - score(A_k^g, \Theta),$$

where Θ is the set of all parameters, $\{A_k^g\}_{n=1}^K$ are gold action sequences to segment the training corpus, A_k is the model output action sequence, λ is a regularization parameter and η is used to tune the loss margins.

For the discrete models, $f(\cdot)$ denotes the features extracted according to the feature templates in Table 1. For the neural models, $f(\cdot)$ denotes the corresponding $\mathbf{h}_{sep}$ and $\mathbf{h}_{app}$. Thus only the output layer is updated, and we further use back-propagation to learn the parameters of the other layers (LeCun et al., 2012). We use online AdaGrad (Duchi et al., 2011) to minimize the objective function for both the discrete and neural models. All the matrix and vector parameters are initialized by uniform sampling in $(-0.01, 0.01)$.

[3]Zhou et al. (2015) find that max-margin training did not yield reasonable results for neural transition-based parsing, which is different from our findings. One likely reason is that when the number of labels is small max-margin is effective.

		CTB60	PKU	MSR
Training	#sent	23k	17k	78k
	#word	641k	1,010k	2,122k
Development	#sent	2.1k	1.9k	8.7k
	#word	60k	100k	246k
Test	#sent	2.8k	1.9k	4.0k
	#word	82k	104k	106k

Table 2: Statistics of datasets.

Type	hyper-parameters
Network structure	$d(\mathbf{h}_{sep}) = 100$, $d(\mathbf{h}_{app}) = 80$
	$d(\mathbf{h}_i^a) = 20$, $d(\mathbf{x}_i^a) = 20$
	$d(\mathbf{h}_i^w) = 50$, $d(\mathbf{x}_i^w) = 50$
	$d(\mathbf{h}_i^{cL}) = d(\mathbf{h}_i^{cR}) = 50$, $d(\mathbf{x}_i^c) = 50$
	$d(\mathbf{e}_w(w_i)) = 50$, $d(\mathbf{e}_a(a_i)) = 20$
	$d(\mathbf{e}_c(c_i)) = 50$, $d(\mathbf{e}_{bc}(c_{i-1}c_i)) = 50$
Training	$\lambda = 10^{-8}$, $\alpha = 0.01$, $\eta = 0.2$

Table 3: Hyper-parameter values in our model.

5 Experiments

5.1 Experimental Settings

Data. We use three datasets for evaluation, namely CTB6, PKU and MSR. The CTB6 corpus is taken from Chinese Treebank 6.0, and the PKU and MSR corpora can be obtained from Bake-Off 2005 (Emerson, 2005). We follow Zhang et al. (2014), splitting the CTB6 corpus into training, development and testing sections. For the PKU and MSR corpora, only the training and test datasets are specified and we randomly split 10% of the training sections for development. Table 1 shows the overall statistics of the three datasets.

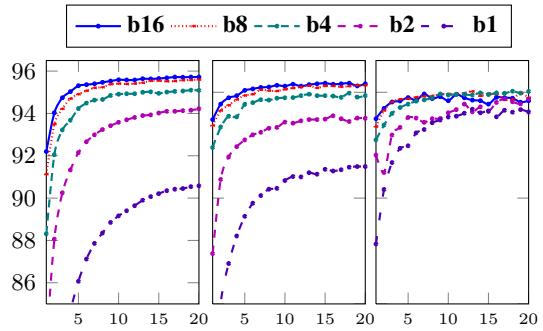
Embeddings. We use *word2vec*[4] to pre-train word, character and bi-character embeddings on Chinese Gigaword corpus (LDC2011T13). In order to train full word embeddings, the corpus is segmented automatically by our baseline model.

Hyper-parameters. The hyper-parameter values are tuned according to development performances. We list their final values in Table 3.

5.2 Development Results

To better understand the word-based neural models, we perform several development experiments. *All* the experiments in this section are conducted on the CTB6 development dataset.

[4]http://word2vec.googlecode.com/



(a) discrete (b) neural(-tune) (c) neural(+tune)

Figure 5: Accuracies against the training epoch using beam sizes 1, 2, 4, 8 and 16, respectively.

5.2.1 Embeddings and beam size

We study the influence of beam size on the baseline and neural models. Our neural model has two choices of using pre-trained word embeddings. We can either fine-tune or fix the embeddings during training. In case of fine-tuning, only words in the training data can be learned, while embeddings of out-of-vocabulary (OOV) words could not be used effectively.[5] In addition, following Dyer et al. (2015) we randomly set words with frequency 1 in the training data as the OOV words in order to learn the OOV embedding, while avoiding overfitting. If the pretrained word embeddings are not fine-tuned, we can utilize all word embeddings.

Figure 5 shows the development results, where the training curve of the discrete baseline is shown in Figure 5(a) and the curve of the neural model without and with fine tuning are shown in 5(b) and 5(c), respectively. The performance increases with a larger beam size in all settings. When the beam increases into 16, the gains levels out. The results of the discrete model and the neural model without fine-tuning are highly similar, showing the usefulness of beam-search.

On the other hand, with fine-tuning, the results are different. The model with beam size 1 gives better accuracies compared to the other models with the same beam size. However, as the beam size increases, the performance increases very little. The results are consistent with Dyer et al. (2015), who find that beam-search improves the results only slightly on dependency parsing. When a beam size of 16 is used, this model performs the worst compared with the discrete model and the neural model without fine-tuning. This is likely because the fine-tuning of embeddings leads to overfitting of in-vocabulary words, and underfitting over OOV words. Based on the observation, we exploit fixed word embeddings in our final models.

[5]We perform experiments using random initialized word embeddings as well when fine-tune is used, which is a fully supervised model. The performance is slightly lower.

Model	P	R	F
neural	95.21	95.69	95.45
-word	91.81	92.00	91.90
-character unigram	94.89	95.56	95.22
-character bigram	94.93	95.53	95.23
-action	95.00	95.31	95.17
+discrete features (combined)	96.38	96.22	96.30

Table 4: Feature experiments.

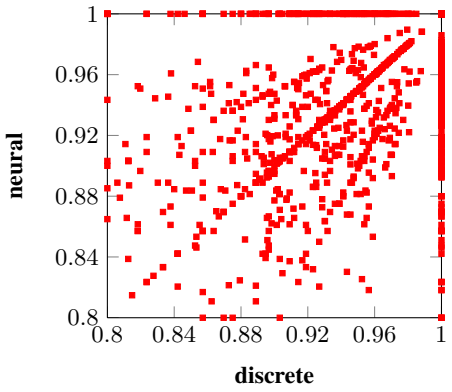


Figure 6: Sentence accuracy comparisons for the discrete and the neural models.

5.2.2 Feature ablation

We conduct feature ablation experiments to study the effects of the word, character unigram, character bigram and action features to the neural model. The results are shown in Table 4. Word features are particularly important to the model, without which the performance decreases by 4.5%. The effects of the character unigram, bigram and action features are relatively much weaker.[6] This demonstrates that in the word-based incremental search framework, words are the most crucial information to the neural model.

5.2.3 Integrating discrete features

Prior work has shown the effectiveness of integrating discrete and neural features for several NLP tasks (Turian et al., 2010; Wang and Manning,

[6]In all our experiments, we fix the character unigram and bigram embeddings, because fine-tuning of these embeddings results in little changes.

Models	P	R	F
word-based models			
discrete	95.29	95.26	95.28
neural	95.34	94.69	95.01
combined	**96.11**	**95.79**	**95.95**
character-based models			
discrete	95.38	95.12	95.25
neural	94.59	94.92	94.76
combined	95.63	95.60	95.61
other models			
Zhang et al. (2014)	N/A	N/A	95.71
Wang et al. (2011)	95.83	95.75	95.79
Zhang and Clark (2011)	95.46	94.78	95.13

Table 5: Main results on CTB60 test dataset.

2013; Durrett and Klein, 2015; Zhang and Zhang, 2015). We investigate the usefulness of such integration to our word-based segmentor on the development dataset. We study it by two ways. First, we compare the error distributions between the discrete and the neural models. Intuitively, different error distributions are necessary for improvements by integration. We draw a scatter graph to show their differences, with the (x, y) values of each point denoting the F-measure scores of the two models with respect to sentences, respectively. As shown in Figure 6, the points are rather dispersive, showing the differences of the two models.

Further, we directly look at the results after integration of both discrete and neural features. As shown in Table 4, the integrated model improves the accuracies from 95.45% to 96.30%, demonstrating that the automatically-induced neural features contain highly complementary information to the manual discrete features.

5.3 Final Results

Table 6 shows the final results on CTB6 test dataset. For thorough comparison, we implement discrete, neural and combined character-based models as well.[7] In particular, the character-based discrete model is a CRF tagging model using character unigrams, bigrams, trigrams and tag transitions (Tseng et al., 2005), and the character-based neural model exploits a bi-directional LSTM layer to model character sequences[8] and a CRF layer for output (Chen et al., 2015b).[9] The combined model uses the same method for integrating discrete and neural features as our word-based model.

Models	PKU	MSR
our word-based models		
discrete	95.1	97.3
neural	95.1	97.0
combined	95.7	**97.7**
character-based models		
discrete	94.9	96.8
neural	94.4	97.2
combined	95.4	97.2
other models		
Cai and Zhao (2016)	95.5	96.5
Ma and Hinrichs (2015)	95.1	96.6
Pei et al. (2014)	95.2	97.2
Zhang et al. (2013a)	**96.1**	97.5
Sun et al. (2012)	95.4	97.4
Zhang and Clark (2011)	95.1	97.1
Sun (2010)	95.2	96.9
Sun et al. (2009)	95.2	97.3

Table 6: Main results on PKU and MSR test datasets.

The word-based models achieve better performances than character-based models, since our model can exploit additional word information learnt from large auto-segmented corpus. We also compare the results with other models. Wang et al. (2011) is a semi-supervised model that exploits word statistics from auto-segmented raw corpus, which is similar with our combined model in using semi-supervised word information. We achieve slightly better accuracies. Zhang et al. (2014) is a joint segmentation, POS-tagging and dependency parsing model, which can exploit syntactic information.

To compare our models with other state-of-the-art models in the literature, we report the performance on the PKU and MSR datasets also.[10] Our combined model gives the best result on the MSR dataset, and the second best on PKU. The method of Zhang et al. (2013a) gives the best performance on PKU by co-training on large-scale data.

5.4 Error Analysis

To study the differences between word-based and character-based neural models, we conduct error analysis on the test dataset of CTB60. First,

[7] The code is released for research reference under GPL at https://github.com/SUTDNLP/NNSegmentation.

[8] We use a concatenation of character unigram and bigram embeddings at each position as the input to LSTM, because our experiments show that the character bigram embeddings are useful, without which character-based neural models are significantly lower than their discrete counterparts.

[9] Bi-directional LSTM is slightly better than a single left-right LSTM used in Chen et al. (2015b).

[10] The results of Chen et al. (2015a) and Chen et al. (2015b) are not listed, because they take a preprocessing step by replacing Chinese idioms with a uniform symbol in their test data.

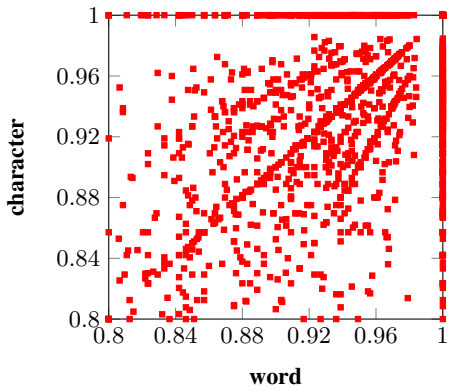


Figure 7: Sentence accuracy comparisons for word- and character-based neural models.

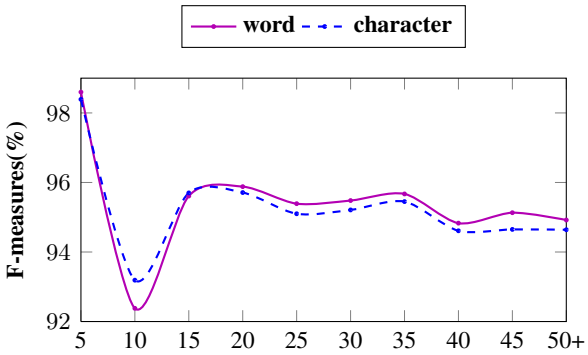


Figure 8: F-measure against character length.

we examine the error distribution on individual sentences. Figure 7 shows the F-measure values of each test sentence by word- and character-based neural models, respectively, where the x-axis value denotes the F-measure value of the word-based neural model, and the y-axis value denotes its performance of the character-based neural model. We can see that the majority scatter points are off the diagonal line, demonstrating strong differences between the two models. This results from the differences in feature sources.

Second, we study the F-measure distribution of the two neural models with respect to sentence lengths. We divide the test sentences into ten bins, with bin i denoting sentence lengths in $[5 * (i-1), 5 * i]$. Figure 8 shows the results. According to the figure, we observe that word-based neural model is relatively weaker for sentences with length in $[5, 10]$, while can better tackle long sentences.

Third, we compare the two neural models by their capabilities of modeling words with different lengths. Figure 9 shows the results. The performances are lower for words with lengths beyond 2, and the performance drops significantly for words with lengths over 3. Overall, the word-based neural model achieves comparable performances with the character-based model, but gives significantly better performances for long words, in particular when the word length is over 3. This demonstrates the advantage of word-level features.

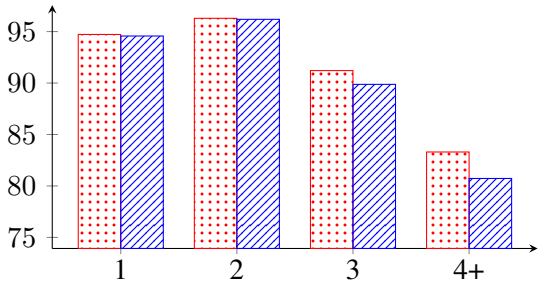


Figure 9: F-measure against word length, where the boxes with red dots denote the performances of word-based neural model, and the boxes with blue slant lines denote character-based neural model.

6 Related Work

Xue (2003) was the first to propose a character-tagging method to Chinese word segmentation, using a maximum entropy model to assign B/I/E/S tags to each character in the input sentence separately. Peng et al. (2004) showed that better results can be achieved by global learning using a CRF model. This method has been followed by most subsequent models in the literature (Tseng et al., 2005; Zhao, 2009; Sun et al., 2012). The most effective features have been character unigrams, bigrams and trigrams within a five-character window, and a bigram tag window. Special characters such as alphabets, numbers and date/time characters are also differentiated for extracting features.

Zheng et al. (2013) built a neural network segmentor, which essentially substitutes the manual discrete features of Peng et al. (2004), with dense real-valued features induced automatically from character embeddings, using a deep neural network structure (Collobert et al., 2011). A tag transition matrix is used for inference, which makes the model effectively. Most subsequent work on neural segmentation followed this method, improving the extraction of emission features by using more complex neural network structures.

Mansur et al. (2013) experimented with embeddings of richer features, and in particular charac-

ter bigrams. Pei et al. (2014) used a tensor neural network to achieve extensive feature combinations, capturing the interaction between characters and tags. Chen et al. (2015a) used a recursive network structure to the same end, extracting more combined features to model complicated character combinations in a five-character window. Chen et al. (2015b) used a LSTM model to capture long-range dependencies between characters in a sentence. Xu and Sun (2016) proposed a dependency-based gated recursive neural network to efficiently integrate local and long-distance features. The above methods are all character-based models, making no use of full word information. In contrast, we leverage both character embeddings and word embeddings for better accuracies.

For word-based segmentation, Andrew (2006) used a semi-CRF model to integrate word features, Zhang and Clark (2007) used a perceptron algorithm with inexact search, and Sun et al. (2009) used a discriminative latent variable model to make use of word features. Recently, there have been several neural-based models using word-level embedding features (Morita et al., 2015; Liu et al., 2016; Cai and Zhao, 2016), which are different from our work in the basic framework. For instance, Liu et al. (2016) follow Andrew (2006) using a semi-CRF for structured inference.

We followed the global learning and beam-search framework of Zhang and Clark (2011) in building a word-based neural segmentor. The main difference between our model and that of Zhang and Clark (2011) is that we use a neural network to induce feature combinations directly from character and word embeddings. In addition, the use of a bi-directional LSTM allows us to leverage non-local information from the word sequence, and look-ahead information from the incoming character sequence. The automatic neural features are complementary to the manual discrete features of Zhang and Clark (2011). We show that our model can accommodate the integration of both types of features. This is similar in spirit to the work of Sun (2010) and Wang et al. (2014), who integrated features of character-based and word-based segmentors.

Transition-based framework with beam search has been widely exploited in a number of other NLP tasks, including syntactic parsing (Zhang and Nivre, 2011; Zhu et al., 2013), information extraction (Li and Ji, 2014) and the work of joint models (Zhang et al., 2013b; Zhang et al., 2014). Recently, the effectiveness of neural features has been studied for this framework. In the natural language parsing community, it has achieved great success. Representative work includes Zhou et al. (2015), Weiss et al. (2015), Watanabe and Sumita (2015) and Andor et al. (2016). In this work, we apply the transition-based neural framework to Chinese segmentation, in order to exploit word-level neural features such as word embeddings.

7 Conclusion

We proposed a word-based neural model for Chinese segmentation, which exploits not only character embeddings as previous work does, but also word embeddings pre-trained from large scale corpus. The model achieved comparable performances compared with a discrete word-based baseline, and also the state-of-the-art character-based neural models in the literature. We further demonstrated that the model can utilize discrete features conveniently, resulting in a combined model that achieved top performances compared with previous work. Finally, we conducted several comparisons to study the differences between our word-based model with character-based neural models, showing that they have different error characteristics.

Acknowledgments

We thank the anonymous reviewers, Yijia Liu and Hai Zhao for their constructive comments, which help to improve the final paper. This work is supported by National Natural Science Foundation of China (NSFC) under grant 61170148, Natural Science Foundation of Heilongjiang Province (China) under grant No.F2016036, the Singapore Ministry of Education (MOE) AcRF Tier 2 grant T2MOE201301 and SRG ISTD 2012 038 from Singapore University of Technology and Design. Yue Zhang is the corresponding author.

References

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

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation.

In *Proceedings of the 2006 Conference on EMNLP*, pages 465–472, Sydney, Australia, July.

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of ACL 2016*.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015a. Gated recursive neural network for chinese word segmentation. In *Proceedings of the 53nd ACL*, pages 1744–1753, July.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015b. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 EMNLP*, pages 1197–1206, September.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53nd ACL*, pages 302–312, July.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53nd ACL*, pages 334–343, July.

Thomas Emerson. 2005. The second international chinese word segmentation bakeoff. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*, pages 123–133.

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

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the ACL 2014*.

Yijia Liu, Wanxiang Che, Jiang Guo, Bing Qin, and Ting Liu. 2016. Exploring segment representations for neural segmentation models. In *Proceedings of IJCAI 2016*.

Jianqiang Ma and Erhard Hinrichs. 2015. Accurate linear-time chinese word segmentation via embedding matching. In *Proceedings of the 53nd ACL*, pages 1733–1743, July.

Mairgup Mansur, Wenzhe Pei, and Baobao Chang. 2013. Feature-based neural language model and chinese word segmentation. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1271–1277, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Hajime Morita, Daisuke Kawahara, and Sadao Kurohashi. 2015. Morphological analysis for unsegmented languages using recurrent neural network language model. In *Proceedings of the 2015 Conference on EMNLP*, pages 2292–2297.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd ACL*, pages 293–303, Baltimore, Maryland, June.

Fuchun Peng, Fangfang Feng, and Andrew McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. In *Proceedings of Coling 2004*, pages 562–568, Geneva, Switzerland, Aug 23–Aug 27.

Yanxin Shi and Mengqiu Wang. 2007. A dual-layer crfs based joint decoding method for cascaded segmentation and labeling tasks. In *IJCAI*, pages 1707–1712.

Weiwei Sun and Jia Xu. 2011. Enhancing chinese word segmentation using unlabeled data. In *Proceedings of the 2011 Conference on EMNLP*, pages 970–979, July.

Xu Sun, Yaozhong Zhang, Takuya Matsuzaki, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2009. A discriminative latent variable chinese segmenter with hybrid word/character information. In *Proceedings of NAACL 2009*, pages 56–64, June.

Xu Sun, Houfeng Wang, and Wenjie Li. 2012. Fast online training with frequency-adaptive learning rates for chinese word segmentation and new word detection. In *Proceedings of the 50th ACL*, pages 253–262, July.

Weiwei Sun. 2010. Word-based and character-based word segmentation models: Comparison and combination. In *Coling 2010: Posters*, pages 1211–1219, August.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proceedings of the fourth SIGHAN workshop*, pages 168–171.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394, July.

Mengqiu Wang and Christopher D. Manning. 2013. Effect of non-linear deep architecture in sequence labeling. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1285–1291, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Yiou Wang, Jun'ichi Kazama, Yoshimasa Tsuruoka, Wenliang Chen, Yujie Zhang, and Kentaro Torisawa. 2011. Improving chinese word segmentation and pos tagging with semi-supervised methods using large auto-analyzed data. In *Proceedings of 5th IJCNLP*, pages 309–317, Chiang Mai, Thailand, November.

Mengqiu Wang, Rob Voigt, and Christopher D. Manning. 2014. Two knives cut better than one: Chinese word segmentation with dual decomposition. In *Proceedings of the 52nd ACL*, pages 193–198, Baltimore, Maryland, June.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd ACL*, pages 1169–1179, July.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd ACL*, pages 323–333, July.

Jingjing Xu and Xu Sun. 2016. Dependency-based gated recursive neural network for chinese word segmentation. In *Proceedings of ACL 2016*.

Nianwen Xue. 2003. Chinese word segmentation as character tagging. *International Journal of Computational Linguistics and Chinese Language Processing*, 8(1).

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. *arXiv preprint arXiv:1506.00196*.

Yue Zhang and Stephen Clark. 2007. Chinese segmentation with a word-based perceptron algorithm. In *Proceedings of the 45th ACL*, pages 840–847, Prague, Czech Republic, June.

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

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th ACL*, pages 188–193, June.

Meishan Zhang and Yue Zhang. 2015. Combining discrete and continuous features for deterministic transition-based dependency parsing. In *Proceedings of the 2015 EMNLP*, pages 1316–1321, September.

Longkai Zhang, Houfeng Wang, Xu Sun, and Mairgup Mansur. 2013a. Exploring representations from unlabeled data with co-training for Chinese word segmentation. In *Proceedings of the EMNLP 2013*, pages 311–321, Seattle, Washington, USA, October.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013b. Chinese parsing exploiting characters. In *Proceedings of the 51st ACL*, pages 125–134, August.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52nd ACL*, pages 1326–1336, Baltimore, Maryland, June.

Hai Zhao, Chang-Ning Huang, Mu Li, and Bao-Liang Lu. 2006. Effective tag set selection in chinese word segmentation via conditional random field modeling. In *Proceedings of PACLIC*, volume 20, pages 87–94. Citeseer.

Hai Zhao. 2009. Character-level dependencies in chinese: Usefulness and learning. In *Proceedings of the EACL*, pages 879–887, Athens, Greece, March.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on EMNLP*, pages 647–657, Seattle, Washington, USA, October.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd ACL*, pages 1213–1222, July.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st ACL*, pages 434–443, August.

A Parallel-Hierarchical Model for Machine Comprehension on Sparse Data

Adam Trischler* adam.trischler
Zheng Ye* jeff.ye
Xingdi Yuan eric.yuan
Jing He jing.he
Philip Bachman phil.bachman
Kaheer Suleman k.suleman@maluuba.com
Maluuba Research
Montreal, Québec, Canada

Abstract

Understanding unstructured text is a major goal within natural language processing. Comprehension tests pose questions based on short text passages to evaluate such understanding. In this work, we investigate machine comprehension on the challenging *MCTest* benchmark. Partly because of its limited size, prior work on *MCTest* has focused mainly on engineering better features. We tackle the dataset with a neural approach, harnessing simple neural networks arranged in a parallel hierarchy. The parallel hierarchy enables our model to compare the passage, question, and answer from a variety of trainable perspectives, as opposed to using a manually designed, rigid feature set. Perspectives range from the word level to sentence fragments to sequences of sentences; the networks operate only on word-embedding representations of text. When trained with a methodology designed to help cope with limited training data, our Parallel-Hierarchical model sets a new state of the art for *MCTest*, outperforming previous feature-engineered approaches slightly and previous neural approaches by a significant margin (over 15 percentage points).

1 Introduction

Humans learn in a variety of ways—by communication with each other and by study, the reading of text. Comprehension of unstructured text by machines, at a near-human level, is a major goal for natural language processing. It has garnered significant attention from the machine learning research community in recent years.

Machine comprehension (MC) is evaluated by posing a set of questions based on a text passage (akin to the reading tests we all took in school). Such tests are objectively gradable and can be used to assess a range of abilities, from basic understanding to causal reasoning to inference (Richardson et al., 2013). Given a text passage and a question about its content, a system is tested on its ability to determine the correct answer (Sachan et al., 2015). In this work, we focus on *MCTest*, a complex but data-limited comprehension benchmark, whose multiple-choice questions require not only extraction but also inference and limited reasoning (Richardson et al., 2013). Inference and reasoning are important human skills that apply broadly, beyond language.

We present a parallel-hierarchical approach to machine comprehension designed to work well in a data-limited setting. There are many use-cases in which comprehension over limited data would be handy: for example, user manuals, internal documentation, legal contracts, and so on. Moreover, work towards more efficient learning from any quantity of data is important in its own right, for bringing machines more in line with the way humans learn. Typically, artificial neural networks require numerous parameters to capture complex patterns, and the more parameters, the more training data is required to tune them. Likewise, deep models learn to extract their own features, but this is a data-intensive process. Our model learns to comprehend at a high level even when data is sparse.

The key to our model is that it compares the question and answer candidates to the text using several distinct *perspectives*. We refer to a question combined with one of its answer candidates as a *hypothesis* (to be detailed below). The *seman-*

*A. Trischler and Z. Ye contributed equally to this work.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 432–441,
Berlin, Germany, August 7-12, 2016.

tic perspective compares the hypothesis to sentences in the text viewed as single, self-contained thoughts; these are represented using a sum and transformation of word embedding vectors, similarly to Weston et al. (2014). The *word-by-word* perspective focuses on similarity matches between individual words from hypothesis and text, at various scales. As in the semantic perspective, we consider matches over complete sentences. We also use a sliding window acting on a subsentential scale (inspired by the work of Hill et al. (2015)), which implicitly considers the linear distance between matched words. Finally, this word-level sliding window operates on two different views of story sentences: the *sequential* view, where words appear in their natural order, and the *dependency* view, where words are reordered based on a linearization of the sentence's dependency graph. Words are represented throughout by embedding vectors (Bengio et al., 2000; Mikolov et al., 2013). These distinct perspectives naturally form a hierarchy that we depict in Figure 1. Language is hierarchical, so it makes sense that comprehension relies on hierarchical levels of understanding.

The perspectives of our model can be considered a type of feature. However, they are implemented by parametric differentiable functions. This is in contrast to most previous efforts on *MCTest*, whose numerous hand-engineered features cannot be trained. Our model, significantly, can be trained end-to-end with backpropagation. To facilitate learning with limited data, we also develop a unique training scheme. We initialize the model's neural networks to perform specific heuristic functions that yield decent (though not impressive) performance on the dataset. Thus, the training scheme gives the model a safe, reasonable baseline from which to start learning. We call this technique *training wheels*.

Computational models that comprehend (insofar as they perform well on MC datasets) have been developed contemporaneously in several research groups (Weston et al., 2014; Sukhbaatar et al., 2015; Hill et al., 2015; Hermann et al., 2015; Kumar et al., 2015). Models designed specifically for *MCTest* include those of Richardson et al. (2013), and more recently Sachan et al. (2015), Wang et al. (2015), and Yin et al. (2016). In experiments, our *Parallel-Hierarchical* model achieves state-of-the-art accuracy on *MCTest*, outperforming these existing methods.

Below we describe related work, the mathematical details of our model, and our experiments, then analyze our results.

2 The Problem

In this section, we borrow from Sachan et al. (2015), who laid out the MC problem nicely. Machine comprehension requires machines to answer questions based on unstructured text. This can be viewed as selecting the best answer from a set of candidates. In the multiple-choice case, candidate answers are predefined, but candidate answers may also be undefined yet restricted (*e.g.*, to *yes*, *no*, or any noun phrase in the text) (Sachan et al., 2015).

For each question q, let T be the unstructured text and $A = \{a_i\}$ the set of candidate answers to q. The machine comprehension task reduces to selecting the answer that has the highest evidence given T. As in Sachan et al. (2015), we combine an answer and a question into a *hypothesis*, $h_i = f(q, a_i)$. To facilitate comparisons of the text with the hypotheses, we also break down the passage into sentences t_j, $T = \{t_j\}$. In our setting, q, a_i, and t_j each represent a sequence of embedding vectors, one for each word and punctuation mark in the respective item.

3 Related Work

Machine comprehension is currently a hot topic within the machine learning community. In this section we will focus on the best-performing models applied specifically to *MCTest*, since it is somewhat unique among MC datasets (see Section 5). Generally, models can be divided into two categories: those that use fixed, engineered features, and neural models. The bulk of the work on *MCTest* falls into the former category.

Manually engineered features often require significant effort on the part of a designer, and/or various auxiliary tools to extract them, and they cannot be modified by training. On the other hand, neural models can be trained end-to-end and typically harness only a single feature: vector-representations of words. Word embeddings are fed into a complex and possibly deep neural network which processes and compares text to question and answer. Among deep models, mechanisms of attention and working memory are common, as in Weston et al. (2014) and Hermann et al. (2015).

3.1 Feature-engineering models

Sachan et al. (2015) treated *MCTest* as a structured prediction problem, searching for a latent *answer-entailing* structure connecting question, answer, and text. This structure corresponds to the best latent alignment of a hypothesis with appropriate snippets of the text. The process of (latently) selecting text snippets is related to the attention mechanisms typically used in deep networks designed for MC and machine translation (Bahdanau et al., 2014; Weston et al., 2014; Hill et al., 2015; Hermann et al., 2015). The model uses event and entity coreference links across sentences along with a host of other features. These include specifically trained word vectors for synonymy; antonymy and class-inclusion relations from external database sources; dependencies and semantic role labels. The model is trained using a latent structural SVM extended to a multitask setting, so that questions are first classified using a pretrained top-level classifier. This enables the system to use different processing strategies for different question categories. The model also combines question and answer into a well-formed statement using the rules of Cucerzan and Agichtein (2005).

Our model is simpler than that of Sachan et al. (2015) in terms of the features it takes in, the training procedure (stochastic gradient descent *vs.* alternating minimization), question classification (we use none), and question-answer combination (simple concatenation or mean *vs.* a set of rules).

Wang et al. (2015) augmented the baseline feature set from Richardson et al. (2013) with features for syntax, frame semantics, coreference chains, and word embeddings. They combined features using a linear latent-variable classifier trained to minimize a max-margin loss function. As in Sachan et al. (2015), questions and answers are combined using a set of manually written rules. The method of Wang et al. (2015) achieved the previous state of the art, but has significant complexity in terms of the feature set.

Space does not permit a full description of all models in this category, but we refer the reader to the contributions of Smith et al. (2015) and Narasimhan and Barzilay (2015) as well.

Despite its relative lack of features, the Parallel-Hierarchical model improves upon the feature-engineered state of the art for *MCTest* by a small amount (about 1% absolute) as detailed in Section 5.

3.2 Neural models

Neural models have, to date, performed relatively poorly on *MCTest*. This is because the dataset is sparse and complex.

Yin et al. (2016) investigated deep-learning approaches concurrently with the present work. They measured the performance of the Attentive Reader (Hermann et al., 2015) and the Neural Reasoner (Peng et al., 2015), both deep, end-to-end recurrent models with attention mechanisms, and also developed an attention-based convolutional network, the HABCNN. Their network operates on a hierarchy similar to our own, providing further evidence of the promise of hierarchical perspectives. Specifically, the HABCNN processes text at the sentence level and the *snippet* level, where the latter combines adjacent sentences (as we do through an n-gram input). Embedding vectors for the question and the answer candidates are combined and encoded by a convolutional network. This encoding modulates attention over sentence and snippet encodings, followed by max-pooling to determine the best matches between question, answer, and text. As in the present work, matching scores are given by cosine similarity. The HABCNN also makes use of a question classifier.

Despite the conceptual overlap between the HABCNN and our approach, the Parallel-Hierarchical model performs significantly better on *MCTest* (more than 15% absolute) as detailed in Section 5. Other neural models tested in Yin et al. (2016) fare even worse.

4 The Parallel-Hierarchical Model

Let us now define our machine comprehension model in full. We first describe each of the perspectives separately, then describe how they are combined. Below, we use subscripts to index elements of sequences, like word vectors, and superscripts to indicate whether elements come from the text, question, or answer. In particular, we use the subscripts k, m, n, p to index sequences from the text, question, answer, and hypothesis, respectively, and superscripts t, q, a, h. We depict the model schematically in Figure 1.

4.1 Semantic Perspective

The semantic perspective is similar to the Memory Networks approach for embedding inputs into memory space (Weston et al., 2014). Each sen-

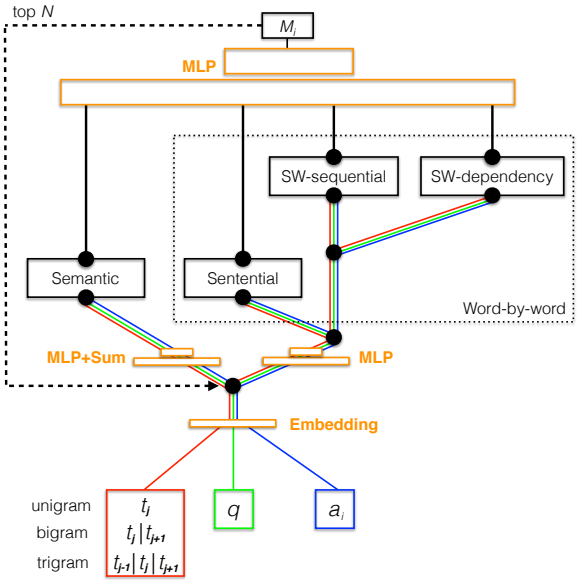


Figure 1: Schematic of the Parallel-Hierarchical model. SW stands for "sliding window." MLP represents a fully connected neural network.

tence of the text is a sequence of d-dimensional word vectors: $t_j = \{\mathbf{t}_k\}$, $\mathbf{t}_k \in \mathbb{R}^d$. The semantic vector $\mathbf{s}^t$ is computed by embedding the word vectors into a D-dimensional space using a two-layer network that implements weighted sum followed by an affine tranformation and a nonlinearity; *i.e.*,

$$\mathbf{s}^t = f\left(\mathbf{A}^t \sum_k \omega_k \mathbf{t}_k + \mathbf{b}_A^t\right). \quad (1)$$

The matrix $\mathbf{A}^t \in \mathbb{R}^{D\times d}$, the bias vector $\mathbf{b}_A^t \in \mathbb{R}^D$, and for f we use the *leaky ReLU* function. The scalar ω_k is a trainable weight associated with each word in the vocabulary. These scalar weights implement a kind of *exogenous* or bottom-up attention that depends only on the input stimulus (Mayer et al., 2004). They can, for example, learn to perform the function of stopword lists in a soft, trainable way, to nullify the contribution of unimportant filler words.

The semantic representation of a hypothesis is formed analogously, except that we concatenate the question word vectors $\mathbf{q}_m$ and answer word vectors $\mathbf{a}_n$ as a single sequence $\{\mathbf{h}_p\} = \{\mathbf{q}_m, \mathbf{a}_n\}$. For semantic vector $\mathbf{s}^h$ of the hypothesis, we use a unique transformation matrix $\mathbf{A}^h \in \mathbb{R}^{D\times d}$ and bias vector $\mathbf{b}_A^h \in \mathbb{R}^D$.

These transformations map a text sentence and a hypothesis into a common space where they can be compared. We compute the semantic match between text sentence and hypothesis using the cosine similarity,

$$M^{\text{sem}} = \cos(\mathbf{s}^t, \mathbf{s}^h). \quad (2)$$

4.2 Word-by-Word Perspective

The first step in building the word-by-word perspective is to transform word vectors from a text sentence, question, and answer through respective neural functions. For the text, $\tilde{\mathbf{t}}_k = f\left(\mathbf{B}^t\mathbf{t}_k + \mathbf{b}_B^t\right)$, where $\mathbf{B}^t \in \mathbb{R}^{D\times d}$, $\mathbf{b}_B^t \in \mathbb{R}^D$ and f is again the *leaky ReLU*. We transform the question and the answer to $\tilde{\mathbf{q}}_m$ and $\tilde{\mathbf{a}}_n$ analogously using distinct matrices and bias vectors. In contrast to the semantic perspective, we keep the question and answer candidates separate in the word-by-word perspective. This is because matches to answer words are inherently more important than matches to question words, and we want our model to learn to use this property.

4.2.1 Sentential

Inspired by the work of Wang and Jiang (2015) in paraphrase detection, we compute matches between hypotheses and text sentences at the word level. This computation uses the cosine similarity as before:

$$c_{km}^q = \cos(\tilde{\mathbf{t}}_k, \tilde{\mathbf{q}}_m), \quad (3)$$
$$c_{kn}^a = \cos(\tilde{\mathbf{t}}_k, \tilde{\mathbf{a}}_n). \quad (4)$$

The word-by-word match between a text sentence and question is determined by taking the maximum over k (finding the text word that best matches each question word) and then taking a weighted mean over m (finding the average match over the full question):

$$M^q = \frac{1}{Z}\sum_m \omega_m \max_k c_{km}^q. \quad (5)$$

Here, ω_m is the word weight for the question word and Z normalizes these weights to sum to one over the question. We define the match between a sentence and answer candidate, M^a, analogously. Finally, we combine the matches to question and answer according to

$$M^{\text{word}} = \alpha_1 M^q + \alpha_2 M^a + \alpha_3 M^q M^a. \quad (6)$$

Here, the α are trainable parameters that control the relative importance of the terms.

4.2.2 Sequential Sliding Window

The sequential sliding window is related to the original *MCTest* baseline by Richardson et al. (2013). Our sliding window decays from its focus word according to a Gaussian distribution, which we extend by assigning a trainable weight to each location in the window. This modification enables the window to use information about the distance between word matches; the original baseline (Richardson et al., 2013) used distance information through a predefined function.

The sliding window scans over the words of the text as one continuous sequence, without sentence breaks. Each window is treated like a sentence in the previous subsection, but we include a location-based weight $\lambda(k)$. This weight is based on a word's position in the window, which, given a window, depends on its global position k. The cosine similarity is adapted as

$$s^q_{km} = \lambda(k) \cos(\tilde{\mathbf{t}}_k, \tilde{\mathbf{q}}_m), \tag{7}$$

for the question and analogously for the answer. We initialize the location weights with a Gaussian and fine-tune them during training. The final matching score, denoted as M^{sws}, is computed as in (5) and (6) with s^q_{km} replacing c^q_{km}.

4.2.3 Dependency Sliding Window

The dependency sliding window operates identically to the linear sliding window, but on a different view of the text passage. The output of this component is M^{swd} and is formed analogously to M^{sws}.

The dependency perspective uses the Stanford Dependency Parser (Chen and Manning, 2014) as an auxiliary tool. Thus, the dependency graph can be considered a fixed feature. Moreover, linearization of the dependency graph, because it relies on an eigendecomposition, is not differentiable. However, we handle the linearization in data preprocessing so that the model sees only reordered word-vector inputs.

Specifically, we run the Stanford Dependency Parser on each text sentence to build a dependency graph. This graph has n_w vertices, one for each word in the sentence. From the dependency graph we form the Laplacian matrix $\mathbf{L} \in \mathbb{R}^{n_w \times n_w}$ and determine its eigenvectors. The second eigenvector $\mathbf{u}_2$ of the Laplacian is known as the *Fiedler* vector. It is the solution to the minimization

$$\underset{g}{\text{minimize}} \sum_{i,j=1}^{N} \eta_{ij} (g(v_i) - g(v_j))^2, \tag{8}$$

where v_i are the vertices of the graph and η_{ij} is the weight of the edge from vertex i to vertex j (Golub and Van Loan, 2012). The Fiedler vector maps a weighted graph onto a line such that connected nodes stay close, modulated by the connection weights.[1] This enables us to reorder the words of a sentence based on their proximity in the dependency graph. The reordering of the words is given by the ordered index set

$$I = \arg\text{sort}(\mathbf{u}_2). \tag{9}$$

To give an example of how this works, consider the following sentence from *MCTest* and its dependency-based reordering:

> *Jenny, Mrs. Mustard 's helper, called the police.*
> *the police, called Jenny helper, Mrs. 's Mustard.*

Sliding-window-based matching on the original sentence will answer the question *Who called the police?* with *Mrs. Mustard*. The dependency reordering enables the window to determine the correct answer, *Jenny*.

4.3 Combining Distributed Evidence

It is important in comprehension to synthesize information found throughout a document. *MCTest* was explicitly designed to ensure that it could not be solved by lexical techniques alone, but would instead require some form of inference or limited reasoning (Richardson et al., 2013). It therefore includes questions where the evidence for an answer spans several sentences.

To perform synthesis, our model also takes in n-grams of sentences, *i.e.*, sentence pairs and triples strung together. The model treats these exactly as it treats single sentences, applying all functions detailed above. A later pooling operation combines scores across all n-grams (including the single-sentence input). This is described in the next subsection.

[1] We experimented with assigning unique edge weights to unique relation types in the dependency graph. However, this had negligible effect. We hypothesize that this is because dependency graphs are trees, which do not have cycles.

With n-grams, the model can combine information distributed across contiguous sentences. In some cases, however, the required evidence is spread across distant sentences. To give our model some capacity to deal with this scenario, we take the top N sentences as scored by all the preceding functions, and then repeat the scoring computations, viewing these top N as a single sentence.

The reasoning behind these approaches can be explained well in a probabilistic setting. If we consider our similarity scores to model the likelihood of a text sentence given a hypothesis, $p(t_j \mid h_i)$, then the n-gram and top N approaches model a joint probability $p(t_{j_1}, t_{j_2}, \ldots, t_{j_k} \mid h_i)$. We cannot model the joint probability as a product of individual terms (score values) because distributed pieces of evidence are likely not independent.

4.4 Combining Perspectives

We use a multilayer perceptron (MLP) to combine M^{sem}, M^{word}, M^{swd}, and M^{sws}, as well as the scores for separate n-grams, as a final matching score M_i for each answer candidate. This MLP has multiple layers of staged input, because the distinct scores have different dimensionality: there is one M^{sem} and one M^{word} for each story sentence, and one M^{swd} and one M^{sws} for each application of the sliding window. The MLP's activation function is linear.

Our overall training objective is to minimize the ranking loss

$$\mathcal{L}(T, q, A) = \max(0, \mu + \max_i M_{i \neq i^*} - M_{i^*}), \tag{10}$$

where μ is a constant margin, i^* indexes the correct answer. We take the maximum over i so that we are ranking the correct answer over the best-ranked incorrect answer (of which there are three). This approach worked better than comparing the correct answer to the incorrect answers individually as in Wang et al. (2015).

Our implementation of the Parallel-Hierarchical model, built in *Theano* (Bergstra et al., 2010) using the *Keras* framework (Chollet, 2015), is available on `Github`.[2]

4.5 Training Wheels

Before training, we initialized the neural-network components of our model to perform sensible heuristic functions. Training did not converge on the small *MCTest* without this vital approach.

[2]https://github.com/Maluuba/mctest-model

Empirically, we found that we could achieve above 50% accuracy on *MCTest* using a simple sum of word vectors followed by a dot product between the story-sentence sum and the hypothesis sum. Therefore, we initialized the network for the semantic perspective to perform this sum, by initializing $\mathbf{A}^x$ as the *identity* matrix and $\mathbf{b}_A^x$ as the zero vector, $x \in \{t, h\}$. Recall that the activation function is a $ReLU$ so that positive outputs are unchanged.

We also found basic word-matching scores to be helpful, so we initialized the word-by-word networks likewise. The network for perspective-combination was initialized to perform a sum of individual scores, using a *zero* bias-vector and a weight matrix of *ones*, since we found that each perspective contributed positively to the overall result.

This *training wheels* approach is related to other techniques from the literature. For instance, Socher et al. (2013) proposed the identity-matrix initialization in the context of parsing, and Le et al. (2015) proposed it in the context of recurrent neural networks (to preserve the error signal through backpropagation). In residual networks (He et al., 2015), shortcut connections bypass certain layers in the network so that a simpler function can be trained in conjunction with the full model.

5 Experiments

5.1 The Dataset

MCTest is a collection of 660 elementary-level children's stories and associated questions, written by human subjects. The stories are fictional, ensuring that the answer must be found in the text itself, and carefully limited to what a young child can understand (Richardson et al., 2013).

The more challenging variant consists of 500 stories with four multiple-choice questions each. Despite the elementary level, stories and questions are more natural and more complex than those found in synthetic MC datasets like *bAbI* (Weston et al., 2014) and *CNN* (Hermann et al., 2015).

MCTest is challenging because it is both complicated and small. As per Hill et al. (2015), "it is very difficult to train statistical models only on *MCTest*." Its size limits the number of parameters that can be trained, and prevents learning any complex language modeling simultaneously with the capacity to answer questions.

5.2 Training and Model Details

In this section we describe important details of the training procedure and model setup. For a complete list of hyperparameter settings, our stopword list, and other minutiæ, we refer interested readers to our `Github` repository.

For word vectors we use Google's publicly available embeddings, trained with `word2vec` on the 100-billion-word *News corpus* (Mikolov et al., 2013). These vectors are kept fixed throughout training, since we found that training them was not helpful (likely because of *MCTest*'s size). The vectors are 300-dimensional ($d = 300$).

We do not use a stopword list for the text passage, instead relying on the trainable word weights to ascribe global importance ratings to words. These weights are initialized with the inverse document frequency (IDF) statistic computed over the *MCTest* corpus.[3] However, we do use a short stopword list for questions. This list nullifies query words such as {*who, what, when, where, how*}, along with conjugations of the verbs *to do* and *to be*.

Following earlier methods, we use a heuristic to improve performance on negation questions (Sachan et al., 2015; Wang et al., 2015). When a question contains the words *which* and *not*, we negate the hypothesis ranking scores so that the minimum becomes the maximum. This heuristic leads to an improvement around 6% on the validation set.

The most important technique for training the model was the *training wheels* approach. Without this, training was not effective at all (see the ablation study in Table 2). The identity initialization requires that the network weight matrices are square ($d = D$).

We found *dropout* (Srivastava et al., 2014) to be particularly effective at improving generalization from the training to the test set, and used 0.5 as the dropout probability. Dropout occurs after all neural-network transformations, if those transformations are allowed to change with training. Our best performing model held networks at the word-by-word level fixed.

For combining distributed evidence, we used up to trigrams over sentences and our best-performing model reiterated over the top two sentences ($N = 2$).

[3]We override the IDF initialization for words like *not*, which are frequent but highly informative.

We used the *Adam* optimizer with the standard settings (Kingma and Ba, 2014) and a learning rate of 0.003. To determine the best hyperparameters we performed a search over 150 settings based on validation-set accuracy. *MCTest*'s original validation set is too small for reliable hyperparameter tuning, so, following Wang et al. (2015), we merged the training and validation sets of *MCTest*-160 and *MCTest*-500, then split them randomly into a 250-story training set and a 200-story validation set. This repartition of the data did not affect overall performance *per se*; rather, the larger validation set made it easier to choose hyperparameters because validation results were more consistent.

5.3 Results

Table 1 presents the performance of feature-engineered and neural methods on the *MCTest* test set. Accuracy scores are divided among questions whose evidence lies in a single sentence (*single*) and across multiple sentences (*multi*), and among the two variants. Clearly, *MCTest*-160 is easier.

The first three rows represent feature-engineered methods. Richardson et al. (2013) + RTE is the best-performing variant of the original baseline published along with *MCTest*. It uses a lexical sliding window and distance-based measure, augmented with rules for recognizing textual entailment. We described the methods of Sachan et al. (2015) and Wang et al. (2015) in Section 3. On *MCTest*-500, the Parallel Hierarchical model significantly outperforms these methods on *single* questions ($> 2\%$) and slightly outperforms the latter two on *multi* questions ($\approx 0.3\%$) and overall ($\approx 1\%$). The method of Wang et al. (2015) achieves the best overall result on *MCTest*-160. We suspect this is because our neural method suffered from the relative lack of training data.

The last four rows in Table 1 are neural methods that we discussed in Section 3. Performance measures are taken from Yin et al. (2016). Here we see our model outperforming the alternatives by a large margin across the board ($> 15\%$). The Neural Reasoner and the Attentive Reader are large, deep models with hundreds of thousands of parameters, so it is unsurprising that they performed poorly on *MCTest*. The specifically-designed HABCNN fared better, its convolutional architecture cutting down on the parameter count. Because there are similarities between our model and the

Method	*MCTest*-160 accuracy (%)			*MCTest*-500 accuracy (%)		
	Single (112)	Multiple (128)	All	Single (272)	Multiple (328)	All
Richardson et al. (2013) + RTE	76.78	62.50	69.16	68.01	59.45	63.33
Sachan et al. (2015)	-	-	-	67.65	67.99	67.83
Wang et al. (2015)	**84.22**	67.85	**75.27**	72.05	67.94	69.94
Attentive Reader	48.1	44.7	46.3	44.4	39.5	41.9
Neural Reasoner	48.4	46.8	47.6	45.7	45.6	45.6
HABCNN-TE	63.3	62.9	63.1	54.2	51.7	52.9
Parallel-Hierarchical	79.46	**70.31**	74.58	**74.26**	**68.29**	**71.00**

Table 1: Experimental results on *MCTest*.

Ablated component	Validation accuracy (%)
-	**70.13**
n-gram	66.25
Top N	66.63
Sentential	65.00
SW-sequential	68.88
SW-dependency	69.75
Word weights	66.88
Trainable embeddings	63.50
Training wheels	34.75

Table 2: Ablation study on *MCTest*-500 (all).

HABCNN, we hypothesize that the performance difference is attributable to the greater simplicity of our model and our *training wheels* methodology.

6 Analysis and Discussion

We measure the contribution of each component of the model by ablating it. Results on the validation set are given in Table 2. Not surprisingly, the n-gram functionality is important, contributing almost 4% accuracy improvement. Without this, the model has almost no means for synthesizing distributed evidence. The top N function contributes similarly to the overall performance, suggesting that there is a nonnegligible number of *multi* questions that have their evidence distributed across noncontiguous sentences. Ablating the sentential component made a significant difference, reducing performance by about 5%. Simple word-by-word matching is obviously useful on *MCTest*. The sequential sliding window contributes about 1.3%, suggesting that word-distance measures are not overly important. Similarly, the dependency-based sliding window makes a very minor contribution. We found this surprising. It may be that linearization of the dependency graph removes too much of its information. The exogenous word weights make a significant contribution of over 3%. Allowing the embeddings to change with training reduced performance fairly significantly, almost 8%. As discussed, this is a case of having too many parameters for the available training data. Finally, we see that the training wheels methodology had enormous impact. Without heuristic-based initialization of the model's various weight matrices, accuracy goes down to about 35%, which is only ten points over random chance.

Analysis reveals that most of our system's test failures occur on questions about quantity (*e.g.*, *How many...?*) and temporal order (*e.g.*, *Who was invited last?*). Quantity questions make up 9.5% of our errors on the validation set, while order questions make up 10.3%. This weakness is not unexpected, since our architecture lacks any capacity for counting or tracking temporal order. Incorporating mechanisms for these forms of reasoning is a priority for future work (in contrast, the Memory Network model (Weston et al., 2014) is quite good at temporal reasoning).

The Parallel-Hierarchical model is simple. It does no complex language or sequence modeling. Its simplicity is a response to the limited data of *MCTest*. Nevertheless, the model achieves state-of-the-art results on the *multi* questions, which (putatively) require some limited reasoning. Our model is able to handle them reasonably well just by stringing important sentences together. Thus, the model imitates reasoning with a heuristic. This suggests that, to learn true reasoning abilities, *MCTest* is too simple a dataset—and it is almost certainly too small for this goal.

However, it may be that human language processing can be factored into separate processes of *comprehension* and *reasoning*. If so, the Parallel-Hierarchical model is a good start on the former. Indeed, if we train the method exclusively on *single* questions then its results become even more impressive: we can achieve a test accuracy of **79.1%** on *MCTest*-500. Note that this boost in performance comes from training on only about *half* the data. The 'single' questions can be con-

sidered a close analogue of the RTE task, at which our model becomes very adept even with less data.

Incorporating the various views of our model amounts to encoding prior knowledge about the problem structure. This is similar to the purpose of feature engineering, except that the views can be fully trained. Encoding problem structure into the structure of neural networks is not new: as another example, the convolutional architecture has led to large gains in vision tasks.

7 Conclusion

We have presented the novel Parallel-Hierarchical model for machine comprehension, and evaluated it on the small but complex *MCTest*. Our model achieves state-of-the-art results, outperforming several feature-engineered and neural approaches.

Working with our model has emphasized to us the following (not necessarily novel) concepts, which we record here to promote further empirical validation.

- Good comprehension of language is supported by hierarchical levels of understanding (*cf.* Hill et al. (2015)).
- Exogenous attention (the trainable word weights) may be broadly helpful for NLP.
- The *training wheels* approach, that is, initializing neural networks to perform sensible heuristics, appears helpful for small datasets.
- *Reasoning* over language is challenging, but easily simulated in some cases.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *In Proc. of SciPy*.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

François Chollet. 2015. keras. `https://github.com/fchollet/keras`.

Silviu Cucerzan and Eugene Agichtein. 2005. Factoid question answering over unstructured and structured web content. In *TREC*, volume 72, page 90.

Gene H Golub and Charles F Van Loan. 2012. *Matrix computations*, volume 3. JHU Press.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*.

Andrew R Mayer, Jill M Dorflinger, Stephen M Rao, and Michael Seidenberg. 2004. Neural networks underlying endogenous and exogenous visual–spatial orienting. *Neuroimage*, 23(2):534–541.

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

Karthik Narasimhan and Regina Barzilay. 2015. Machine comprehension with discourse relations. In *53rd Annual Meeting of the Association for Computational Linguistics*.

Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. 2015. Towards neural network-based reasoning. *arXiv preprint arXiv:1508.05508*.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2.

Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answerentailing structures for machine comprehension. In *Proceedings of ACL*.

Ellery Smith, Nicola Greco, Matko Bosnjak, and Andreas Vlachos. 2015. A strong lexical matching method for the machine comprehension test. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1693–1698, Lisbon, Portugal, September. Association for Computational Linguistics.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Proceedings of ACL, Volume 2: Short Papers*, page 700.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. 2016. Attention-based convolutional neural network for machine comprehension. *arXiv preprint arXiv:1602.04341*.

Combining Natural Logic and Shallow Reasoning for Question Answering

Gabor Angeli
Stanford University
Stanford, CA 94305

Neha Nayak
Stanford University
Stanford, CA 94305

Christopher D. Manning
Stanford University
Stanford, CA 94305

{angeli, nayakne, manning}@cs.stanford.edu

Abstract

Broad domain question answering is often difficult in the absence of structured knowledge bases, and can benefit from shallow lexical methods (broad coverage) and logical reasoning (high precision). We propose an approach for incorporating both of these signals in a unified framework based on natural logic. We extend the breadth of inferences afforded by natural logic to include relational entailment (e.g., *buy* → *own*) and meronymy (e.g., a person born in a city is born the city's country). Furthermore, we train an *evaluation function* – akin to gameplaying – to evaluate the expected truth of candidate premises on the fly. We evaluate our approach on answering multiple choice science questions, achieving strong results on the dataset.

1 Introduction

Question answering is an important task in NLP, and becomes both more important and more difficult when the answers are not supported by hand-curated knowledge bases. In these cases, viewing question answering as textual entailment over a very large premise set can offer a means of generalizing reliably to open domain questions.

A natural approach to textual entailment is to treat it as a logical entailment problem. However, this high-precision approach is not feasible in cases where a formal proof is difficult or impossible. For example, consider the following hypothesis (H) and its supporting premise (P) for the question *Which part of a plant produces the seeds?*:

P: *Ovaries are the female part of the flower, which produces eggs that are needed for making seeds.*

H: *A flower produces the seeds.*

This requires a relatively large amount of inference: the most natural atomic fact in the sentence is that ovaries produce eggs. These inferences are feasible in a limited domain, but become difficult the more open-domain reasoning they require. In contrast, even a simple lexical overlap classifier could correctly predict the entailment. In fact, such a bag-of-words entailment model has been shown to be surprisingly effective on the Recognizing Textual Entailment (RTE) challenges (MacCartney, 2009). On the other hand, such methods are also notorious for ignoring even trivial cases of nonentailment that are easy for natural logic, e.g., recognizing negation in the example below:

P: *Eating candy for dinner is an example of a poor health habit.*

H: *Eating candy is an example of a good health habit.*

We present an approach to leverage the benefits of both methods. Natural logic – a proof theory over the syntax of natural language – offers a framework for logical inference which is already familiar to lexical methods. As an inference system searches for a valid premise, the candidates it explores can be evaluated on their similarity to a premise by a conventional lexical classifier.

We therefore extend a natural logic inference engine in two key ways: first, we handle relational entailment and meronymy, increasing the total number of inferences that can be made. We further implement an *evaluation function* which quickly provides an estimate for how likely a candidate premise is to be supported by the knowledge base, without running the full search. This can then more easily match a known premise despite still not matching exactly.

We present the following contributions: (1) we extend the classes of inferences NaturalLI can perform on real-world sentences by incorporating relational entailment and meronymy, and by operat-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 442–452,
Berlin, Germany, August 7-12, 2016.

ing over dependency trees; (2) we augment NaturalLI with an evaluation function to provide an estimate of entailment for any query; and (3) we run our system over the Aristo science questions corpus, achieving the strong results.

2 Background

We briefly review natural logic and NaturalLI – the existing inference engine we use. Much of this paper will extend this system, with additional inferences (Section 3) and a soft lexical classifier (Section 4).

2.1 Natural Logic

Natural logic is a formal proof theory that aims to capture a subset of logical inferences by appealing directly to the structure of language, without needing either an abstract logical language (e.g., Markov Logic Networks; Richardson and Domingos (2006)) or denotations (e.g., semantic parsing; Liang and Potts (2015)). We use the logic introduced by the NatLog system (MacCartney and Manning, 2007; 2008; 2009), which was in turn based on earlier theoretical work on Monotonicity Calculus (van Benthem, 1986; Sánchez Valencia, 1991). We adopt the precise semantics of Icard and Moss (2014); we refer the reader to this paper for a more thorough introduction to the formalism.

At a high level, natural logic proofs operate by mutating spans of text to ensure that the mutated sentence follows from the original – each step is much like a syllogistic inference. Each mutation in the proof follows three steps:

1. An atomic lexical relation is induced by either inserting, deleting or mutating a span in the sentence. For example, in Figure 1, mutating *The* to *No* induces the $\curlywedge$ relation; mutating *cat* to *carnivore* induces the $\sqsubseteq$ relation. The relations $\equiv$ and $\sqsubseteq$ are variants of entailment; $\curlywedge$ and $\Downarrow$ are variants of negation.
2. This lexical relation between words is projected up to yield a relation between sentences, based on the *polarity* of the token. For instance, *The cat eats animals* $\sqsubseteq$ *some carnivores eat animals*. We explain this in more detail below.
3. These sentence level relations are *joined* together to produce a relation between a premise, and a hypothesis multiple mutations away. For example in Figure 1, if we join $\sqsubseteq$, $\equiv$, $\sqsubseteq$, and $\curlywedge$, we get negation ($\Downarrow$).

The notion of *projecting* a relation from a lexical item to a sentence is important to understand.[1] To illustrate, *cat* $\sqsubseteq$ *animal*, and *some cat meows* $\sqsubseteq$ *some animal meows* (recall, $\sqsubseteq$ denotes entailment), but *no cat barks* $\not\sqsubseteq$ *no animal barks*. Despite differing by the same lexical relation, the sentence-level relation is different in the two cases.

We appeal to two important concepts: *monotonicity* – a property of arguments to natural language operators; and *polarity* – a property of tokens. From the example above, *some* is monotone in its first argument (i.e., *cat* or *animal*), and *no* is antitone in its first argument. This means that the first argument to *some* is allowed to mutate up the specified hierarchy (e.g., hypernymy), whereas the first argument to *no* is allowed to mutate down.

Polarity is a property of tokens in a sentence determined by the operators acting on it. All lexical items have *upward* polarity by default; monotone operators – like *some*, *several*, or *a few* – preserve polarity. Antitone operators – like *no*, *not*, and *all* (in its first argument) – reverse polarity. For example, *mice* in ***no*** *cats eat mice* has downward polarity, whereas *mice* in ***no*** *cats do**n't*** *eat mice* has upward polarity (it is in the scope of two downward monotone operators).

As a final note, although we refer to the monotonicity calculus described above as *natural logic*, this formalism is only one of many possible natural logics. For example, McAllester and Givan (1992) introduce a syntax for first order logic which they call *Montagovian syntax*. This syntax has two key advantages over first order logic: first, the "quantifier-free" version of the syntax (roughly equivalent to the monotonicity calculus we use) is computationally efficient while still handling limited quantification. Second, the syntax more closely mirrors that of natural language.

2.2 NaturalLI

We build our extensions within the framework of NaturalLI, introduced by Angeli and Manning (2014). NaturalLI casts inference as a search problem: given a hypothesis and an arbitrarily large corpus of text, it searches through the space of lexical mutations (e.g., *cat* $\rightarrow$ *carnivore*), with associated costs, until a premise is found.

An example search using NaturalLI is given in Figure 1. The relations along the edges denote re-

[1] For clarity we describe a simplified semantics here; NaturalLI implements the semantics described in Icard and Moss (2014).

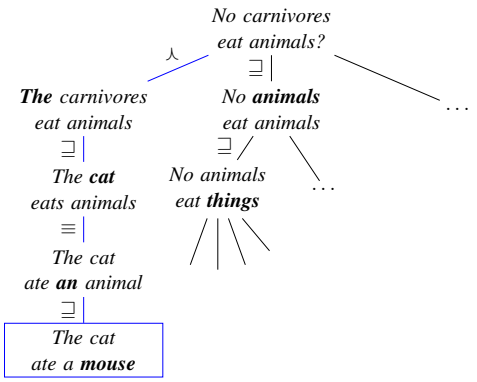


Figure 1: An illustration of NaturalLI searching for a candidate premise to support the hypothesis at the root of the tree. We are searching from a hypothesis *no carnivores eat animals*, and find a contradicting premise *the cat ate a mouse*. The edge labels denote Natural Logic inference steps.

lations between the associated sentences – i.e., the projected lexical relations from Section 2.2. Importantly, and in contrast with traditional entailment systems, NaturalLI searches over an arbitrarily large knowledge base of textual premises rather than a single premise/hypothesis pair.

3 Improving Inference in NaturalLI

We extend NaturalLI in three ways to improve its coverage. We adapt the search algorithm to operate over dependency trees rather than the surface forms (Section 3.1). We enrich the class of inferences warranted by natural logic beyond hypernymy and operator rewording to also encompass meronymy and relational entailment (Section 3.2). Lastly, we handle token insertions during search more elegantly (Section 3.3).

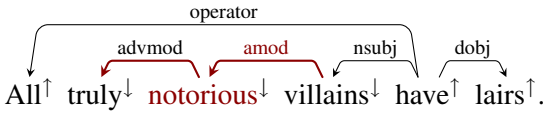
The general search algorithm in NaturalLI is parametrized as follows: First, an order is chosen to traverse the tokens in a sentence. For example, the original paper traverses tokens left-to-right. At each token, one of three operations can be performed: *deleting* a token (corresponding to inserting a word in the proof derivation), *mutating* a token, and *inserting* a token (corresponding to deleting a token in the proof derivation).

3.1 Natural logic over Dependency Trees

Operating over dependency trees rather than a token sequence requires reworking (1) the semantics of deleting a token during search, and (2) the order in which the sentence is traversed.

We recently defined a mapping from Stanford Dependency relations to the associated lexical relation deleting the dependent subtree would induce (Angeli et al., 2015). We adapt this mapping to yield the relation induced by *inserting* a given dependency edge, corresponding to our deletions in search; we also convert the mapping to use Universal Dependencies (de Marneffe et al., 2014). This now lends a natural deletion operation: at a given node, the subtree rooted at that node can be deleted to induce the associated natural logic relation.

For example, we can infer that *all truly notorious villains have lairs* from the premise *all villains have lairs* by observing that deleting an *amod* arc induces the relation $\sqsupseteq$, which in the downward polarity context of *villains*$^{\downarrow}$ projects to $\sqsubseteq$ (entailment):



An admittedly rare but interesting subtlety in the order we chose to traverse the tokens in the sentence is the effect mutating an operator has on the polarity of its arguments. For example, mutating *some* to *all* changes the polarity of its first argument. There are cases where we must mutate the argument to the operator before the operator itself, as well as cases where we must mutate the operator before its arguments. Consider, for instance:

P: *All felines have a tail*
H: *Some cats have a tail*

where we must first mutate *cat* to *feline*, versus:

P: *All cats have a tail*
H: *Some felines have a tail*

where we must first mutate *some* to *all*. Therefore, our traversal first visits each operator, then performs a breadth-first traversal of the tree, and then visits each operator a second time.

3.2 Meronymy and Relational Entailment

Although natural logic and the underlying monotonicity calculus has only been explored in the context of hypernymy, the underlying framework can be applied to any partial order.

Natural language operators can be defined as a mapping from denotations of objects to truth values. The domain of word denotations is then or-

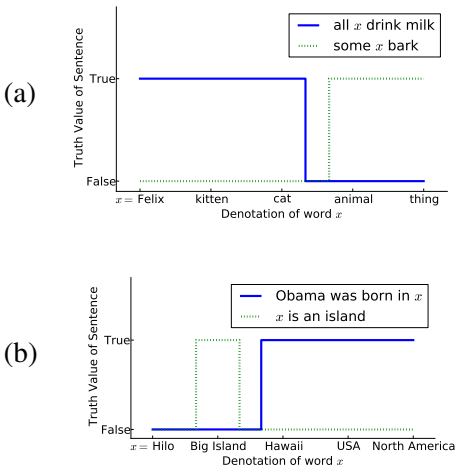


Figure 2: An illustration of monotonicity using different partial orders. (a) The monotonicity of *all* and *some* in their first arguments, over a domain of denotations. (b) An illustration of the *born in* monotone operator over the meronymy hierarchy, and the operator *is an island* as neither monotone or antitone.

dered by the subset operator, corresponding to ordering by hypernymy over the words.[2] However, hypernymy is not the only useful partial ordering over denotations. We include two additional orderings as motivating examples: relational entailment and meronymy.

Relational Entailment For two verbs v_1 and v_2, we define $v_1 \leq v_2$ if the first verb entails the second. In many cases, a verb v_1 may entail a verb v_2 even if v_2 is not a hypernym of v_1. For example, to *sell* something (hopefully) entails *own*ing that thing. Apart from context-specific cases (e.g., *orbit* entails *launch* only for man-made objects), these hold largely independent of context. Note that the usual operators apply to relational entailments – if *all cactus owners live in Arizona* then *all cactus sellers live in Arizona*.

This information was incorporated using data from VERBOCEAN (Chklovski and Pantel, 2004), adapting the confidence weights as transition costs. VERBOCEAN uses lexicosyntactic patterns to score pairs of verbs as candidate participants in a set of relations. We approximate the VERBOCEAN relations $\textit{stronger-than}(v_1, v_2)$ (e.g., to kill is stronger than to wound) and $\textit{happens-before}(v_2, v_1)$ (e.g., buying happens before owning) to indicate that v_1 entails v_2. These verb entailment transitions are incorporated using costs derived from the original weights from Chklovski and Pantel (2004).

Meronymy The most salient use-case for meronymy is with locations. For example, if Obama was born in Hawaii, then we know that Obama was born in America, because Hawaii is a meronym of (part of) America. Unlike relational entailment and hypernymy, meronymy is operated on by a distinct set of operators: if *Hawaii is an island*, we cannot necessarily entail that *America is an island*.

We semi-automatically collect a set of 81 operators (e.g., *born in*, *visited*) which then compose in the usual way with the conventional operators (e.g., *some*, *all*). These operators consist of dependency paths of length 2 that co-occurred in newswire text with a named entity of type *PERSON* and two different named entities of type *LOCATION*, such that one location was a meronym of the other. All other operators are considered non-monotone with respect to the meronym hierarchy.

Note that these are not the only two orders that can be incorporated into our framework; they just happen to be two which have lexical resources available and are likely to be useful in real-world entailment tasks.

3.3 Removing the Insertion Transition

Inserting words during search poses an inherent problem, as the space of possible words to insert at any position is on the order of the size of the vocabulary. In NaturalLI, this was solved by keeping a trie of possible insertions, and using that to prune this space. This is both computationally slow and adapts awkwardly to a search over dependency trees.

Therefore, this work instead opts to perform a bidirectional search: when constructing the knowledge base, we add not only the original sentence but also all entailments with subtrees deleted. For example, a premise of *some furry cats have tails* would yield two facts for the knowledge base: *some furry cats have tails* as well as *some cats have tails*. For this, we use the process described in Angeli et al. (2015) to generate short entailed sentences from a long utterance using natural logic. This then leaves the reverse search to only deal with mutations and inference insertions,

[2] Truth values are a trivial partial order corresponding to entailment: if $t_1 \leq t_2$ (i.e., $t_1 \sqsubseteq t_2$), and you know that t_1 is true, then t_2 must be true.

which are relatively easier.

The new challenge this introduces, of course, is the additional space required to store the new facts. To mitigate this, we hash every fact into a 64 bit integer, and store only the hashed value in the knowledge base. We construct this hash function such that it operates over a bag of edges in the dependency tree. This has two key properties: it allows us to be invariant to the word order of of the sentence, and more importantly it allows us to run our search directly over modifications to this hash function.

To elaborate, we notice that each of the two classes of operations our search is performing are done locally over a single dependency edge. When adding an edge, we can simply take the XOR of the hash saved in the parent state and the hash of the added edge. When mutating an edge, we XOR the hash of the parent state with the edge we are mutating, and again with the mutated edge. In this way, each search node need only carry an 8 byte hash, local information about the edge currently being considered (8 bytes), global information about the words deleted during search (5 bytes), a 3 byte backpointer to recover the inference path, and 8 bytes of operator metadata – 32 bytes in all, amounting to exactly half a cache line on our machines. This careful attention to data structures and memory layout turn out to have a large impact on runtime efficiency. More details are given in Angeli (2016).

4 An Evaluation Function for NaturalLI

There are many cases – particularly as the length of the premise and the hypothesis grow – where despite our improvements NaturalLI will fail to find any supporting premises; for example:

P: *Food serves mainly for growth, energy and body repair, maintenance and protection.*

H: *Animals get energy for growth and repair from food.*

In addition to requiring reasoning with multiple implicit premises (a concomitant weak point of natural logic), a correct interpretation of the sentence requires fairly nontrivial nonlocal reasoning: *Food serves mainly for* $x \rightarrow$ *Animals get* x *from food*.

Nonetheless, there enough lexical clues in the sentence that even a simple entailment classifier would get the example correct. We build such a classifier and adapt it as an evaluation function inside NaturalLI in case no premises are found during search.

4.1 A Standalone Entailment Classifier

Our entailment classifier is designed to be as domain independent as possible; therefore we define only 5 unlexicalized real-valued features, with an optional sixth feature encoding the score output by the Solr information extraction system (in turn built upon Lucene). In fact, this classifier is a stronger baseline than it may seem: evaluating the system on RTE-3 (Giampiccolo et al., 2007) yielded **63.75**% accuracy – 2 points above the median submission.

All five of the core features are based on an alignment of keyphrases between the premise and the hypothesis. A keyphrase is defined as a span of text which is either (1) a possibly empty sequence of adjectives and adverbs followed by a sequence of nouns, and optionally followed by either *of* or the possessive marker (*'s*), and another noun (e.g., *sneaky kitten* or *pail of water*); (2) a possibly empty sequence of adverbs followed by a verb (e.g., *quietly pounce*); or (3) a gerund followed by a noun (e.g., *flowing water*). The verb *to be* is never a keyphrase. We make a distinction between a *keyphrase* and a *keyword* – the latter is a single noun, adjective, or verb.

We then align keyphrases in the premise and hypothesis by applying a series of sieves. First, all exact matches are aligned to each other. Then, prefix or suffix matches are aligned, then if either keyphrase contains the other they are aligned as well. Last, we align a keyphrase in the premise p_i to a keyphrase in the hypothesis h_k if there is an alignment between p_{i-1} and h_{k-1} and between p_{i+1} and h_{k+1}. This forces any keyphrase pair which is "sandwiched" between aligned pairs to be aligned as well. An example alignment is given in Figure 3.

Features are extracted for the number of alignments, the numbers of alignments which do and do not match perfectly, and the number of keyphrases in the premise and hypothesis which were not aligned. A feature for the Solr score of the premise given the hypothesis is optionally included; we revisit this issue in the evaluation.

4.2 An Evaluation Function for Search

A version of the classifier constructed in Section 4.1, but over *keywords* rather than keyphrases can be incorporated directly into NaturalLI's search to give a score for each candidate premise

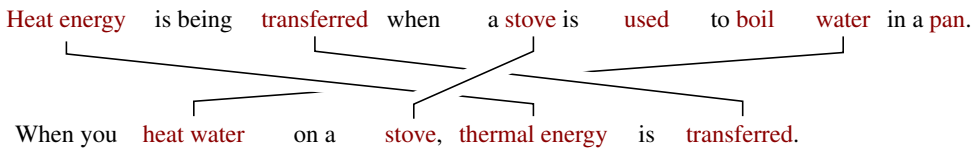


Figure 3: An illustration of an alignment between a premise and a hypothesis. Keyphrases can be multiple words (e.g., *heat energy*), and can be approximately matched (e.g., to *thermal energy*). In the premise, *used*, *boil* and *pan* are unaligned. Note that *heat water* is incorrectly tagged as a compound noun.

visited. This can be thought of as analogous to the evaluation function in game-playing search – even though an agent cannot play a game of Chess to completion, at some depth it can apply an evaluation function to its leaf states.

Using keywords rather than keyphrases is in general a hindrance to the fuzzy alignments the system can produce. Importantly though, this allows the feature values to be computed incrementally as the search progresses, based on the score of the parent state and the mutation or deletion being performed. For instance, if we are deleting a word which was previously aligned perfectly to the premise, we would subtract the weight for a perfect and imperfect alignment, and add the weight for an unaligned premise keyphrase. This has the same effect as applying the trained classifier to the new state, and uses the same weights learned for this classifier, but requires substantially less computation.

In addition to finding entailments from candidate premises, our system also allows us to encode a notion of likely negation. We can consider the following two statements naïvely sharing every keyword. Each token marked with its polarity:

P: *some*$^{\uparrow}$ *cats*$^{\uparrow}$ *have*$^{\uparrow}$ *tails*$^{\uparrow}$
H: *no*$^{\uparrow}$ *cats*$^{\downarrow}$ *have*$^{\downarrow}$ *tails*$^{\downarrow}$

However, we note that all of the keyword pairs are in opposite polarity contexts. We can therefore define a pair of keywords as *matching* in NaturalLI if the following two conditions hold: (1) their lemmatized surface forms match exactly, and (2) they have the same polarity in the sentence. The second constraint encodes a good approximation for negation. To illustrate, consider the polarity signatures of common operators:

Operators	**Subj. polarity**	**Obj. polarity**
Some, few, etc.	$\uparrow$	$\uparrow$
All, every, etc.	$\downarrow$	$\uparrow$
Not all, etc.	$\uparrow$	$\downarrow$
No, not, etc.	$\downarrow$	$\downarrow$
Most, many, etc.	–	$\uparrow$

We note that most contradictory operators (e.g., *some*/*no*; *all*/*not all*) induce the exact opposite polarity on their arguments. Otherwise, pairs of operators which share half their signature are usually compatible with each other (e.g., *some* and *all*).

This suggests a criterion for likely negation: If the highest classifier score is produced by a contradictory candidate premise, we have reason to believe that we may have found a contradiction. To illustrate with our example, NaturalLI would mutate *no cats have tails* to *the cats have tails*, at which point it has found a contradictory candidate premise which has perfect overlap with the premise *some cats have tails*. Even had we not found the exact premise, this suggests that the hypothesis is likely false.

5 Related Work

This work is similar in many ways to work on recognizing textual entailment – e.g., Schoenmackers et al. (2010), Berant et al. (2011), Lewis and Steedman (2013). In the RTE task, a single premise and a single hypothesis are given as input, and a system must return a judgment of either *entailment* or *nonentailment* (in later years, *nonentailment* is further split into contradiction and independence). These approaches often rely on alignment features, similar to ours, but do not generally scale to large premise sets (i.e., a comprehensive knowledge base). The discourse commitments in Hickl and Bensley (2007) can be thought of as similar to the additional entailed facts we add to the knowledge base (Section 3.3). In another line of work, Tian et al. (2014) approach the

RTE problem by parsing into Dependency Compositional Semantics (DCS) (Liang et al., 2011). This work particularly relevant in that it also incorporates an evaluation function (using distributional similarity) to augment their theorem prover – although in their case, this requires a translation back and forth between DCS and language. Beltagy et al. (To appear 2016) takes a similar approach, but encoding distributional information directly in entailment rules in a Markov Logic Network (Richardson and Domingos, 2006).

Many systems make use of structured knowledge bases for question answering. Semantic parsing methods (Zettlemoyer and Collins, 2005; Liang et al., 2011) use knowledge bases like Freebase to find support for a complex question. Knowledge base completion (e.g., Chen et al. (2013), Bordes et al. (2011), or Riedel et al. (2013)) can be thought of as entailment, predicting novel knowledge base entries from the original database. In contrast, this work runs inference over arbitrary text without needing a structured knowledge base. Open IE (Wu and Weld, 2010; Mausam et al., 2012) QA approaches – e.g., Fader et al. (2014) are closer to operating over plain text, but still requires structured extractions.

Of course, this work is not alone in attempting to incorporate strict logical reasoning into question answering systems. The COGEX system (Moldovan et al., 2003) incorporates a theorem prover into a QA system, boosting overall performance on the TREC QA task. Similarly, Watson (Ferrucci et al., 2010) incorporates logical reasoning components alongside shallower methods. This work follows a similar vein, but both the theorem prover and lexical classifier operate over text, without requiring either the premises or axioms to be in logical forms.

On the Aristo corpus we evaluate on, Hixon et al. (2015) proposes a dialog system to augment a knowledge graph used for answering the questions. This is in a sense an oracle measure, where a human is consulted while answering the question; although, they show that their additional extractions help answer questions other than the one the dialog was collected for.

6 Evaluation

We evaluate our entailment system on the Regents Science Exam portion of the Aristo dataset (Clark et al., 2013; Clark, 2015). The dataset consists of a collection of multiple-choice science questions from the New York Regents 4th Grade Science Exams (NYSED, 2014). Each multiple choice option is translated to a candidate hypotheses. A large corpus is given as a knowledge base; the task is to find support in this knowledge base for the hypothesis.

Our system is in many ways well-suited to the dataset. Although certainly many of the facts require complex reasoning (see Section 6.4), the majority can be answered from a single premise. Unlike FraCaS (Cooper et al., 1996) or the RTE challenges, however, the task does not have explicit premises to run inference from, but rather must infer the truth of the hypothesis from a large collection of supporting text.

6.1 Data Processing

We make use of two collections of unlabeled corpora for our experiments. The first of these is the Barron's study guide (BARRON'S), consisting of 1200 sentences. This is the corpus used by Hixon et al. (2015) for their conversational dialog engine Knowbot, and therefore constitutes a more fair comparison against their results. However, we also make use of the full SCITEXT corpus (Clark et al., 2014). This corpus consists of 1 316 278 supporting sentences, including the Barron's study guide alongside simple Wikipedia, dictionaries, and a science textbook.

Since we lose all document context when searching over the corpus with NaturalLI, we first pre-process the corpus to resolve high-precision cases of pronominal coreference, via a set of very simple high-precision sieves. This finds the most recent candidate antecedent (NP or named entity) which, in order of preference, matches either the pronoun's animacy, gender, and number. Filtering to remove duplicate sentences and sentences containing non-ASCII characters yields a total of 822 748 facts in the corpus.

These sentences were then indexed using Solr. The set of promising premises for the soft alignment in Section 4, as well as the Solr score feature in the lexical classifier (Section 4.1), were obtained by querying Solr using the default similarity metric and scoring function. On the query side, questions were converted to answers using the same methodology as Hixon et al. (2015). In cases where the question contained multiple sentences, only the last sentence was considered. As

discussed in Section 6.4, we do not attempt reasoning over multiple sentences, and the last sentence is likely the most informative sentence in a longer passage.

6.2 Training an Entailment Classifier

To train a soft entailment classifier, we needed a set of positive and negative entailment instances. These were collected on Mechanical Turk. In particular, for each true hypothesis in the training set and for each sentence in the Barron's study guide, we found the top 8 results from Solr and considered these to be candidate entailments. These were then shown to Turkers, who decided whether the premise entailed the hypothesis, the hypothesis entailed the premise, both, or neither. Note that each pair was shown to only one Turker, lowering the cost of data collection, but consequently resulting in a somewhat noisy dataset. The data was augmented with additional negatives, collected by taking the top 10 Solr results for each false hypothesis in the training set. This yielded a total of 21 306 examples.

The scores returned from NaturalLI incorporate negation in two ways: if NaturalLI finds a contradictory premise, the score is set to zero. If NaturalLI finds a soft negation (see Section 4.2), and did not find an explicit supporting premise, the score is discounted by 0.75 – a value tuned on the training set. For all systems, any premise which did not contain the candidate answer to the multiple choice query was discounted by a value tuned on the training set.

6.3 Experimental Results

We present results on the Aristo dataset in Table 1, alongside prior work and strong baselines. In all cases, NaturalLI is run with the evaluation function enabled; the limited size of the text corpus and the complexity of the questions would cause the basic NaturalLI system to perform poorly. The test set for this corpus consists of only 68 examples, and therefore both perceived large differences in model scores and the apparent best system should be interpreted cautiously. NaturalLI consistently achieves the best training accuracy, and is more stable between configurations on the test set. For instance, it may be consistently discarding lexically similar but actually contradictory premises that often confuse some subset of the baselines.

KNOWBOT is the dialog system presented in Hixon et al. (2015). We report numbers for two variants of the system: *held-out* is the system's performance when it is not allowed to use the dialog collected from humans for the example it is answering; *oracle* is the full system. Note that the *oracle* variant is a human-in-the-loop system.

System	Barron's		SCITEXT	
	Train	Test	Train	Test
KNOWBOT (held-out)	45	–	–	–
KNOWBOT (oracle)	57	–	–	–
Solr Only	49	42	62	58
Classifier	53	52	68	60
+ Solr	53	48	66	64
Evaluation Function	52	**54**	61	63
+ Solr	50	45	62	58
NaturalLI	52	51	65	61
+ Solr	**55**	49	73	61
+ Solr + Classifier	**55**	49	**74**	**67**

Table 1: Accuracy on the Aristo science questions dataset. All NaturalLI runs include the evaluation function. Results are reported using only the Barron's study guide or SCITEXT as the supporting KNOWBOT is the dialog system presented in Hixon et. al (2015). The held-out version uses additional facts from other question's dialogs; the oracle version made use of human input on the question it was answering. The test set did not exist at the time KNOWBOT was published.

We additionally present three baselines. The first simply uses Solr's IR confidence to rank entailment (*Solr Only* in Table 1). The max IR score of any premise given a hypothesis is taken as the score for that hypothesis. Furthermore, we report results for the entailment classifier defined in Section 4.1 (*Classifier*), optionally including the Solr score as a feature. We also report performance of the evaluation function in NaturalLI applied directly to the premise and hypothesis, without any inference (*Evaluation Function*).

Last, we evaluate NaturalLI with the improvements presented in this paper (*NaturalLI* in Table 1). We additionally tune weights on our training set for a simple model combination with (1) Solr (with weight 6:1 for NaturalLI) and (2) the standalone classifier (with weight 24:1 for NaturalLI). Empirically, both parameters were observed to be fairly robust.

To demonstrate the system's robustness on a larger dataset, we additionally evaluate on a test set of 250 additional science exam questions, with

System	Test Accuracy
Solr Only	46.8
Classifier	43.6
NaturalLI	46.4
+ Solr	**48.0**

Table 2: Results of our baselines and NaturalLI on a larger dataset of 250 examples. All NaturalLI runs include the evaluation function.

an associated 500 example training set (and 249 example development set). These are substantially more difficult as they contain a far larger number of questions that require an understanding of a more complex process. Nonetheless, the trend illustrated in Table 1 holds for this larger set, as shown in Table 2. Note that with a web-scale corpus, accuracy of an IR-based system can be pushed up to 51.4%; a PMI-based solver, in turn, achieves an accuracy of 54.8% – admittedly higher than our best system (Clark et al., 2016).[3] An interesting avenue of future work would be to run NaturalLI over such a large web-scale corpus, and to incorporate PMI-based statistics into the evaluation function.

6.4 Discussion

We analyze some common types of errors made by the system on the training set. The most common error can be attributed to the question requiring complex reasoning about multiple premises. 29 of 108 questions in the training set (26%) contain multiple premises. Some of these cases can be recovered from (e.g., *This happens because the smooth road has less friction.*), while others are trivially out of scope for our method (e.g., *The volume of water most likely decreased.*). Although there is usually still some signal for which answer is most likely to be correct, these questions are fundamentally out-of-scope for the approach.

Another class of errors which deserves mention are cases where a system produces the same score for multiple answers. This occurs fairly frequently in the standalone classifier (7% of examples in training; 4% loss from random guesses), and especially often in NaturalLI (11%; 6% loss from random guesses). This offers some insight into why incorporating other models – even with low weight – can offer significant boosts in the performance of NaturalLI. Both this and the previous class could be further mitigated by having a notion of a *process*, as in Berant et al. (2014).

Other questions are simply not supported by any single sentence in the corpus. For example, *A human offspring can inherit blue eyes* has no support in the corpus that does not require significant multi-step inferences.

A remaining chunk of errors are simply classification errors. For example, *Water freezing is an example of a gas changing to a solid* is marked as the best hypothesis, supported incorrectly by *An ice cube is an example of matter that changes from a solid to a liquid to a gas*, which after mutating *water* to *ice cube* matches every keyword in the hypothesis.

7 Conclusion

We have improved NaturalLI to be more robust for question answering by running the inference over dependency trees, pre-computing deletions, and incorporating a soft evaluation function for predicting likely entailments when formal support could not be found. Lastly, we show that relational entailment and meronymy can be elegantly incorporated into natural logic. These features allow us to perform large-scale broad domain question answering, achieving strong results on the Aristo science exams corpus.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments. We gratefully acknowledge the support of the Allen Institute for Artificial Intelligence, and in particular Peter Clark and Oren Etzioni for valuable discussions, as well as for access to the Aristo corpora and associated preprocessing. We would also like to acknowledge the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of AI2, DARPA, AFRL, or the US government.

[3] Results from personal correspondence with the authors.

References

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *EMNLP*.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *ACL*.

Gabor Angeli. 2016. *Learning Open Domain Knowledge From Text*. Ph.D. thesis, Stanford University.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. To appear, 2016. Representing meaning with a combination of logical and distributional models. *Computational Linguistics*.

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *Proceedings of ACL*, Portland, OR.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Brad Huang, Christopher D Manning, Abby Vander Linden, Brittany Harding, and Peter Clark. 2014. Modeling biological processes for reading comprehension. In *Proc. EMNLP*.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *AAAI*.

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*.

Timothy Chklovski and Patrick Pantel. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *EMNLP*.

Peter Clark, Philip Harrison, and Niranjan Balasubramanian. 2013. A study of the knowledge base requirements for passing an elementary science test. In *AKBC*.

Peter Clark, Niranjan Balasubramanian, Sumithra Bhakthavatsalam, Kevin Humphreys, Jesse Kinkead, Ashish Sabharwal, and Oyvind Tafjord. 2014. Automatic construction of inference-supporting knowledge bases. *AKBC*.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions.

Peter Clark. 2015. Elementary school science and math tests as a driver for AI: Take the Aristo challenge! *AAAI*.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. Using the framework. Technical report, The FraCaS Consortium.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *KDD*.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, Jmes Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. The AI behind Watson. *The AI Magazine*.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL workshop on textual entailment and paraphrasing*. Association for Computational Linguistics.

Andrew Hickl and Jeremy Bensley. 2007. A discourse commitment-based framework for recognizing textual entailment. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Ben Hixon, Peter Clark, and Hannaneh Hajishirzi. 2015. Learning knowledge graphs for question answering through conversational dialog. *NAACL*.

Thomas Icard, III and Lawrence Moss. 2014. Recent progress on monotonicity. *Linguistic Issues in Language Technology*.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *TACL*, 1:179–192.

Percy Liang and Christopher Potts. 2015. Corpus-based semantics and pragmatics. *Annual Review of Linguistics*, 1(1).

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*.

Bill MacCartney and Christopher D Manning. 2007. Natural logic for textual inference. In *ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Bill MacCartney and Christopher D Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Coling*.

Bill MacCartney and Christopher D Manning. 2009. An extended model of natural logic. In *Proceedings of the eighth international conference on computational semantics*.

Bill MacCartney. 2009. *Natural Language Inference*. Ph.D. thesis, Stanford.

Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. 2012. Open language learning for information extraction. In *EMNLP*.

David A McAllester and Robert Givan. 1992. Natural language syntax and first-order inference. *Artificial Intelligence*, 56(1):1–20.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. COGEX: A logic prover for question answering. In *NAACL*.

NYSED. 2014. The grade 4 elementary-level science test. `http://www.nysedregents.org/Grade4/Science/home.html`.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL-HLT*.

Víctor Manuel Sánchez Valencia. 1991. *Studies on natural logic and categorial grammar*. Ph.D. thesis, University of Amsterdam.

Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *EMNLP*.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *ACL*.

Johan van Benthem. 1986. *Essays in logical semantics*. Springer.

Fei Wu and Daniel S Weld. 2010. Open information extraction using wikipedia. In *ACL*. Association for Computational Linguistics.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*. AUAI Press.

Easy Questions First?
A Case Study on Curriculum Learning for Question Answering

Mrinmaya Sachan **Eric P. Xing**
School of Computer Science
Carnegie Mellon University
{mrinmays, epxing}@cs.cmu.edu

Abstract

Cognitive science researchers have emphasized the importance of ordering a complex task into a sequence of easy to hard problems. Such an ordering provides an easier path to learning and increases the speed of acquisition of the task compared to conventional learning. Recent works in machine learning have explored a curriculum learning approach called self-paced learning which orders data samples on the easiness scale so that easy samples can be introduced to the learning algorithm first and harder samples can be introduced successively. We introduce a number of heuristics that improve upon self-paced learning. Then, we argue that incorporating easy, yet, a diverse set of samples can further improve learning. We compare these curriculum learning proposals in the context of four non-convex models for QA and show that they lead to real improvements in each of them.

1 Introduction

A key challenge in building an intelligent agent is in modeling the incrementality and the cumulative nature of human learning (Skinner, 1958; Peterson, 2004; Krueger and Dayan, 2009). Children typically learn grade by grade, progressing from simple concepts to more complex ones. Given a complex set of concepts, it is often the case that some concepts are easier than others. Some concepts can even be prerequisite to learning other concepts. Hence, evolving a useful curriculum where easy concepts are presented first and more complex concepts are gradually introduced can be beneficial for learning.

We explore methods for learning a curriculum in the context of non-convex models for question answering. Curriculum learning (CL) (Bengio et al., 2009) and self-paced learning (SPL) (Kumar et al., 2010) have been recently introduced in machine learning literature. However, their usefulness in the context of NLP tasks such as QA has not been studied so far. The main challenge in learning a curriculum is that it requires the identification of easy and hard concepts in the given training dataset. However, in real-world applications, such a ranking of training samples is difficult to obtain. Furthermore, a human judgement of 'easiness' of a task might not correlate with what is easy for the algorithm in the feature and hypothesis space employed for the given application. SPL combines the selection of the curriculum and the learning task in a single objective. The easiness of a question in self-paced learning is defined by its local loss. We propose and study other heuristics that define a measure of easiness and learn the curriculum by selecting samples using this measure. These heuristics are similar to those used in active learning, but with one key difference. In curriculum learning, all the training examples and labels are already known, which is not the case in active learning. Our experiments show that these heuristics work well in practice.

While the strategy of learning from easy questions first and then gradually handling harder questions is supported by many cognitive scientists, others (Cantor, 1946) argue that it is also important to expose the learning to diverse (even if sometimes harder) examples. We argue that the right curriculum should not only be arranged in the increasing order of difficulty but also introduce the learner to sufficient number of diverse examples that are sufficiently dissimilar from what has already been introduced to the learning process. We showed that the above heuristics when coupled with diversity lead to significant improvements.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 453–463,
Berlin, Germany, August 7-12, 2016.

We provide empirical evaluation on four QA models: (a) an alignment-based approach (Sachan et al., 2015) for machine comprehension – a reading comprehension task (Richardson et al., 2013) with a set of questions and associated texts, (b) an alignment-based approach (Sachan et al., 2016) for a multiple-choice elementary science test (Clark and Etzioni, 2016), (c) *QANTA* (Iyyer et al., 2014) – a recursive neural network for answering quiz bowl questions, and (d) memory networks (Weston et al., 2014) – a recurrent neural network with a long-term memory component for answering 20 pre-defined tasks for machine comprehension. We show value in our approaches for curriculum learning on all these settings.
Our paper has the following contributions:

1. In our knowledge, this is the first application of curriculum learning to the task of QA and one of the first in NLP. We hope to make the NLP and ML communities aware of the benefits of CL for non-convex optimization.
2. We perform an in-depth analysis of SPL, and propose heuristics which offer significant improvements over SPL; the state-of-the-art in curriculum learning.
3. We stress on diversity of questions in the curriculum during learning and propose a method that learns a curriculum while capturing diversity to gain more improvements.

2 Problem Setting for QA

For each question $q_i \in Q$, let $A_i = \{a_{i1}, \ldots, a_{im}\}$ be the set of candidate answers to the question. Let a_i^* be the correct answer. The candidate answers may be pre-defined, as in multiple-choice QA, or may be undefined but easy to extract with a high degree of confidence (e.g., by using a pre-existing system). We want to learn a function $f : (q, \mathcal{K}) \rightarrow a$ that, given a question q_i and background knowledge $\mathcal{K}$ (texts/resources required to answer the question), outputs an answer $\hat{a}_i \in A_i$. We consider a scoring function $S_{\mathbf{w}}(q, a; \mathcal{K})$ (with model parameters $\mathbf{w}$) and a prediction rule $f_{\mathbf{w}}(q_i) = \hat{a}_i = \underset{a_{ij} \in A_i}{\arg\max}\, S_{\mathbf{w}}(q_i, a_{ij}; \mathcal{K})$. Let $\Delta(\hat{a}_i, a_i^*)$ be the cost of giving a wrong answer. We consider the empirical risk minimization (ERM) framework given a loss function L and a regularizer Ω:

$$\min_{\mathbf{w}} \sum_{q_i \in Q} L_{\mathbf{w}}(a_i^*, f_{\mathbf{w}}(q_i); \mathcal{K}) + \Omega(\mathbf{w}) \qquad (1)$$

3 QA Models

The field of QA is quite rich. Solutions proposed have ranged from various IR based approaches that treat this as a problem of retrieval from existing knowledge bases or perform inference using a large corpus of unstructured texts by learning a similarity between the question and a set of candidate answers (Yih et al., 2013). A comprehensive review of QA is out of scope of this paper. So we point the interested readers to Jurafsky and Martin (2000), chapter 28 for a more comprehensive review. In this paper, we will explore curriculum learning in the context of non-convex models for QA. The models will be (1) latent structural SVM (Yu and Joachims, 2009) based solutions for standardized question-answering tests and (2) deep learning models (Iyyer et al., 2014; Weston et al., 2014) for QA.

Recently, researchers have proposed standardized tests as 'drivers for progress in AI' (Clark and Etzioni, 2016). Some example standardized tests are reading comprehensions (Richardson et al., 2013), algebra word problems (Kushman et al., 2014), geometry problems (Seo et al., 2014), entrance exams (Fujita et al., 2014; Arai and Matsuzaki, 2014), etc. These tests are usually in the form of question-answers and focus on elementary learning. The idea of learning the curriculum could be especially useful in the context of standardized tests. Standardized tests (Clark and Etzioni, 2016) are implicitly incremental in nature, covering various levels of difficulty. Thus they are rich sources of data for building systems that learn incrementally. These datasets can also help us understand the *shaping* hypothesis as we can use them to verify if easier questions are indeed getting picked by our incremental learning algorithm before harder questions.

On the other hand, deep learning models (LeCun et al., 2015) have recently shown good performance in many standard NLP and vision tasks, including QA. These models usually learn representations of data and the QA model jointly. The models use a cascade of many layers of nonlinear processing units, leading to a highly non-convex model and a large parameter space. This renders these models susceptible to local-minima. Hence, the idea of learning the curricula is also very useful in the context of deep-learning models, as the technique of processing questions in the increasing order of difficulty often leads to better minima

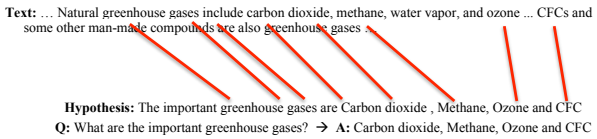


Figure 1: Alignment structure for an example question from the science QA dataset. The question and answer candidate are combined to generate a hypothesis sentence. Then alignments (shown by red lines) are found between the hypothesis and the appropriate snippet in the texts.

(as shown in our results).

3.1 Alignment Based Models

Alignment based models for QA (Yih et al., 2013; Sachan et al., 2015; Sachan et al., 2016) cast QA as a textual entailment problem by converting each question-answer candidate pair (q_i, a_{ij}) into a hypothesis statement h_{ij}. For example, the question "What are the important greenhouse gases?" and answer candidate "Carbon dioxide, Methane, Ozone and CFC" in Figure 1 can be combined to achieve a hypothesis "The important greenhouse gases are Carbon dioxide , Methane, Ozone and CFC.". A set of question matching/rewriting rules are used to achieve this transformation. These rules match the question into one of a large set of pre-defined templates and apply a unique transformation to the question and answer candidate to achieve the hypothesis statement. For each question q_i, the QA task thereby reduces to picking the hypothesis $\hat{h}_i$ that has the highest likelihood among the set of hypotheses $\mathbf{h}_i = \{h_{i1}, \ldots, h_{im}\}$ generated for that question of being entailed by a body of relevant texts. The body of relevant texts can vary for each instance of the QA task. For example, it could be just the passage in a reading comprehension task, or a set of science textbooks in the science QA task. Let $h_i^* \in \mathbf{h}_i$ be the correct hypothesis. The model considers the quality of word alignment from a hypothesis h_{ij} (formed by combining question-answer candidates (q_i, a_{ij})) to snippets in the textbooks as a proxy for the evidence. The alignment depends on: (a) snippet from the relevant texts chosen to be aligned to the hypothesis and (b) word alignment from the hypothesis to the snippet. The snippet from the texts to be aligned to the hypothesis is determined by picking a subset of sentences in the texts. Then each hypothesis word is aligned to a unique word in the snippet. See Figure 1 for an illustration. The choice of snippets composed with the word alignment is latent. Let $\mathbf{z}_{ij}$ represent the latent structure for the question-answer candidate pair $(q_i, a_{i,j})$. A natural solution is to treat QA as a problem of ranking the hypothesis set $\mathbf{h}_i$ such that the correct hypothesis is at the top of this ranking. Hence, a scoring function $S_{\mathbf{w}}(h, \mathbf{z})$ is learnt such that the score given to the correct hypothesis h_i^* and the corresponding latent structure $\mathbf{z}_i^*$ is higher than the score given to any other hypothesis and its corresponding latent structure. In fact, in a max-margin fashion, the model learns the scoring function such that $S_{\mathbf{w}}(h_i^*, \mathbf{z}_i^*) > S_{\mathbf{w}}(h_{ij}, \mathbf{z}_{ij}) + \Delta(h_i^*, h_{ij}) - \xi_i$ for all $h_j \in \mathbf{h} \setminus h^*$ for some slack ξ_i. This can be formulated as the following optimization problem:

$$\min_{||\mathbf{w}||} \quad \frac{1}{2}||\mathbf{w}||_2^2 + C\sum_i \xi_i$$
$$s.t. \quad S_{\mathbf{w}}(h_i^*, \mathbf{z}_i^*) \geq \max_{\mathbf{z}_{ij}} S_{\mathbf{w}}(h_{ij}, \mathbf{z}_{ij}) + \Delta(h_i^*, h_{ij}) - \xi_i$$

It is intuitive to use 0-1 cost, i.e. $\Delta(h_i^*, h_{ij}) = \mathbb{1}(h_i^* \neq h_{ij})$ If the scoring function is convex then this objective is in concave-convex form and can be minimized by the concave-convex programming procedure (CCCP) (Yuille and Rangarajan, 2003). The scoring function is assumed to be linear: $S_{\mathbf{w}}(h, \mathbf{z}) = \mathbf{w}^T\psi(h, \mathbf{z})$. Here, $\psi(h, \mathbf{z})$ is a task-dependent feature map (see Sachan et al. (2015) and Sachan et al. (2016) for details).

3.2 Deep Learning Models

We briefly review two neural network models for QA – Iyyer et al. (2014) and Weston et al. (2014).

QANTA: QANTA (Iyyer et al., 2014) answers quiz bowl questions using a dependency tree structured recursive neural network. It combines predictions across sentences to produce a question answering neural network with trans-sentential averaging. The model is optimized using AdaGrad (Duchi et al., 2011). In quiz bowl, questions typically consist of four to six sentences and are associated with factoid answers. Every sentence in the question is guaranteed to contain clues that uniquely identify its answer, even without the context of previous sentences[1]. Recently, QANTA had beaten the well-known Jeopardy! star Ken Jennings at an exhibition quiz bowl contest.

Memory Networks: Memory networks (Weston et al., 2014) are essentially recurrent neural networks with a long-term memory component. The memory can be read and written to, and can be used for prediction. The memory can be seen as

[1]Refer to Figure 1 in (Iyyer et al., 2014) for an example

acting like a dynamic knowledge base. The model is trained using a margin ranking loss and stochastic gradient descent. It was evaluated on a set of synthetic QA tasks. For each task, a set of statements were generated by a simulation of 4 characters, 3 objects and 5 rooms using an automated grammar with characters moving around, picking up and dropping objects are given, followed by a question whose answer is typically a single word[2].

4 Curriculum Learning

Studies in cognitive science (Skinner, 1958; Peterson, 2004; Krueger and Dayan, 2009) have shown that humans learn much better when the training examples are not randomly presented but organized in increasing order of difficulty. The idea of *shaping*, which consists of training a machine learning algorithm with a curriculum was first introduced by (Elman, 1993) in the context of grammatical structure learning using a recurrent connectionist network. This idea also lent support for the much debated Newport's "less is more" hypothesis (Goldowsky and Newport, 1993; Newport, 1990) that child language acquisition is aided, rather than hindered, by limited cognitive resources. Curriculum learning (Bengio et al., 2009) is a recent idea in machine learning, where a curriculum is designed by ranking samples based on manually curated difficulty measures. These measurements are usually not known in real-world scenarios, and are hard to elicit from humans.

4.1 Self-paced Learning

Self-paced learning (SPL) (Kumar et al., 2010; Jiang et al., 2014a; Jiang et al., 2015) reformulates curriculum learning as an optimization problem by jointly modeling the curriculum and the task at hand. Let $\mathbf{v} \in [0,1]^{|Q|}$ be the weight vector that models the weight of the sample questions in the curriculum. The SPL model includes a weighted loss term on all samples and an additional self-paced regularizer imposed on sample weights $\mathbf{v}$. SPL formulation for the ERM framework described in eq 1 can be rewritten as:

$$\min_{\mathbf{w},\mathbf{v}\in[0,1]^{|Q|}} \sum_{q_i \in Q} v_i L_{\mathbf{w}}(a_i^*, f_{\mathbf{w}}(q_i); \mathcal{K}) + g(v_i, \lambda) + \Omega(\mathbf{w})$$

[2]Refer to Table 1 in (Weston et al., 2015) for examples

The problem usually has closed-form solution with respect to $\mathbf{v}$ (described later; lets call the solution $\mathbf{v}^*(\lambda; L)$ for now). $g(\mathbf{v}, \lambda)$ is usually called the self-paced regularizer with the "age" or "pace" parameter λ. g is convex with respect to $\mathbf{v} \in [0,1]^{|Q|}$. Furthermore, $\mathbf{v}(\lambda; L)$ is monotonically decreasing with respect to L, and $\lim_{L\to 0} \mathbf{v}^*(\lambda; L) = 1$ and $\lim_{L\to\infty} \mathbf{v}^*(\lambda; L) = 0$. This means that the model inclines to select easy samples (with smaller losses) in favor of complex samples (with larger losses). Finally, $v^*(\lambda; L)$ is monotonically increasing with respect to λ, and $\lim_{\lambda\to 0} \mathbf{v}^*(\lambda; L) = 0$ and $\lim_{\lambda\to\infty} \mathbf{v}^*(\lambda; L) \leq 1$. This means that when the model "ages" (i.e. the age parameter λ gets larger), it tends to incorporate more, probably complex samples to train a 'mature' model.

Four popular self-paced regularizers in the literature (Kumar et al., 2010; Jiang et al., 2014a; Zhao et al., 2015) are *hard*, *soft logarithmic*, *soft linear* and *mixture*. These SP-regularizers, summarized with corresponding closed form solutions for $\mathbf{v}$ are shown in Table 1. Hard weighting is usually less appropriate as it cannot discriminate the importance of samples. However, soft weighting assigns real-valued weights and reflects the latent importance of samples in training. The soft linear regularizer linearly weighs samples with respect to their losses and the soft logarithmic penalizes the weight logarithmically. Mixture weighting combines both hard and soft weighting schemes. We can solve the model in the SPL regime by iteratively updating $\mathbf{v}$ (closed form solution for $\mathbf{v}$ is shown in Table 1) and $\mathbf{w}$ (by CCCP, AdaGrad or SGD), and gradually increasing the age parameter λ to let harder and harder problems in.

Since its inception, variations of SPL such as self-paced re-ranking (Jiang et al., 2014a), self-paced learning with diversity (Jiang et al., 2014b), self-paced multiple-instance learning (Zhang et al., 2015) and self-paced curriculum learning (Jiang et al., 2015) have been proposed. The techniques have been shown to be useful in some computer vision tasks (Lee and Grauman, 2011; Kumar et al., 2011; Tang et al., 2012; Supancic and Ramanan, 2013; Jiang et al., 2014a). SPL is different from active learning (Settles, 1995) in the sense that the training examples (and labels) are already provided and the solution only orders the examples to achieve a better solution. On the other hand, active learning tries to interactively query

Regularizer	$g(\mathbf{v};\lambda)$	$v^*(\lambda;L)$
Hard	$-\lambda\mathbf{v}$	$\begin{cases} 1, \textit{if } L \leq \lambda \\ 0, \textit{o/w} \end{cases}$
Soft Linear	$\lambda(\frac{1}{2}\mathbf{v}^2 - \mathbf{v})$	$\begin{cases} -\frac{L}{\lambda} + 1, \textit{if } L \leq \lambda \\ 0, \textit{otherwise} \end{cases}$
Soft Logarithmic	$\sum_{q_i \in Q} \left((1-\lambda)v_i - \frac{(1-\lambda)^{v_i}}{log(1-\lambda)} \right)$	$\begin{cases} \frac{log(L+1-\lambda)}{log(1-\lambda)}, \textit{if } L \leq \lambda \\ 0, \textit{o/w} \end{cases}$
Mixed	$\frac{\gamma^2}{\mathbf{v}+\frac{\gamma}{\lambda}}$	$\begin{cases} 1, \textit{if } L \leq \left(\frac{\lambda\gamma}{\lambda+\gamma}\right)^2 \\ 0, \textit{if } L \geq \lambda^2 \\ \gamma\left(\frac{1}{\sqrt{L}} - \frac{1}{\lambda}\right), \textit{o/w} \end{cases}$

Table 1: Various SP-regularizers for SPL.

the user (or another information source) to achieve a better model with few queries. Curriculum learning is also related to teaching dimension (Khan et al., 2011) which studies the strategies that humans follow as they teach a target concept to a robot by assuming a teaching goal of minimizing the learner's expected generalization error at each iteration. One can also think of curriculum learning as an approach for achieving a better local optimum in non-convex problems.

5 Improved Curriculum Learning Heuristics

SPL selects questions based on the local loss term of the question. This is not the only way to define 'easiness' of the question. Hence, we suggest some other heuristics for selecting the order of questions to be presented to our learning algorithm. The heuristics select the next question $q_i \in Q \setminus Q_0$ given the current model ($\mathcal{M}$) and the set of questions already presented for learning (Q_0). We assume access to a minimization oracle (CCCP/AdaGrad/SGD) for the QA models. We explore the following heuristics:

1) Greedy Optimal (GO): The simplest and greedy optimal heuristic (Schohn and Cohn, 2000) would be to pick a question $q_i \in Q \setminus Q_0$ which has the minimum expected effect on the model. The expected effect on adding q_i can be written as:
$\sum_{a_{ij} \in A_i} p(a_i^* = a_{ij}) \sum_{q_j \in Q_0 \cup q_i} \mathbb{E}\left[L_{\mathbf{w}}(a_j^*, f_{\mathbf{w}}(q_j); \mathcal{K})\right]$.
$p(a_i^* = a_{ij})$ can be estimated by normalizing $S_{\mathbf{w}}(q, a; \mathcal{K})$. $\sum_{q_j \in Q_0 \cup q_i} \mathbb{E}\left[L_{\mathbf{w}}(a_j^*, f_{\mathbf{w}}(q_j); \mathcal{K})\right]$ can be estimated by retraining the model on $Q_0 \cup q_i$.

2) Change in Objective (CiO): Choose the question $q_i \in Q \setminus Q_0$ that causes the smallest increase in the objective. If there are multiple questions with the smallest increase in objective, pick one of them randomly.

3) Mini-max ($\mathbf{M^2}$): Chooses question $q_i \in Q \setminus Q_0$ that minimizes the regularized expected risk when including the question with the answer candidate a_{ij} that yields the maximum error.

$$\hat{q}_i = \arg\min_{q_i \in Q \setminus Q_0} \max_{a_{ij} \in A_i} L_{\mathbf{w}}(a_{ij}, f_{\mathbf{w}}(q_i); \mathcal{K})$$

4) Expected Change in Objective (ECiO): In this greedy heuristic, we pick a question $q_i \in Q \setminus Q_0$ which has the minimum expected effect on the model. The expected effect can be written as $\sum_{a_{ij} \in A_i} p(a_i^* = a_{ij}) \times \mathbb{E}\left[L_{\mathbf{w}}(a_i^*, f_{\mathbf{w}}(q_i); \mathcal{K})\right]$. Here, $p(a_i^* = a_{ij})$ can be achieved by normalizing $S_{\mathbf{w}}(q, a; \mathcal{K})$ and $\mathbb{E}\left[L_{\mathbf{w}}(a_i^*, f_{\mathbf{w}}(q_i); \mathcal{K})\right]$ can be estimated by running inference for q_i.

4) Change in Objective-Expected Change in Objective (CiO - ECiO): We pick a question $q_i \in Q \setminus Q_0$ which has the minimum value of the difference between the change in objective and the expected change in objective. Intuitively, the difference represents how much the model is surprised to see this new question.

5) Correctly Answered (CA): Pick a question $q_i \in Q \setminus Q_0$ which is answered by the model $\mathcal{M}$ with the minimum cost $\Delta(\hat{a}_i, a_i^*)$. If there are multiple questions with minimum cost, pick one of them randomly.

6) Farthest from Decision Boundary (FfDB): This heuristic applies for latent structural SVMs only. Here, we choose the question $q_i \in Q \setminus Q_0$ whose predicted answer $\hat{a}_i$ is farthest from the decision boundary: $\max_{\mathbf{z}^*} w^T \phi(q_i, a^*, \mathbf{z}^*, \mathcal{K}) = \max_{\hat{\mathbf{z}}} w^T \phi(q, \hat{a}, \hat{\mathbf{z}}, \mathcal{K}) + \Delta(\hat{a}, a^*)$.

5.1 Timing Considerations:

A key consideration in applying the above heuristics is efficiency as the QA models considered (latent structural SVM and deep learning) are compu-

tationally expensive. Among our selection strategies, *GO* and *CiO* require updating the model, M^2, *ECiO*, *CA* and *FfDB* require performing inference on the candiate questions, while *CiO* - *ECiO* requires both retraining as well as inference. Consequently, M^2, *ECiO*, *CA* and *FfDB* are most efficient. We can also gain considerable speed-up by picking questions in batches. This results in significant speed-up with small loss in accuracy. We will discuss the batch question selection setup in more detail in our experiments.

5.2 Smarter Selection Strategies:

We further describe some improvements to the above selection strategies:

1) Ensemble Strategy: In this strategy, we combine all of the above heuristics into an ensemble. The ensemble computes the ratio of the score of the suggested question pick and the average score over remaining $Q \setminus Q_0$ questions for all the heuristics and picks the question with the highest ratio. As we will see in our results, this ensemble works well in practice.

2) Importance-Weighting (IW): Importance weighting is a common technique in active learning literature (Tong and Koller, 2002; Beygelzimer et al., 2009; Beygelzimer et al., 2010), which mitigates the problem that if we query questions actively instead of selecting them uniformly at random, the training (and test) question sets are no longer independent and identically distributed (i.i.d.). In other words, the training set will have a sample selection bias that can impair prediction performance. To mitigate this, we propose to sample questions from a biased sample distribution $\widetilde{D}$. To achieve $\widetilde{D}$, we introduce the weighted loss $\widetilde{L}_{\mathbf{w}}(a, f_{\mathbf{w}}(q); \mathcal{K}) = \widetilde{w}(q,a) \times L_{\mathbf{w}}(a, f_{\mathbf{w}}(q); \mathcal{K})$ where $\widetilde{w}(q,a)$ is the weighting function $\widetilde{w}(q,a) = \frac{p_D(q,a)}{p_{\widetilde{D}}(q,a)}$ which represents how likely it is to observe (q,a) under D compared to $\widetilde{D}$. In this setting, we can show that the generalization error under $\widetilde{D}$ is the same as that under D:

$$
\begin{aligned}
&\mathbb{E}_{(q,a)\sim\widetilde{D}}\left[\widetilde{L}_{\mathbf{w}}(a, f_{\mathbf{w}}(q); \mathcal{K})\right] \\
&= \int_{(q,a)} p_{\widetilde{D}}(q,a) \frac{p_D(q,a)}{p_{\widetilde{D}}(q,a)} L_{\mathbf{w}}(a, f_{\mathbf{w}}(q); \mathcal{K}) d(q,a) \\
&= \int_{(q,a)} p_D(q,a) L_{\mathbf{w}}(a, f_{\mathbf{w}}(q); \mathcal{K}) d(q,a) \\
&= \mathbb{E}_{(q,a)\sim D}\left[L_{\mathbf{w}}(a, f_{\mathbf{w}}(q); \mathcal{K})\right]
\end{aligned}
$$

Thus, given appropriate weights $\widetilde{w}(q,a)$, we modify our loss-function in order to compute an unbiased estimator of the generalization error. Each question-answer is assigned with a non-negative weight. For latent structural SVMs, one can minimize the weighted loss by simply multiplying the corresponding regularization parameter C_i with a corresponding term. In neural networks, this is simply achieved by multiplying the gradients with the corresponding weights. The weights can be set by an appropriate heuristic, e.g. proportional to distance from the decision boundary.

5.3 Incorporating Diversity with Explore and Exploit (E&E):

The strategy of learning from easy questions first and then gradually handling harder questions is intuitive as it helps the learning process. Yet, it has one key deficiency. Under curriculum learning, by focusing on easy questions first, our learning algorithm is usually not exposed to a diverse set of questions. This is particularly a problem for deep-learning approaches that learn representations during the process of learning. Hence, when a harder question arrives, it is usually hard for the learner to adjust to this new question as the current representation may not be appropriate for the new level of difficulty. This motivates our E&E strategy.

The explore and exploit strategy ensures that while we still select easy questions first, we also want to make our selection as diverse as possible. We define a measure for diversity as the angle between the hyperplanes that the question samples induce in feature space: $\angle(\phi(q_i, a_i^*, \mathbf{z}_i^*, \mathcal{K}), \phi(q_{i'}, a_{i'}^*, \mathbf{z}_{i'}^*, \mathcal{K})) = \textit{Cosine}^{-1}\left(\frac{|\phi(q_i,a_i^*,\mathbf{z}_i^*,\mathcal{K})\phi(q_{i'},a_{i'}^*,\mathbf{z}_{i'}^*,\mathcal{K})|}{||\phi(q_i,a_i^*,\mathbf{z}_i^*,\mathcal{K})||||\phi(q_{i'},a_{i'}^*,\mathbf{z}_{i'}^*,\mathcal{K})||}\right)$. The *E&E* solution picks the question which optimizes a convex combination of the curriculum learning objective and the sum of angles between the candidate question pick and questions in Q_0. The convex combination is tuned on the development set.

6 Experiments

6.1 Datasets

As described, we study curriculum learning on four different tasks. The first task is question answering for reading comprehensions. We use MCTest-500 dataset (Richardson et al., 2013), a freely available set of 500 stories (300 train, 50 dev and 150 test) and associated questions to evaluate our model. Each story in MCTest has four

		Machine Comprehension	Science QA	QANTA	Memory Networks
	No Curriculum (NC)	66.62±0.22	42.77±0.04	70.40±0.07	75.03±0.06
SPL	Hard	67.36±0.16	43.85±0.18	70.66±0.19	71.01±0.09
SPL	Soft Linear	68.04±0.17	43.80±0.22	71.65±0.18	72.33±0.07
SPL	Soft Log	68.89±0.16	44.19±0.20	71.92±0.16	73.32±0.09
SPL	Mixed	69.47±0.18	44.86±0.20	72.89±0.19	74.28±0.13
Heuristics	CA	66.86±0.06	42.93±0.08	70.78±0.08	70.96±0.04
Heuristics	M^2	66.98±0.12	43.19±0.17	71.02±0.18	69.73±0.06
Heuristics	ECiO	67.39±0.14	44.00±0.22	71.66±0.19	71.01±0.07
Heuristics	GO	67.65±0.12	44.35±0.15	71.94±0.17	71.28±0.06
Heuristics	CiO	68.20±0.10	44.56±0.12	72.61±0.14	71.98±0.06
Heuristics	FfDB	68.32±0.11	44.78±0.13	-	-
Heuristics	CiO-ECiO	68.65±0.13	44.97±0.11	73.34±0.10	73.22±0.05
Heur++	Ensemble	69.26±0.08	45.48±0.07	74.11±0.07	74.24±0.04
Heur++	+IW	69.86±0.10	45.86±0.12	75.02±0.15	74.55±0.05
Heur++	+E&E	69.93±0.13	46.57±0.17	76.24±0.15	77.64±0.11
Heur++	+IW+E&E	**70.16**±0.14	**46.68**±0.19	**76.89**±0.18	**77.85**±0.09
SPL+E&E	Hard	68.03±0.17	44.50±0.20	72.34±0.19	74.43±0.06
SPL+E&E	Soft Linear	68.51±0.19	44.43±0.21	73.16±0.19	75.74±0.07
SPL+E&E	Soft Log	69.27±0.18	44.92±0.20	73.47±0.18	76.63±0.10
SPL+E&E	Mixed	69.89±0.21	45.58±0.21	74.39±0.21	77.12±0.15

Table 2: Accuracy on the test set obtained on the four experiments, comparing results when no curriculum (NC) was learnt, when we use self-paced learning (SPL) with four variations of SP-regularizers, the six heuristics and four improvements proposed by us. Each cell reports the mean±se (standard error) accuracy over 10 repetitions of each experimental configuration.

multiple-choice questions, each with four answer choices. Each question has exactly one correct answer. The second task is science question answering. We use a mix of 855 third, fourth and fifth grade science questions derived from a variety of regional and state science exams[3] for training and evaluating our model. We used publicly available science textbooks available through $ck12.org$ and Simple English Wikipedia[4] as texts required to answer the questions. The model retrieves a section from the textbook or a Wikipedia page (using a lucene index on the sections and Wikipedia pages) by querying for the hypothesis h_{ij} and then aligning the hypothesis to snippets in the document. For QANTA (Iyyer et al., 2014), we use questions from quiz bowl tournaments for training as in Iyyer et al. (2014). The dataset contains 20,407 questions with 2347 answers. For each answer in the dataset, its corresponding Wikipedia page is also provided. Finally, for memory networks (Weston et al., 2014), we use the synthetic QA tasks defined in Weston et al. (2015) (version 1.1 of the dataset). There are 20 different types of tasks that probe different forms of reasoning and deduction. Each task consists of a set of statements, followed by a question whose answer is typically a single word or a set of words. We report mean accuracy across these 20 tasks.

6.2 Results

We implemented and compared the six selection heuristics (§5) with the suggested improvements (§5.2) and self-paced learning (§4) with the explore and exploit extension for both alignment based models (§3.1) and two deep learning models (§3.2). We use accuracy (proportion of test questions correctly answered) as our evaluation metric. In all our experiments, we begin with zero training data (random initialization). For alignment based models, we select 1 percent of training set questions after every epoch (an epoch is defined as a single pass through the current training set by the optimization oracle) and add them to the training set based on the selection strategy. For deep learning models, we discovered that the learning was a lot slower so we added 0.1 percent of new training set questions after every epoch. Hyper parameters of the alignment based models and the deep learning models were fixed to the corresponding values proposed in their corresponding papers (pre-tuned for the optimization oracle on a held-out development set). All the results reported in this paper are averaged over 10 runs of each experiment.

Table 5.3 reports test accuracies obtained on all the QA tasks, comparing the aforementioned proposals against corresponding models when curriculum learning is not used. We can observe from

[3] http://aristo-public-data.s3.amazonaws.com/AI2-Elementary-NDMC-Feb2016.zip

[4] https://dumps.wikimedia.org/simplewiki/20151102/

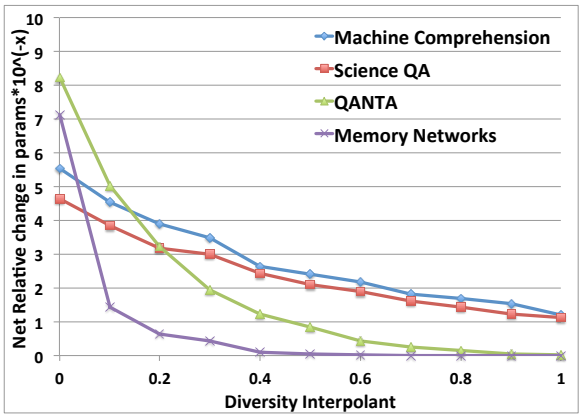


Figure 2: Relative change in parameters*10^{-x} where $x = 2$ for machine comprehension and science QA, 4 for QANTA and memory networks when CL is used.

these results that variants of SPL (and *E&E*) as well as the heuristics (and improvements) lead to improvements in the final test accuracy for both alignment-based models and QANTA.

The surprising ineffectiveness of the heuristics and SPL for memory networks essentially boils down to the abrupt restructure of memory the model has to do for curriculum learning. We provide support for this argument in Figure 2 which plots the net relative change in all the parameters $\mathbf{W}$ until convergence $\left(\frac{1}{\textit{No. of parameters}} \sum_{e:epoch=1}^{\infty} \frac{||\mathbf{W}_{e+1}-\mathbf{W}_e||_1}{||\mathbf{W}_e||_1}\right)$ for each of the four tasks on the model *Ensemble+E&E* against the linear interpolant used to tune the explore and exploit combination. As the interpolant grows from 0 to 1, more and more diverse questions get selected. We can observe that the change in parameters decreases as more diverse questions are selected for all the four tasks. Furthermore, once we bring in diversity (change the interpolant from 0 to 0.1), the relative change in parameters drops sharply for both neural network approaches. The drop is sharpest for memory networks. Easier examples usually require less memory than hard examples. Memory networks have no incentive to utilize only a fraction of its state for easy examples. They simply use the entire memory capacity. This implies that harder examples appearing later require a restructuring of all memory patterns. The network needs to change its memory representation every time in order to free space and accommodate the harder example. This process of memory pattern restructuring is difficult to achieve, so it could be the reason for the relatively poor performance of naive curriculum learning and SPL strategies. However, as we can see from the previous results, the explore and exploit strategy of mixing in some harder examples avoids the problem of having to abruptly restructure memory patterns. The extra samples of all difficulties prevent the network from utilizing all the memory on the easy examples, thus eliminating the need to restructure memory patterns.

From Table 5.3 , we can observe that the choice of the SP-regularizer is important. The soft regularizers perform better than the hard regularizer. The mixed regularizer (with mixture weighting) performs even better. We can also observe that all the heuristics work as well as SPL, despite being a lot simpler. The heuristics arranged in increasing order of performance are: *CA*, M^2, *ECiO*, *GO*, *CiO*, *FfDB* and *CiO-ECiO*,. The differences between the heuristics are larger for alignment-based models and smaller for deep learning models. The *ECiO* heuristic has very similar performance to SPL with hard SP-regularizer. This is understandable as SPL also selects 'easy' questions based on their expected objective value. The *Ensemble* is a significant improvement over the individual heuristics. Importance weighting (*IW*) and the explore and exploit strategies (*E&E*) provide further improvements. *E&E* is crucial to making curriculum learning work for deep learning approaches as described before. Motivated by the success of *E&E*, we also extended it to SPL[5] by tuning a convex combination as before. *E&E* provides improvements across all the experiments for all the SPL experiments. While, the strategy is more important for memory networks, it leads to improvements on all the tasks.

In order to understand the curriculum learning process and to test the hypothesis that the procedure indeed selects easy questions first, successively moving on to harder questions, we plot the number of questions of grade 3, 4 and 5 picked by *SPL*, *Ensemble* and *Ensemble+E&E* against the epoch number in Figure 3. We can observe that all the three methods pick more questions from grade 3 initially, successively moving on to more and more grade 4 questions and finally more grade 5 questions. Both *Ensemble* and *Ensemble+E&E* are more aggressive at learning this curriculum than *SPL*. *Ensemble* becomes too aggressive so

[5]This is different from Jiang et al. (2014c) which encourages diversity in samples across groups. On the other hand, we encourage diversity in feature space.

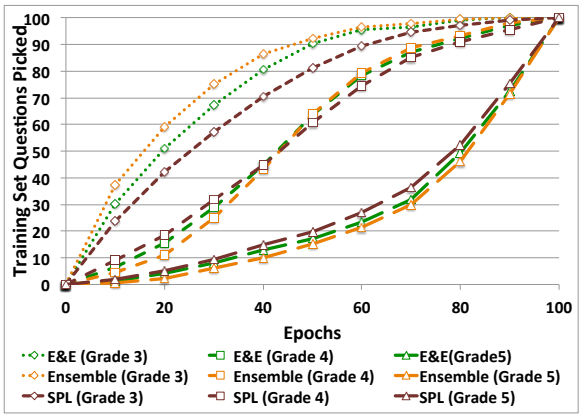


Figure 3: Number of grade 3, 4 and 5 questions picked vs Epoch for various CL approaches for Science QA.

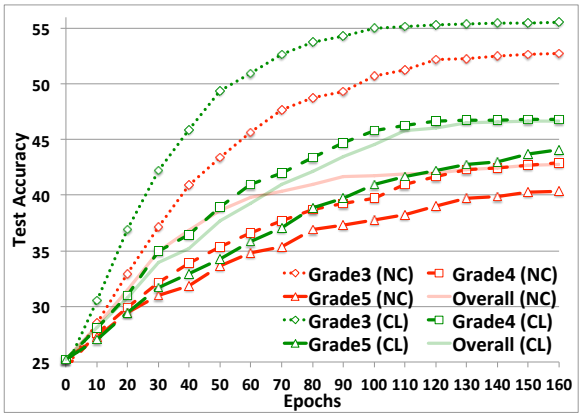


Figure 4: Test split accuracy on grade 3, 4 and 5 questions picked vs Epoch for Science QA when CL is used/not used.

E&E, initially increases the number of grade 4 and grade 5 questions received by the learner, thereby incorporating diversity in learning.

In order to further the claim that curriculum learning follows the principal of learning simpler concepts first and then learning successively harder and harder concepts, we plot the test accuracy on grade 3, 4 and 5 questions with curriculum learning (CL) – i.e. *Ensemble+E&E* and without curriculum learning (NC) against the epoch number in Figure 4. Here, we can see that the test accuracy increases for questions in all three grade levels. With curriculum learning, the accuracy on grade 3 questions rises sharply in the beginning. This rise is sharper than the case when curriculum learning is not used. Grade 3 test accuracy for curriculum learning then saturates (saturates earlier compared to the case when curriculum learning is not used). The improvements due to curriculum learning for grade 4 questions mainly occur in epochs 30-140. The final epochs of curriculum learning see greater gain in test accuracy for grade 5 questions over the case when curriculum learning is not used. All these experiments together support the intuition of curriculum learning. The models indeed pick and learn from easier questions first and successively learn from harder and harder questions. We also tried variants of our models where we used curriculum learning on grade 3 questions, followed by grade 4 and grade 5 questions. However, this did not lead to significant improvements. Perhaps, this is because questions that are easy for humans may not always correspond to what is easy for our algorithms. Characterizing what is easy for algorithms and how it relates to what is easy for humans is an interesting question for future research.

7 Conclusion

Curriculum learning is inspired by the way humans acquire knowledge and skills: by mastering simple concepts first, and progressing through information with increasing difficulty to grasp more complex topics. We studied self-paced learning, an approach for curriculum learning that expresses the difficulty of a data sample in terms of the value of the objective function and builds the curriculum via a joint optimization framework. We proposed a number of heuristics, an ensemble, and several improvements for selecting the curriculum that improves upon self-paced learning. We stressed on another important aspect of human learning – diversity, that requires that the right curriculum should not only arrange the data samples in increasing order of difficulty but should also introduce the learner to a small number of samples that are sufficiently dissimilar to the samples that have already been introduced to the learning process. We showed that our heuristics when coupled with diversity lead to significant improvements in a number of question answering tasks. The approach is quite general and we hope that this paper will encourage more NLP researchers to explore curriculum learning in their own works.

Acknowledgments

We thank the anonymous reviewers, along with Emmanouil A. Platanios and Snigdha Chaturvedi for their valuable comments and suggestions that helped improve the quality of this paper. This work was supported by the following research grants: NSF IIS1218282, NSF IIS1447676 and AFOSR FA95501010247.

References

[Arai and Matsuzaki2014] Noriko H Arai and Takuya Matsuzaki. 2014. The impact of ai on education–can a robot get into the university of tokyo? In *Proc. ICCE*, pages 1034–1042.

[Bengio et al.2009] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM.

[Beygelzimer et al.2009] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. 2009. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 49–56. ACM.

[Beygelzimer et al.2010] Alina Beygelzimer, John Langford, Zhang Tong, and Daniel J Hsu. 2010. Agnostic active learning without constraints. In *Advances in Neural Information Processing Systems*, pages 199–207.

[Cantor1946] Nathaniel Freeman Cantor. 1946. *Dynamics of learning*. Foster and Stewart publishing corporation, Buffalo, NY.

[Clark and Etzioni2016] Peter Clark and Oren Etzioni. 2016. My computer is an honor student - but how intelligent is it? standardized tests as a measure of ai. In *Proceedings of AI Magazine*.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

[Elman1993] Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99.

[Fujita et al.2014] Akira Fujita, Akihiro Kameda, Ai Kawazoe, and Yusuke Miyao. 2014. Overview of todai robot project and evaluation framework of its nlp-based problem solving. *World History*, 36:36.

[Goldowsky and Newport1993] B.N. Goldowsky and E.L. Newport. 1993. Modeling the effects of processing limitations on the acquisition of morphology: The less is more hypothesis. In *Proceedings of the 11th West Coast Conference on Formal Linguistics*.

[Iyyer et al.2014] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Proceedings of Empirical Methods in Natural Language Processing*.

[Jiang et al.2014a] Lu Jiang, Deyu Meng, Teruko Mitamura, and Alexander G Hauptmann. 2014a. Easy samples first: Self-paced reranking for zero-example multimedia search. In *Proceedings of the ACM International Conference on Multimedia*, pages 547–556. ACM.

[Jiang et al.2014b] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014b. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086.

[Jiang et al.2014c] Lu Jiang, Deyu Meng, Shoou-I Yu, Zhenzhong Lan, Shiguang Shan, and Alexander Hauptmann. 2014c. Self-paced learning with diversity. In *Advances in Neural Information Processing Systems*, pages 2078–2086.

[Jiang et al.2015] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. 2015. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[Jurafsky and Martin2000] Daniel Jurafsky and James H Martin. 2000. *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall.

[Khan et al.2011] Faisal Khan, Bilge Mutlu, and Xiaojin Zhu. 2011. How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems*, pages 1449–1457.

[Krueger and Dayan2009] Kai A Krueger and Peter Dayan. 2009. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394.

[Kumar et al.2010] M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197.

[Kumar et al.2011] M Pawan Kumar, Haithem Turki, Dan Preston, and Daphne Koller. 2011. Learning specific-class segmentation from diverse data. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1800–1807. IEEE.

[Kushman et al.2014] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[LeCun et al.2015] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

[Lee and Grauman2011] Yong Jae Lee and Kristen Grauman. 2011. Learning the easy things first: Self-paced visual category discovery. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1721–1728. IEEE.

[Newport1990] Elissa L Newport. 1990. Maturational constraints on language learning. *Cognitive science*, 14(1):11–28.

[Peterson2004] Gail B Peterson. 2004. A day of great illumination: Bf skinner's discovery of shaping. *Journal of the Experimental Analysis of Behavior*, 82(3):317–328.

[Richardson et al.2013] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*.

[Sachan et al.2015] Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[Sachan et al.2016] Mrinmaya Sachan, Avinava Dubey, and Eric P. Xing. 2016. Science question answering using instructional materials. *CoRR*, abs/1602.04375.

[Schohn and Cohn2000] Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proceedings of the 17th Annual International Conference on Machine Learning*. ACM.

[Seo et al.2014] Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni. 2014. Diagram understanding in geometry questions. In *Proceedings of AAAI*.

[Settles1995] Burr Settles. 1995. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

[Skinner1958] Burrhus F Skinner. 1958. Reinforcement today. *American Psychologist*, 13(3):94.

[Supancic and Ramanan2013] James Supancic and Deva Ramanan. 2013. Self-paced learning for long-term tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2379–2386.

[Tang et al.2012] Kevin Tang, Vignesh Ramanathan, Li Fei-Fei, and Daphne Koller. 2012. Shifting weights: Adapting object detectors from image to video. In *Advances in Neural Information Processing Systems*, pages 638–646.

[Tong and Koller2002] Simon Tong and Daphne Koller. 2002. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66.

[Weston et al.2014] Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR*, abs/1410.3916.

[Weston et al.2015] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

[Yih et al.2013] Wentau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

[Yu and Joachims2009] Chun-Nam Yu and T. Joachims. 2009. Learning structural svms with latent variables. In *Proceedings of International Conference on Machine Learning (ICML)*.

[Yuille and Rangarajan2003] A. L. Yuille and Anand Rangarajan. 2003. The concave-convex procedure. *Neural Comput.*

[Zhang et al.2015] Dingwen Zhang, Deyu Meng, Chao Li, Lu Jiang, Qian Zhao, and Junwei Han. 2015. A self-paced multiple-instance learning framework for co-saliency detection. June.

[Zhao et al.2015] Qian Zhao, Deyu Meng, Lu Jiang, Qi Xie, Zongben Xu, and Alexander G Hauptmann. 2015. Self-paced learning for matrix factorization. In *Proceedings of AAAI*.

Improved Representation Learning for Question Answer Matching

Ming Tan, Cicero dos Santos, Bing Xiang & Bowen Zhou
IBM Watson Core Technologies
Yorktown Heights, NY, USA
{mingtan,cicerons,bingxia,zhou}@us.ibm.com

Abstract

Passage-level question answer matching is a challenging task since it requires effective representations that capture the complex semantic relations between questions and answers. In this work, we propose a series of deep learning models to address passage answer selection. To match passage answers to questions accommodating their complex semantic relations, unlike most previous work that utilizes a single deep learning structure, we develop hybrid models that process the text using both convolutional and recurrent neural networks, combining the merits on extracting linguistic information from both structures. Additionally, we also develop a simple but effective attention mechanism for the purpose of constructing better answer representations according to the input question, which is imperative for better modeling long answer sequences. The results on two public benchmark datasets, InsuranceQA and TREC-QA, show that our proposed models outperform a variety of strong baselines.

1 Introduction

Passage-level answer selection is one of the essential components in typical question answering (QA) systems. It can be defined as follows: Given a question and a pool of candidate passages, select the passages that contain the correct answer. The performance of the passage selection task is not only crucial to non-factoid QA systems, where a question is expected to be answered with a sequence of descriptive text (e.g. the question in Table 1), but also very important to factoid QA systems, where the answer passage selection step is also known as passage scoring. In factoid QA, if the sentences selected by the passage scorer module do not contain the answer, it will definitely lead to an incorrect response from the QA system.

Question: Does Medicare cover my spouse?
Ground-truth answer: If your spouse has worked and paid Medicare taxes for the entire required 40 quarters, or is eligible for Medicare by virtue of being disabled or some other reason, your spouse can receive his/her own medicare benefits. If your spouse has not met those qualifications, if you have met them, and if your spouse is age 65, he/she can receive Medicare based on your eligibility.
Another candidate answer: If you were married to a Medicare eligible spouse for at least 10 years, you may qualify for Medicare. If you are widowed, and have not remarried, and you were married to your spouse at least 9 months before your spouse's death, you may be eligible for Medicare benefits under a spouse provision.

Table 1: An example of a question with the ground-truth answer and a negative answer extracted from the InsuranceQA dataset.

One central challenge of this task lies in the complex and versatile semantic relations observed between questions and passage answers. For example, while the task of supporting passage selection for factoid QA may be largely cast as a textual entailment problem, what makes an answer better than another in the real world for non-factoid QA often depends on many factors.

Specifically, different from many other pair-matching NLP tasks, the linguistic similarities between questions and answers may or may not be indicative for our task. This is because, depending on what the question is looking for, a good answer may come in different forms: sometimes a correct

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 464–473,
Berlin, Germany, August 7-12, 2016.

answer completes the question precisely with the missing information, and in other scenarios, good answers need to elaborate part of the question to rationalize it, and so on. For instance, the question in Table 1 only contains five words, while the best answer uses 60 words for elaboration. On the other hand, the best answers from a pool can also be noisy and include extraneous information irrelevant to the question. Additionally, while a good answer must relate to the question, they often do not share common lexical units. For instance, in the example question, "cover" is not directly mentioned in the answer. This issue may confuse simple word-matching systems.

These challenges consequently make handcrafting features much less desirable compared to deep learning based methods. Furthermore, they also require our systems to learn how to distinguish useful pieces from irrelevant ones, and further, to focus more on the former.

Finally, the system should be capable of capturing the nuances between the best answer and an acceptable one. For example, the second answer in Table 1 is suitable for a questioner, whose spouse is Medicare eligible, asking about his/her own coverage, while the example question is more likely asked by a person, who is Medicare eligible, asking about his/her spouse' coverage. Clearly, the first answer is more appropriate for the question, although the second one implicitly answers it. A good system should reflect this preference.

While this task is usually approached as a pairwise-ranking problem, the best strategy to capture the association between the questions and answers is still an open problem. Established approaches normally suffer from two weaknesses at this point. First, prior work, such as (Feng et al., 2015; Wang and Nyberg, 2015), resort to either convolutional neural network (CNN) or recurrent neural network (RNN) respectively. However, each structure describes only one semantic perspective of the text. CNN emphasizes the local interaction within n-gram, while RNN is designed to capture long range information and forget unimportant local information. How to combine the merits from both has not been sufficiently explored. Secondly, previous approaches are usually based on independently generated question and answer embeddings; the quality of such representations, however, usually degrades as the answer sequences grow longer.

In this work, we propose a series of deep learning models in order to address such weaknesses. We start with the basic discriminative framework for answer selection. We first propose two independent models, Convolutional-pooling LSTM and Convolution-based LSTM, which are designed to benefit from both of the two popular deep learning structures to distinguish better between useful and irrelevant pieces presented in questions and answers. Next, by breaking the independence assumption of the question and answer embedding, we introduce an effective attention mechanism to generate answer representations according to the question, such that the embeddings do not overlook informative parts of the answers. We report experimental results for two answer selection datasets: (1) InsuranceQA (Feng et al., 2015) [1], a recently released large-scale non-factoid QA dataset from the insurance domain, and (2) TREC-QA [2], which was created by Wang et al. (2007) based on Text REtrieval Conference (TREC) QA track data.

The contribution of this paper is hence threefold: 1) We propose hybrid neural networks, which learn better representations for both questions and answers by combining merits of both RNN and CNN. 2) We prove the effectiveness of attention on the answer selection task, which has not been sufficiently explored in prior work. 3) We achieve the state-of-the-art results on both TREC-QA and InsuranceQA datasets.

The rest of the paper is organized as follows: Section 2 describes the related work for answer selection; Section 3 provides the details of the proposed models; Experimental settings and results are discussed in Section 4 and 5; Finally, we draw conclusions in Section 6.

2 Related work

Previous work on answer selection normally used feature engineering, linguistic tools, or external resources. For example, semantic features were constructed based on WordNet in (Yih et al., 2013). This model pairs semantically related words based on word semantic relations. In (Wang and Manning, 2010; Wang et al., 2007), the answer selection problem was transformed to a syntacti-

[1] git clone https://github.com/shuzi/insuranceQA.git (We use the V1 version of this dataset).

[2] The data is obtained from (Yao et al., 2013) http://cs.jhu.edu/~xuchen/packages/jacana-qa-naacl2013-data-results.tar.bz2

cal matching between the question/answer parse trees. Some work tried to fulfill the matching using minimal edit sequences between dependency parse trees (Heilman and Smith, 2010; Yao et al., 2013). Discriminative tree-edit feature extraction and engineering over parsing trees were automated in (Severyn and Moschitti, 2013). Such methods might suffer from the availability of additional resources, the effort of feature engineering and the systematic complexity introduced by the linguistic tools, such as parse trees and dependency trees.

Some recent work has used deep learning methods for the passage-level answer selection task. The approaches normally pursue the solution on the following directions. First, a joint feature vector is constructed based on both the question and the answer, and then the task can be converted into a classification or ranking problem (Wang and Nyberg, 2015; Hu et al., 2014). Second, recently proposed models for text generation can intrinsically be used for answer selection and generation (Bahdanau et al., 2015; Vinyals and Le, 2015). Finally, the question and answer representations can be learned and then matched by certain similarity metrics (Feng et al., 2015; Yu et al., 2014; dos Santos et al., 2015; Qiu and Huang, 2015). Fundamentally, our proposed models belong to the last category.

Meanwhile, attention-based systems have shown very promising results on a variety of NLP tasks, such as machine translation (Bahdanau et al., 2015; Sutskever et al., 2014), machine reading comprehension (Hermann et al., 2015), text summarization (Rush et al., 2015) and text entailment (Rocktäschel et al., 2016). Such models learn to focus their attention to specific parts of their input and most of them are based on a one-way attention, in which the attention is basically performed merely over one type of input based on another (e.g. over target languages based on the source languages for machine translation, or over documents according to queries for reading comprehension). Most recently, several two-way attention mechanisms are proposed, where the information from the two input items can influence the computation of each others representations. Rocktäschel et al. (2016) develop a two-way attention mechanism including another one-way attention over the premise conditioned on the hypothesis, in addition to the one over hypothesis conditioned on premise. dos Santos et al. (2016) and Yin et al. (2015) generate interactive attention weights on both inputs by assignment matrices. Yin et al. (2015) use a simple Euclidean distance to compute the interdependence between the two input texts, while dos Santos et al. (2016) resort to attentive parameter matrices.

3 Approaches

In this section, we first present our basic discriminative framework for answer selection based on long short-term memory (LSTM), which we call QA-LSTM. Next, we detail the proposed hybrid and attentive neural networks that are built on top of the QA-LSTM framework.

3.1 LSTM for Answer Selection

Our LSTM implementation is similar to the one in (Graves et al., 2013) with minor modifications. Given an input sequence $\mathbf{X} = \{\mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(n)\}$, where $\mathbf{x}(t)$ is an E-dimension word vector in this paper, the hidden vector $\mathbf{h}(t)$ (with size H) at the time step t is updated as follows.

$$i_t = \sigma(\mathbf{W}_i\mathbf{x}(t) + \mathbf{U}_i\mathbf{h}(t-1) + \mathbf{b}_i) \quad (1)$$
$$f_t = \sigma(\mathbf{W}_f\mathbf{x}(t) + \mathbf{U}_f\mathbf{h}(t-1) + \mathbf{b}_f) \quad (2)$$
$$o_t = \sigma(\mathbf{W}_o\mathbf{x}(t) + \mathbf{U}_o\mathbf{h}(t-1) + \mathbf{b}_o) \quad (3)$$
$$\tilde{C}_t = \tanh(\mathbf{W}_c\mathbf{x}(t) + \mathbf{U}_c\mathbf{h}(t-1) + \mathbf{b}_c) \quad (4)$$
$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \quad (5)$$
$$\mathbf{h}_t = o_t * \tanh(C_t) \quad (6)$$

There are three gates (input i, forget f and output o), and a cell memory vector C_t. σ is the $sigmoid$ function. $\mathbf{W} \in R^{H\times E}$, $\mathbf{U} \in R^{H\times H}$ and $\mathbf{b} \in R^{H\times 1}$ are the network parameters.

Single-direction LSTMs suffer from the weakness of not making use of the contextual information from the future tokens. Bidirectional LSTMs (biLSTMs) use both the previous and future context by processing the sequence in two directions, and generate two sequences of output vectors. The output for each token is the concatenation of the two vectors from both directions, i.e. $h_t = \overrightarrow{h_t} \parallel \overleftarrow{h_t}$.

QA-LSTM: Our basic answer selection framework is shown in Figure 1. Given an input pair (q,a), where q is a question and a is a candidate answer, first we retrieve the word embeddings (WEs) of both q and a. Then, we separately apply a biLSTM over the two sequences of WEs. Next,

we generate a fixed-sized distributed vector representations using one of the following three approaches: (1) the concatenation of the last vectors on both directions of the biLSTM; (2) average pooling over all the output vectors of the biLSTM; (3) max pooling over all the output vectors. Finally, we use cosine similarity $sim(q, a)$ to score the input (q, a) pair. It is important to note that the same biLSTM is applied to both q and a.

Similar to (Feng et al., 2015; Weston et al., 2014; Hu et al., 2014), we define the training objective as a hinge loss.

$$\mathcal{L} = \max\{0, M - sim(q, a_+) + sim(q, a_-)\} \quad (7)$$

where a_+ is a ground truth answer, a_- is an incorrect answer randomly chosen from the entire answer space, and M is a margin. We treat any question with more than one ground truth as multiple training examples. During training, for each question we randomly sample K negative answers, but only use the one with the highest $\mathcal{L}$ to update the model. Finally, dropout operation is performed on the representations before cosine similarity matching.

The same scoring function, loss function and negative sampling procedure is also used in the NN architectures presented in what follows.

3.2 Convolutional LSTMs

The pooling strategies used in QA-LSTM suffer from the incapability of filtering important local information, especially when dealing with long answer sequences.

Also, it is well known that LSTM models successfully keep the useful information from long-range dependency. But the strength has a trade-off effect of ignoring the local n-gram coherence. This can be partially alleviated with bidirectional architectures.

Meanwhile, the convolutional structures have been widely used in the question answering tasks, such as (Yu et al., 2014; Feng et al., 2015; Hu et al., 2014). Classical convolutional layers usually emphasize the local lexical connections of the n-gram. However, the local pieces are associated with each other only at the pooling step. No long-range dependencies are taken into account during the formulation of convolution vectors.

Fundamentally, recurrent and convolutional neural networks have their own pros and cons, due to their different topologies. How to keep both merits motivates our studies of the following two hybrid models.

3.2.1 Convolutional-pooling LSTMs

In Figure 2 we detail the convolutional-pooling LSTM architecture. In this NN architecture, we replace the simple pooling layers (average/max-pooling) by a convolutional layer, which allows to capture richer local information by applying a convolution over sequences of LSTM output vectors. The number of output vectors k (context window size) considered by the convolution is a hyper-parameter of the model.

The convolution structure adopted in this work is as follows: $Z \in \mathcal{R}^{k|h| \times L}$ is a matrix where the m-th column is the concatenation of k hidden vectors generated from biLSTM centralized in the m-th word of the sequence, L is the length of the sequence after wide convolution (Kalchbrenner et al., 2014). The output of the convolution with c filters is,

$$C = \tanh(\mathbf{W}_{cp} Z) \quad (8)$$

where W_{cp} are network parameters, and $C \in \mathcal{R}^{c \times L}$. The j-th element of the representation vectors ($\mathbf{o}_q$ and $\mathbf{o}_a$) is computed as follows,

$$[o_j] = \max_{1<l<L} [C_{j,l}] \quad (9)$$

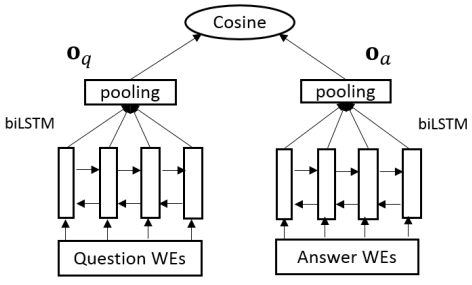


Figure 1: Basic Model: QA-LSTM

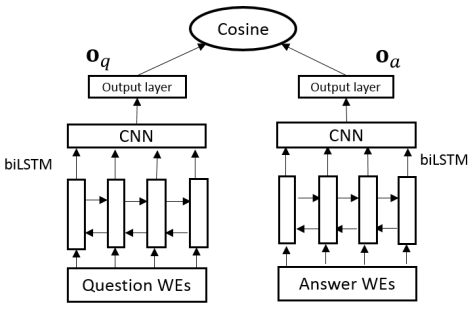


Figure 2: Convolutional-pooling LSTM

3.2.2 Convolution-based LSTMs

In Figure 3, we detail our second hybrid NN architecture. The aim of this approach is to capture the local n-gram interaction at the lower level using a convolution. At the higher level, we build bidirectional LSTMs, which extract the long range dependency based on convoluted n-gram. Combining convolutional and recurrent structures have been investigated in prior work other than question answering (Donahue et al., 2015; Zuo et al., 2015; Sainath et al., 2015).

As shown in Figure 3, the model first retrieves word vectors for each token in the sequence. Next, we compose the matrix $D \in \mathcal{R}^{kE \times L}$, where each column l in D consists of the concatenation of k word vectors of size E centered at the l-th word. The matrix $\mathbf{X} \in \mathcal{R}^{c \times L}$, which is the output of the convolution with c filters is computed as follows:

$$\mathbf{X} = \tanh(\mathbf{W}_{cb} D) \quad (10)$$

The matrix $\mathbf{X}$ is the input to the biLSTM structure in Eqs. 1-6. After the biLSTM step, we use max-pooling over the biLSTM output vectors to obtain the representations of both q and a.

3.3 Attentive LSTMs

In the previous subsections, the two most popular deep learning architectures are integrated to generate semantic representations for questions and answers from both the long-range sequential and local n-gram perspectives.

QA-LSTM and the two proposed hybrid models are basically siamese networks (Chopra et al., 2005). These structures overlook another potential issue. The answers might be extremely long and contain lots of words that are not related to the question at hand. No matter what advanced neural networks are exploited at the answer side, the resulting representation might still be distracted by non-useful information. A typical example is the second candidate answer in Table 1. If the construction of the answer representation is not aware of the input question, the representation might be strongly influenced by n-grams such as "are widowed" and "your spouse's death", which are informative if we only look at the candidate answer, but are not so important for the input question. We address this problem by developing a simple attention model for the answer vector generation, in order to alleviate this weakness by dynamically aligning the more informative parts of answers to the questions.

Inspired by the work in (Hermann et al., 2015), we develop a very simple but efficient word-level attention on the basic model. In Figure 4, we detail our Attentive LSTM architecture. Prior to the average or mean pooling, each biLSTM output vector is multiplied by a softmax weight, which is determined by the question representation from biLSTM. Specifically, given the output vector of biLSTM on the answer side at time step t, $\mathbf{h}_a(t)$, and the question representation, $\mathbf{o}_q$, the updated vector $\widetilde{\mathbf{h}}_a(t)$ for each answer token are formulated below.

$$\mathbf{m}_{a,q}(t) = \mathbf{W}_{am}\mathbf{h}_a(t) + \mathbf{W}_{qm}\mathbf{o}_q \quad (11)$$

$$s_{a,q}(t) \propto \exp(\mathbf{w}_{ms}^T \tanh(\mathbf{m}_{a,q}(t))) \quad (12)$$

$$\widetilde{\mathbf{h}}_a(t) = \mathbf{h}_a(t) s_{a,q}(t) \quad (13)$$

where $\mathbf{W}_{am}$, $\mathbf{W}_{qm}$ and $\mathbf{w}_{ms}$ are attention parameters. Conceptually, the attention mechanism gives more weight to certain words of the candidate answer, where the weights are computed by taking into consideration information from the question. The expectation is that words in the candidate answer that are more important with regard to the input question should receive larger weights.

The attention mechanism in this paper is conceptually analogous to the one used in one-layer

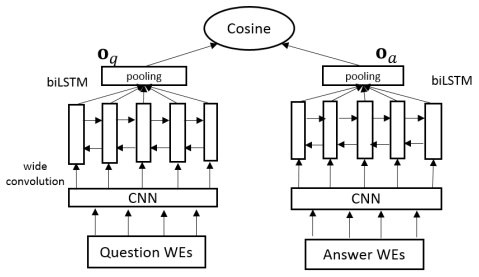


Figure 3: Convolution-based LSTM

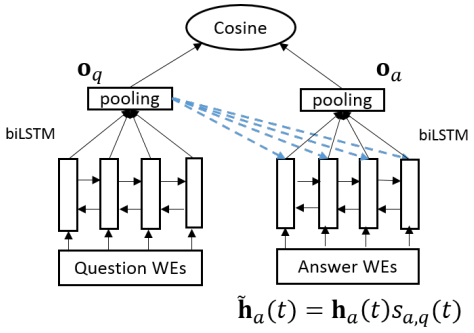


Figure 4: Attentive LSTM

	Train	Validation	Test1	Test2
# of Qs	12887	1000	1800	1800
# of As	18540	1454	2616	2593

Table 2: Numbers of Qs and As in InsuranceQA.

memory network (Sukhbaatar et al., 2015). The fundamental difference is that the transformed question vector and answer unit vectors are combined in an inner-product pattern in order to generate attentive weights in memory network, whereas this work adopts a summation operation (Eq. 11).

4 InsuranceQA Experiments

The first dataset we use to evaluate the proposed approaches is the InsuranceQA, which has been recently proposed by Feng et al. (2015). We use the first version of this dataset. This dataset contains question and answer pairs from the insurance domain and is already divided into a training set, a validation set, and two test sets. We do not see any obvious categorical differentiation between two tests' questions. We list the numbers of questions and answers of the dataset in Table 2. We refer the reader to (Feng et al., 2015), for more details regarding the InsuranceQA data. In this dataset, a question may have multiple correct answers, and normally the questions are much shorter than answers. The average length of questions in tokens is 7, while the average length of answers is 94. Such difference posts additional challenges for the answer selection task. This corpus contains 24981 unique answers in total. For the development and test sets, the InsuranceQA also includes an answer pool of 500 candidate answers for each question. These answer pools were constructed by including the correct answer(s) and randomly selected candidates from the complete set of unique answers. The top-1 accuracy of the answer selection is reported.

4.1 Setup

The proposed models are implemented with Theano (Bastien et al., 2012) and all experiments are conducted in a GPU cluster. We use the accuracy on validation set to select the best epoch and best hyper-parameter settings for testing.

The word embeddings are pre-trained, using word2vec (Mikolov et al., 2013) [3]. The training data for the word embeddings is a Wikipedia corpus of 164 million tokens combined with the questions and answers in the InsuranceQA training set. The word vector size is set to 100. Word embeddings are also part of the parameters and are optimized during the training. Stochastic Gradient Descent (SGD) is the optimization strategy. The learning rate λ is 1.1. We get the best performances when the negative answer count $K = 50$. We also tried different margins in the hing loss function, and finally fixed the margin as M=0.2. We train our models in mini-batches (with batch size as 20), and the maximum length L of questions and answers is 200. Any tokens out of this range are discarded. In order to get more obvious comparison between the proposed models and the basic framework, with respect to the ground-truth answer length in Fig. 5, we also provide the results of $K = 1$. In this case, we set $M = 0.1$, $\lambda = 0.1$ and mini-batches as 100 to get the best performance on the validation set. Also, the dimension of LSTM output vectors is 141x2 for bidirectional LSTM in QA-LSTM, Attentive LSTM and Convolutional-pooling LSTM, such that biLSTM has a comparable number of parameters with a single-direction LSTM with 200 dimensions. For Convolution-based LSTM, since LSTM structure is built on the top of CNN, we fixed the CNN output as 282 dimensions and tune the biLSTM hidden vector size in the experiments.

Because the sequences within a mini-batch have different lengths, we use a mask matrix to indicate the real length of each sequence.

4.2 Baselines

For comparison, we report the performances of four baselines in the top group in Table 3: two state-of-the-art non-DL approaches and two variations of a strong DL approach based on CNN.

Bag-of-word: The idf-weighted sum of word vectors is used as a feature vector. The candidates are ranked by the cosine similarity to the question.

Metzler-Bendersky IR model: A state-of-the-art weighted dependency model (Bendersky et al., 2010; Bendersky et al., 2011), which employs a weighted combination of term-based and term proximity-based features to score each candidate.

Architecture-II in (Feng et al., 2015): A CNN model is employed to learn distributed representations of questions and answers. Cosine similarity is used to rank answers.

[3] https://code.google.com/p/word2vec/

	Model	Validation	Test1	Test2
	Bag-of-word	31.9	32.1	32.2
	Metzler-Bendersky IR model	52.7	55.1	50.8
	CNN (Feng et al., 2015)	61.8	62.8	59.2
	CNN with GESD (Feng et al., 2015)	**65.4**	**65.3**	**61.0**
A	QA-LSTM (head/tail)	54.8	53.6	51.0
B	QA-LSTM (avg pooling,K=50)	55.0	55.7	52.4
C	QA-LSTM (max pooling,K=1)	64.3	63.1	58.0
D	QA-LSTM (max pooling,K=50)	66.6	66.6	63.7
E	Conv-pooling LSTM (c=4000,K=1)	66.2	64.6	62.2
F	Conv-pooling LSTM (c=200,K=50)	66.4	67.4	63.5
G	Conv-pooling LSTM (c=400,K=50)	67.8	67.5	64.4
H	Conv-based LSTM ($\|h\|$=200,K=50)	66.0	66.1	63.0
I	Conv-based LSTM ($\|h\|$=400,K=50)	67.1	67.6	64.4
J	QA-CNN (max-pooling, $k = 3$)	61.6	62.2	57.9
K	Attentive CNN (max-pooling, $k = 3$)	62.3	63.3	60.2
L	Attentive LSTM (avg-pooling K=1)	68.4	68.1	62.2
M	Attentive LSTM (avg-pooling K=50)	68.4	67.8	63.2
N	Attentive LSTM (max-pooling K=50)	**68.9**	**69.0**	**64.8**

Table 3: The experimental results of InsuranceQA.

Architecture-II with Geometricmean of Euclidean and Sigmoid Dot product (GESD): Cosine similarity is replaced by GESD, which got the best performance in (Feng et al., 2015).

4.3 Results and discussions

In this section, we provide detailed analysis on the experimental results. Table 3 summarizes the results of our models on InsuranceQA. From Row (A) to (D), we list QA-LSTM without either CNN structure or attention. They vary on the pooling method used. We can see that by concatenating the last vectors from both directions, (A) performs the worst. We see that using max-pooling (C) is much better than average pooling (B). The potential reason may be that the max-pooling extracts more local values for each dimension. Compared to (C), (D) is better, showing the need of multiple negative answers in training.

Row (E) to (I) show the results of Convolutional-pooling LSTMs and Convolution-based LSTMs with different filter sizes c, biLSTM hidden sizes $|h|$ and negative answer pool size K. Increasing the negative answer pool size, we are allowed to use less filter counts (F vs E). Larger filter counts help on the test accuracies (G vs F) for Convolutional-pooling LSTMs. We have the same observation with larger biLSTM hidden vector size for Convolution-based LSTMs. Both convolutional models outperform the plain QA-LSTM (D) by about 1.0% on test1, and 0.7% on test2.

Rows (L-N) correspond to QA-LSTM with the attention model, with either max-pooling or average pooling. We observe that max-pooling is better than avg-pooling, which is consistent with QA-LSTMs. In comparison to Model (D), Model (N) shows over 2% improvement on both validation and Test1 sets. And (N) gets improvements over the best baseline in Table 3 by 3.5%, 3.7% and 3.8% on the validation, Test1 and Test2 sets, respectively. Compared to Architecture II in (Feng et al., 2015), which involved a large number of CNN filters, (N) model also has fewer parameters.

We also test the proposed attention mechanism on convolutional networks. (J) replaces the LSTM in QA-LSTM with a convolutional layer. We set the filter size $c = 400$ and window size $k = 3$ according to the validation accuracy. (K) performs the similar attention on the convolutional output of the answers. Similar to biLSTM, the attention on the convolutional layer gives over 2% accuracy improvement on both test sets, which proves the attention's efficiency on both CNN and RNN structures.

Finally, we investigate the proposed models on how they perform with respect to long answers. To better illustrate the performance difference, we

Models	MAP	MRR
(Yao et al., 2013)	0.631	0.748
(Severyn and Moschitti, 2013)	0.678	0.736
(Yih et al., 2013)-BDT	0.694	0.789
(Yih et al., 2013)-LCLR	0.709	0.770
(Wang and Nyberg, 2015)	0.713	0.791
Architecture-II (Feng et al., 2015)	0.711	0.800
(Severyn and Moschitti, 2015) w/o additional features	0.671	0.728
(Severyn and Moschitti, 2015) with additional features	**0.746**	**0.808**
A. QA-CNN	0.714	0.807
B. QA-LSTM (max-pooling)	0.733	0.819
C. Conv-pooling LSTM	0.742	0.819
D. Conv-based LSTM	0.737	0.827
E. Attentive LSTM	**0.753**	**0.830**

Table 4: The test set results on TREC-QA

compare the models with $K = 1$ (i.e. the models C, E, L). We divide the questions of Test1 and Test2 sets into eleven buckets, according to the average length of their ground truth answers. As shown in Figure 5, QA-LSTM gets better or similar performance compared to the proposed models on buckets with shorter answers ($L \leq 50$, $50 < L \leq 55$, $55 < L \leq 60$). As the answer lengths increase, the gap between QA-LSTM and other models becomes more obvious. It suggests the effectiveness of Convolutional-pooling LSTM and Attentive LSTM for long-answer questions.

In (Feng et al., 2015), GESD outperforms cosine similarity in their models. However, the proposed models with GESD as similarity scores do not provide any improvement on the accuracy.

5 TREC-QA Experiments

In this section we detail our experimental setup and results using the TREC-QA dataset.

5.1 Data, metrics and baselines

We test the models on TREC-QA dataset, created based on Text REtrieval Conference (TREC) QA track (8-13) data. More detail of the generation steps for this data can be found in (Wang et al., 2007). We follow the exact approach of train/dev/test questions selection in (Wang and Nyberg, 2015), in which all questions with only positive or negative answers are removed. Finally, we have 1162 training, 65 development and 68 test questions. Similar to previous work, we use Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) as evaluation metrics, which are evaluated using the official scripts.

In the top part of Table 4, we list the performance of recent prior work on this dataset. We implemented the Architecture II in (Feng et al., 2015) from scratch. The CNN structure in (Severyn and Moschitti, 2015) combined with additional human-designed features achieved the best MAP and MRR.

5.2 Setup

We keep the configurations same as those in InsuranceQA in section 4.1, except the following differences: 1) Following Wang and Nyberg (2015), we use 300-dimensional vectors that were trained and provided by word2vec (Mikolov et al., 2013) using a part of the Google News dataset [4]. 2) Since the word vectors of TREC-QA have a greater dimension than InsuranceQA, we accordingly have larger biLSTM hidden vectors and CNN filters, in order not to lose information from word vectors. Here we set both of them as 600. 3) We use the models from the epoch with the best MAP on the validation set. 4) We also observe that because of the smaller data size, we need a decayed learning rate λ in order to stablize the models' training. Specifically, we set the initial $\lambda_0 = 1.1$, and decrease it for each epoch $T > 1$ as $\lambda_T = \lambda_0/T$. 5) We fix the negative answer size $K = 50$ during training.

5.3 Results

The bottom part of Table 4 shows the performance of the proposed models. For the comparison purpose, we replace biLSTM with a convolution in Model (A), and also use max-pooling to get question and answer embeddings, and call this model QA-CNN. QA-LSTM (B) improves MAP and MRR in more than 1% when compared to QA-CNN (A). Compared to (B), convolutional-pooling (C) performs better on MAP by 0.9%, and convolution-based models on MAP by 0.4% and MRR by 0.8%. Attentive LSTM is the best proposed model, and outperforms the best baseline (Severyn and Moschitti, 2015) by 0.7% on MAP and 2.2% on MRR. Note that the best result in (Severyn and Moschitti, 2015) was obtained by combining CNN-based features with additional human-defined features. In contrast, our attentive LSTM model achieves higher performance without using any human-defined features.

6 Conclusion

In this paper, we address the following problem for the answer passage selection: how can we construct the embeddings for questions and candidate

[4] https://code.google.com/archive/p/word2vec/

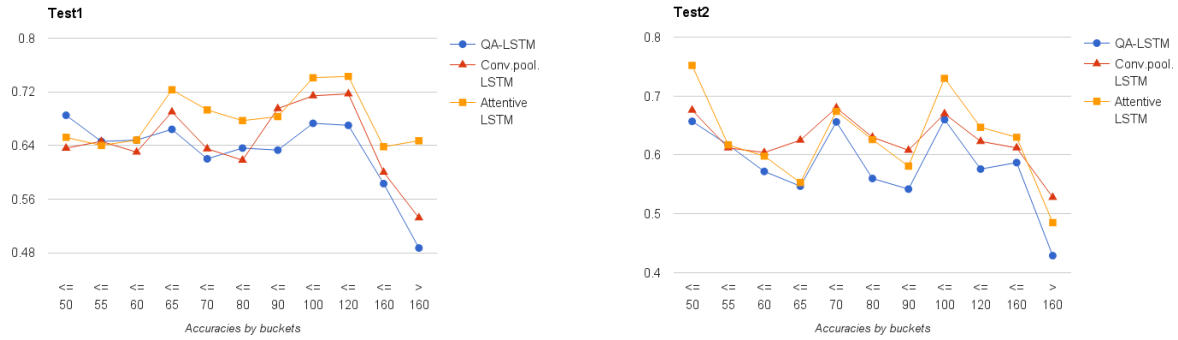


Figure 5: The accuracy of Test1 and Test2 of InsuranceQA sets for three models, i.e. maxpooling QA-LSTM (C), Convolutional-pooling LSTM (E) and Attentive LSTM (L) in Table 3, on different levels of ground truth answer lengths on each test set. The figures show the accuracy of each bucket.

answers, in order to better distinguish the correct answers from other candidates? We propose three independent models in two directions. First, we develop two hybrid models which combine the strength of both recurrent and convolutional neural networks. Second, we introduce a simple one-way attention mechanism, in order to generate answer embeddings influenced by the question context. Such attention fixes the issue of independent generation of the question and answer embeddings in previous work. All proposed models are departed from a basic architecture, built on bidirectional LSTMs. We conduct experiments on InsuranceQA and TREC-QA datasets, and the experimental results demonstrate that the proposed models outperform a variety of strong baselines. Potential future work include: 1) Evaluating the proposed approaches for different tasks, such as community QA and textual entailment; 2) Including the sentential attention mechanism; 3) Integrating the hybrid and the attentive mechanisms into a single framework.

References

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua. Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of International conference of learning representations*.

Frederic Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2010. Learning concept importance using a weighted dependence model. In *in Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM)*.

Michael Bendersky, Donald Metzler, and W. Bruce Croft. 2011. Parameterized concept weighting in verbose queries. In *in Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. *Computer Vision and Pattern Recognition (CVPR)*.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Cícero dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of ACL*, pages 694–699, Beijing, China, July.

Cícero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *CoRR*, abs/1602.03609.

Minwei Feng, Bing Xiang, Michael Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. *In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. *Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics (NAACL)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *In Advances in Neural Information Processing Systems (NIPS)*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. *Advances in Neural Information Processing Systems (NIPS)*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems (NIPS)*.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomás Kociský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. *International Conference on Learning Representations (ICLR)*.

Alexander Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for sentence summarization. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tara N. Sainath, Andrew Senior Oriol Vinyals, and Hasim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. *In Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. *In Advances in Neural Information Processing Systems (NIPS)*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *In Advances in Neural Information Processing Systems (NIPS)*.

Oriol Vinyals and Quoc V. Le. 2015. A neural conversational model. *Proceedings of the 31st International Conference on Machine Learning*.

Mengqiu Wang and Christopher Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. *The Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Mengqiu Wang, Noah Smith, and Mitamura Teruko. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. *The Proceedings of EMNLP-CoNLL*.

Jason Weston, Sumit Chopra, and Keith Adams. 2014. #tagspace: Semantic embeddings from hashtags. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Xuchen Yao, Benjamin Durme, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. *Proceedings of NAACL-HLT*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. *Proceedings of the 51st Annual Meeting of the Association for Computational Linguist (ACL)*.

Wenpeng Yin, Hinrich Schutze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: attention-based convolutional neural network for modeling sentence pairs. *CoRR*, abs/1512.05193.

Lei Yu, Karl M. Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS Deep Learning Workshop*.

Zhen Zuo, Bing Shuai, Gang Wang, Xiao Liu, Xingxing Wang, Bing Wang, and Yushi Chen. 2015. Convolutional recurrent neural networks: Learning spatial dependencies for image representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*.

Tables as Semi-structured Knowledge for Question Answering

Sujay Kumar Jauhar
Carnegie Mellon University
Pittsburgh, PA, USA
sjauhar@cs.cmu.edu

Peter D. Turney
Allen Institute for Artificial Intelligence
Seattle, WA, USA
petert@allenai.org

Eduard Hovy
Carnegie Mellon University
Pittsburgh, PA, USA
hovy@cs.cmu.edu

Abstract

Question answering requires access to a knowledge base to check facts and reason about information. Knowledge in the form of natural language text is easy to acquire, but difficult for automated reasoning. Highly-structured knowledge bases can facilitate reasoning, but are difficult to acquire. In this paper we explore tables as a semi-structured formalism that provides a balanced compromise to this trade-off. We first use the structure of tables to guide the construction of a dataset of over 9000 multiple-choice questions with rich alignment annotations, easily and efficiently via crowd-sourcing. We then use this annotated data to train a semi-structured feature-driven model for question answering that uses tables as a knowledge base. In benchmark evaluations, we significantly outperform both a strong unstructured retrieval baseline and a highly-structured Markov Logic Network model.

1 Introduction

Question answering (QA) has emerged as a practical research problem for pushing the boundaries of artificial intelligence (AI). Dedicated projects and open challenges to the research community include examples such as Facebook AI Research's challenge problems for AI-complete QA (Weston et al., 2015) and the Allen Institute for AI's (AI2) Aristo project (Clark, 2015) along with its recently completed Kaggle competition[1]. The reason for this emergence is the diversity of core language and reasoning problems that a complex, integrated task like QA exposes: information extraction (Srihari and Li, 1999), semantic modelling (Shen and Lapata, 2007; Narayanan and Harabagiu, 2004), logic and reasoning (Moldovan et al., 2003), and inference (Lin and Pantel, 2001).

Complex tasks such as QA require some form of knowledge base to store facts about the world and reason over them. By knowledge base, we mean any form of knowledge: structured (e.g., tables, ontologies, rules) or unstructured (e.g., natural language text). For QA, knowledge has been harvested and used in a number of different modes and formalisms: large-scale extracted and curated knowledge bases (Fader et al., 2014), structured models such as Markov Logic Networks (Khot et al., 2015), and simple text corpora in information retrieval approaches (Tellex et al., 2003).

There is, however, a fundamental trade-off in the structure and regularity of a formalism and its ability to be curated, modelled or reasoned with easily. For example, simple text corpora contain no structure, and are therefore hard to reason with in a principled manner. Nevertheless, they are easily and abundantly available. In contrast, Markov Logic Networks come with a wealth of theoretical knowledge connected with their usage in principled inference. However, they are difficult to induce automatically from text or to build manually.

In this paper we explore tables as semi-structured knowledge for multiple-choice question (MCQ) answering. Specifically, we focus on tables that represent general knowledge facts, with cells that contain free-form text (Secton 3 details the nature and semantics of these tables). The structural properties of tables, along with their free-form text content represents a semi-structured balanced compromise in the trade-off between degree of structure and ubiquity. We present two main contributions, with tables and their structural properties playing a crucial role in both. First,

[1] https://www.kaggle.com/c/the-allen-ai-science-challenge

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 474–483,
Berlin, Germany, August 7-12, 2016.

we crowd-source a collection of over 9000 MCQs with alignment annotations to table elements, using tables as guidelines in efficient data harvesting. Second, we develop a feature-driven model that uses these MCQs to perform QA, while fact-checking and reasoning over tables.

Others have used tables in the context of QA. Question bank creation for tables has been investigated (Pasupat and Liang, 2015), but without structural guidelines or the alignment information that we propose. Similarly, tables have been used in QA reasoning (Yin et al., 2015b; Neelakantan et al., 2015; Sun et al., 2016) but have not explicitly attempted to encode all the semantics of table structure (see Section 3.1). To the best of our knowledge, no previous work uses tables for both creation and reasoning in a connected framework.

We evaluate our model on MCQ answering for three benchmark datasets. Our results consistently and significantly outperform a strong retrieval baseline as well as a Markov Logic network model (Khot et al., 2015). We thus show the benefits of semi-structured data and models over unstructured or highly-structured counterparts. We also validate our curated MCQ dataset and its annotations as an effective tool for training QA models. Finally, we find that our model learns generalizations that permit inference when exact answers may not even be contained in the knowledge base.

2 Related Work

Our work with tables, semi-structured knowledge bases and QA relates to several parallel lines of research. In terms of dataset creation via crowdsourcing, Aydin et al. (2014) harvest MCQs via a gamified app, although their work does not involve tables. Pasupat and Liang (2015) use tables from Wikipedia to construct a set of QA pairs. However their annotation setup does not impose structural constraints from tables, and does not collect fine-grained alignment to table elements.

On the inference side Pasupat and Liang (2015) also reason over tables to answer questions. Unlike our approach, they do not require alignments to table cells. However, they assume knowledge of the table that contains the answer, a priori – which we do not. Yin et al. (2015b) and Neelakantan et al. (2015) also use tables in the context of QA, but deal with synthetically generated query data. Sun et al. (2016) perform cell search over web tables via relational chains, but are more generally interested in web queries. Clark et al. (2016) combine different levels of knowledge for QA, including an integer-linear program for searching over table cells. None of these other efforts leverage tables for generation of data.

Our research more generally pertains to natural language interfaces for databases. Answering questions in this context refers to executing queries over relational databases (Cafarella et al., 2008; Pimplikar and Sarawagi, 2012). Yin et al. (2015a) consider databases where information is stored in n-tuples, which are essentially tables. Also, investigation of the relational structure of tables is connected with research on database schema analysis and induction (Venetis et al., 2011; Syed et al., 2010). Finally, unstructured text and structured formats links to work on open information extraction (Etzioni et al., 2008) and knowledge base population (Ji and Grishman, 2011).

3 Tables as Semi-structured Knowledge Representation

Tables can be found on the web containing a wide range of heterogenous data. To focus and facilitate our work on QA we select a collection of tables that were specifically designed for the task. Specifically we use AI2's Aristo Tablestore[2]. However, it should be noted that the contributions of this paper are not tied to specific tables, as we provide a general methodology that could equally be applied to a different set of tables. The structural properties of this class of tables is further described in Section 3.1.

The Aristo Tablestore consists of 65 hand-crafted tables organized by topic. Some of the topics are bounded, containing only a fixed number of facts, such as the possible phase changes of matter (see Table 1). Other topics are unbounded, containing a very large or even infinite number of facts, such as the kind of energy used in performing an action (the corresponding tables can only contain a sample subset of these facts). A total of 3851 facts (one fact per row) are present in the manually constructed tables. An individual table has between 2 and 5 content columns.

The target domain for these tables is two 4th grade science exam datasets. The majority of the tables were constructed to contain topics and facts

[2] http://allenai.org/content/data/AristoTablestore-Nov2015Snapshot.zip

Phase Change		Initial State		Final State		Form of Energy Transfer
Melting	causes a	solid	to change into a	liquid	by	adding heat
Vaporization	causes a	liquid	to change into a	gas	by	adding heat
Condensation	causes a	gas	to change into a	liquid	by	removing heat
Sublimation	causes a	solid	to change into a	gas	by	adding heat

Table 1: Part of a table concerning phase changes in matter. Rows are facts. Columns without header text provide filler text, so that each row forms a sentence. In columns with header text, the header describes the type of entry in the column; the header is a hypernym of the text in the body below.

from the publicly available Regents dataset[3]. The rest were targeted at an unreleased dataset called Monarch. In both cases only the training partition of each dataset was used to formulate and handcraft tables. However, for unbounded topics, additional facts were added to each table, using science education text books and websites.

3.1 Table Semantics and Relations

Part of a table from the Aristo Tablestore is given as an example in Table 1. The format is semi-structured: the rows of the table (with the exception of the header) are a list of sentences, but with well-defined recurring filler patterns. Together with the header, these patterns divide the rows into meaningful columns. This semi-structured data format is flexible. Since facts are presented as sentences, the tables can act as a text corpus for information retrieval. At the same time the structure can be used – as we do – to focus on specific nuggets of information. The flexibility of these tables allows us to compare our table-based system to an information retrieval baseline.

Such tables have some interesting structural semantics, which we will leverage throughout the paper. A row in a table corresponds to a *fact*[4]. The cells in a row correspond to concepts, entities, or processes that participate in this fact. A content column[5] corresponds to a group of concepts, entities, or processes that are the same *type*. The header cell of the column is an abstract description of the *type*. We may view the head as a hypernym and the cells in the column below as co-hyponyms of the head. The header row defines a generalization of which the rows in the table are specific instances.

This structure is directly relevant to multiple-choice QA. Facts (rows) form the basis for creating or answering questions, while instances of a type (columns) act as the choices of an MCQ. We use these observations both for crowd-sourcing MCQ creation as well as for designing features to answer MCQs with tables.

4 Crowd-sourcing Multiple-choice Questions from Tables

We use Amazon's Mechanical Turk (MTurk) service to generate MCQs by imposing constraints derived from the structure of the tables. These constraints help annotators create questions with scaffolding information, and lead to consistent quality in the generated output. An additional benefit of this format is the alignment information, linking cells in the tables to the MCQs generated by the Turkers. The alignment information is generated as a by-product of making the MCQs.

We present Turkers with a table such as the one in Figure 1. Given this table, we choose a target cell to be the correct answer for a new MCQ; for example, the red cell in Figure 1. First, Turkers create a question by using information from the rest of the row containing the target (i.e., the blue cells in Figure 1), such that the target is its correct answer. Then they select the cells in the row that they used to construct the question. Following this, they construct four succinct choices for the question, one of which is the correct answer and the other three are distractors. Distractors are formed from other cells in the column containing the target (i.e. yellow cells in Figure 1). If there are insufficient unique cells in the column Turkers create their own. Annotators can rephrase and shuffle the contents of cells as required.

In addition to an MCQ, we obtain alignment information with no extra effort from annotators. We know which table, row, and column contains the answer, and thus we know which header cells might be relevant to the question. We also know the cells of a row that were used to construct a question.

[3] `http://allenai.org/content/data/Regents.zip`

[4] Also predicates, or more generally frames with typed arguments.

[5] Different from filler columns, which only contain a recurring pattern, and no information in their header cells.

PHASE CHANGE		INITIAL PHASE		FINAL PHASE		HEAT TRANSFER
Melting	causes a	solid	to change into a	liquid	by	adding heat
Vaporization	causes a	liquid	to change into a	gas	by	adding heat
Sublimation	causes a	solid	to change into a	gas	by	adding heat
Freezing	causes a	liquid	to change into a	solid	by	removing heat
Deposition	causes a	gas	to change into a	solid	by	removing heat
Condensation	causes a	gas	to change into a	liquid	by	removing heat

Figure 1: Example table from MTurk annotation task illustrating constraints. We ask Turkers to construct questions from blue cells, such that the red cell is the correct answer, and yellow cells form distractors.

Task	Avg. Time (s)	$/hour	% Reject
Rewrite	345	2.61	48
Paraphrase	662	1.36	49
Add choice	291	2.47	24
Write new	187	**5.78**	38
TabMCQ	**72**	5.00	**2**

Table 2: Comparison of different ways of generating MCQs with MTurk.

What is the orbital event with the longest day and the shortest night?
A) **Summer solstice**
B) Winter solstice
C) Spring equinox
D) Fall equinox
Steel is a/an _____ of electricity
A) Separator
B) Isolator
C) Insulator
D) Conductor

Table 3: Examples of MCQs generated by MTurk. Correct answer choices are in bold.

4.1 The TabMCQ Dataset

We created a HIT (the MTurk acronym for Human Intelligence Task) for every non-filler cell (see Section 3) from each one of the 65 manually constructed tables of the Aristo Tablestore. We paid annotators 10 cents per MCQ, and asked for 1 annotation per HIT for most tables. For an initial set of four tables which we used in a pilot study, we asked for three annotations per HIT[6]. We required Turkers to have a HIT approval rating of 95% or higher, with a minimum of at least 500 HITs approved. We restricted the demographics of our workers to the US.

Table 2 compares our method with other studies conducted at AI2 to generate MCQs. These methods attempt to generate new MCQs from existing ones, or write them from scratch, but do not involve tables in any way. Our annotation procedure leads to faster data creation, with consistent output quality that resulted in the lowest percentage of rejected HITs. Manual inspection of the generated output also revealed that questions are of consistently good quality. They are good enough for training machine learning models and many are good enough as evaluation data for QA. A sample of generated MCQs is presented in Table 3.

We implemented some simple checks to evaluate the data before approving HITs. These included things like checking whether an MCQ has at least three choices and whether choices are repeated. We had to further prune our data to discard some MCQs due to corrupted data or badly constructed MCQs. A total of 159 MCQs were lost through the cleanup. In the end our complete data consists of 9092 MCQs, which is – to the best of our knowledge – orders of magnitude larger than any existing collection of science exam style MCQs available for research. These MCQs also come with alignment information to tables, rows, columns and cells. The dataset, bundled together with the Aristo Tablestore, can be freely downloaded[7].

5 Solving MCQs with Table Cell Search

Consider the MCQ "What is the process by which water is changed from a liquid to a gas?" with choices "melting, sublimation, vaporization, condensation", and the table given in Figure 1. Finding the correct answer amounts to finding a cell in the table that is most relevant to a candidate QA pair. In other words, a relevant cell should confirm the assertion made by a particular QA pair.

By applying the reasoning used to create MCQs

[6]The goal was to obtain diversity in the MCQs created for a target cell. The results were not sufficiently conclusive to warrant a threefold increase in the cost of creation.

[7]http://ai2-website.s3.amazonaws.com/data/TabMCQ_v_1.0.zip

(see Section 4) in the inverse direction, finding these relevant cells becomes the task of finding an intersection between rows and columns of interest. Consider the table in Figure 1: assuming we have some way of aligning a question to a row (blue cells) and choices to a column (yellow cells), then the relevant cell is at the intersection of the two (the red cell). This alignment is precisely what we get as a by-product of the annotation task we setup in Section 4 to harvest MCQs.

We can thus featurize connections between MCQs and elements of tables and use the alignment data to train a model over the features. This is outlined in the next section, describing our Feature Rich Table Embedding Solver (FRETS).

5.1 Model and Training Objective

Let $\mathcal{Q} = \{q_1, ..., q_N\}$ denote a set of MCQs, and $\mathcal{A}_n = \{a_n^1, ..., a_n^k\}$ be the set of candidate answer choices for a given question q_n. Let the set of tables be defined as $\mathcal{T} = \{T_1, ..., T_M\}$. Given a table T_m, let t_m^{ij} be the cell in that table corresponding to the ith row and jth column.

We define a log-linear model that scores every cell t_m^{ij} of every table in our collection according to a set of discrete weighted features, for a given QA pair. We have the following:

$$\log p(t_m^{ij} | q_n, a_n^k; \mathcal{A}_n, \mathcal{T}) = \sum_d \lambda_d f_d(q_n, a_n^k, t_m^{ij}; \mathcal{A}_n, \mathcal{T}) - \log Z \quad (1)$$

Here λ_d are weights and $f_d(q_n, a_n^k, t_m^{ij}; \mathcal{A}_n, \mathcal{T})$ are features. These features should ideally leverage both structure and content of tables to assign high scores to relevant cells, while assigning low scores to irrelevant cells. Z is the partition function, defined as follows:

$$Z = \sum_{m,i,j} \exp\left(\sum_d \lambda_d f_d(q_n, a_n^k, t_m^{ij}; \mathcal{A}_n, \mathcal{T})\right) \quad (2)$$

Z normalizes the scores associated with every cell over all the cells in all the tables to yield a probability distribution. During inference the partition term $\log Z$ can be ignored, making scoring cells of every table for a given QA pair efficient.

These scores translate to a solution for an MCQ. Every QA pair produces a hypothetical fact, and as noted in Section 3.1, the row of a table is in essence a fact. Relevant cells (if they exist) should confirm the hypothetical fact asserted by a given QA pair. During inference, we assign the score of the highest scoring row (or the most likely fact) to a hypothetical QA pair. Then the correct solution to the MCQ is simply the answer choice associated with the QA pair that was assigned the highest score. Mathematically, this is expressed as follows:

$$a_n^* = \arg\max_{a_n^k} \max_{m,i} \sum_j \sum_d \lambda_d f_d(q_n, a_n^k, t_m^{ij}; \mathcal{A}_n, \mathcal{T}) \quad (3)$$

5.1.1 Training

Since FRETS is a log-linear model, training involves optimizing a set of weights λ_d. As training data, we use alignment information between MCQs and table elements (see Section 4.1). The predictor value that we try to maximize with our model is an alignment score that is closest to the true alignments in the training data. True alignments to table cells for a given QA pair are essentially indicator values but we convert them to numerical scores as follows[8]. For a correct QA hypothesis we assign a score of 1.0 to cells whose row and column and both aligned to the MCQ (i.e. cells that exactly answer the question), 0.5 to cells whose row but not column is aligned in some way to the question (i.e. cells that were used to construct the question), and 0.0 otherwise. For an incorrect QA hypothesis we assign a score of 0.1 to random cells from tables that contain no alignments to the QA (so all except one), with a probability of 1%, while all other cells are scored 0.0. The intuition behind this scoring scheme is to guide the model to pick relevant cells for correct answers, while encouraging it to pick faulty evidence with low scores for incorrect answers.

Given these scores assigned to all cells of all tables for all QA pairs in the training set, suitably normalized to a probability distribution over tables for a given QA pair, we can then proceed to train our model. We use cross-entropy, which minimizes the following loss:

[8] On training data, we experimented with a few different scoring heuristics and found that these ones worked well.

Level	Feature	Description	Intuition	S-Var	Cmpct
Table	Table score	Ratio of words in **t** to **q+a**	Topical consistency	◊	
	†TF-IDF table score	Same but TF-IDF weights	Topical consistency	◊	●
Row	Row-question score	Ratio of words in **r** to **q**	Question align	◊	●
	Row-question w/o focus score	Ratio of words in **r** to **q-($\mathbf{a_f}$+$\mathbf{q_f}$)**	Question align	◊	
	Header-question score	Ratio of words in **h** to **q**	Prototype align	◊	
Column	Column overlap	Ratio of elements in **c** and **A**	Choices align	◊	●
	Header answer-type match	Ratio of words in $\mathbf{c_h}$ to $\mathbf{a_f}$	Choices hypernym align	◊	●
	Header question-type match	Ratio of words in $\mathbf{c_h}$ to $\mathbf{q_f}$	Question hypernym align	◊	
Cell	†Cell salience	Salience of **s** to **q+a**	QA hypothesis assert	◊	●
	†Cell answer-type entailment	Entailment score between **s** and $\mathbf{a_f}$	Hypernym-hyponym align		●
	Cell answer-type similarity	Avg. vector sim between **s** and $\mathbf{a_f}$	Hypernym-hyponym sim.		

Table 4: Summary of features. For a question (**q**) and answer (**a**) we compute scores for elements of tables: whole tables (**t**), rows (**r**), header rows (**h**), columns (**c**), column headers ($\mathbf{c_h}$) and cells (**s**). Answer-focus ($\mathbf{a_f}$) and question-focus ($\mathbf{q_f}$) terms added where appropriate. Features marked ◊ denote soft-matching variants, marked with while those marked with a † are described in further detail in Section 5.2. Finally, ● features denote those that received high weights during training with all features, and were subsequently selected to form a compact FRETS model.

$$L(\vec{\lambda}) = \sum_{\substack{q_n \\ a_n^k \in \mathcal{A}_n}} \sum_{m,i,j} p(t_m^{*ij} | q_n, a_n^k; \mathcal{T}) \cdot \log p(t_m^{ij} | q_n, a_n^k; \mathcal{A}_n, \mathcal{T}) \quad (4)$$

Here $p(t_m^{*ij} | q_n, a_n^k; \mathcal{T})$ is the normalized probability of the true alignment scores.

While this is an indirect way to train our model to pick the best answer, in our pilot experiments it worked better than direct maximum likelihood or ranking with hinge loss, achieving a training accuracy of almost 85%. Our experimental results on the test suite, presented in the next section, also support the empirical effectiveness of this approach.

5.2 Features

The features we use are summarized in Table 4. These features compute statistics between question-answer pairs and different structural components of tables. While the features are weighted and summed for each cell individually, they can capture more global properties such as scores associated with tables, rows or columns in which the specific cell is contained. Features are divided into four broad categories based on the level of granularity at which they operate. In what follows we give some details of Table 4 that require further elaboration.

5.2.1 Soft matching

Many of the features that we implement are based on string overlap between bags of words. However, since the tables are defined statically in terms of a fixed vocabulary (which may not necessarily match words contained in an MCQ), these overlap features will often fail. We therefore soften the constraint imposed by hard word overlap by a more forgiving soft variant. More specifically we introduce a word-embedding based soft matching overlap variant for every feature in the table marked with ◊. The soft variant targets high recall while the hard variant aims at providing high precision. We thus effectively have almost twice the number of features listed.

Mathematically, let a hard overlap feature define a score $|S_1 \cap S_2| \ / \ |S_1|$ between two bags of words S_1 and S_2. We can define the denominator S_1 here, without loss of generality. Then, a corresponding word-embedding soft overlap feature is given by this formula:

$$\frac{1}{|S_1|} \sum_{w_i \in S_1} \max_{w_j \in S_2} sim(\vec{w_i}, \vec{w_j}) \quad (5)$$

Intuitively, rather than matching a word to its exact string match in another set, we instead match it to its most similar word, discounted by the score of that similarity.

5.2.2 Question parsing

We parse questions to find the desired answer-type and, in rarer cases, question-type words. For example, in the question "What form of energy is required to convert water from a liquid to a gas?", the type of the answer we are expecting is a "form of energy". Generally, this answer-type corresponds to a hypernym of the answer choices, and can help find relevant information in the table, specifically related to columns.

By carefully studying the kinds of question patterns in our data, we implemented a rule-based parser that finds answer-types from queries. This parser uses a set of hand-coded regular expressions over phrasal chunks. The parser is designed to have high accuracy, so that we only produce an output for answer-types in high confidence situations. In addition to producing answer-types, in some rarer cases we also detect hypernyms for parts of the questions. We call this set of words question-type words. Together, the question-type and answer-type words are denoted as focus words in the question.

5.2.3 TF-IDF weighting

TF-IDF scores for weighting terms are precomputed for all words in all the tables. We do this by treating every table as a unique document. At run-time we discount scores by table length as well as length of the QA pair under consideration to avoid disproportionately assigning high scores to large tables or long MCQs.

5.2.4 Salience

The salience of a string for a particular QA pair is an estimate of how relevant it is to the hypothesis formed from that QA pair. It is computed by taking words in the question, pairing them with words in an answer choice and then computing PMI statistics between these pairs from a large corpus. A high salience score indicates words that are particularly relevant for a given QA pair hypothesis.

5.2.5 Entailment

To calculate the entailment score between two strings, we use several features, such as overlap, paraphrase probability, lexical entailment likelihood, and ontological relatedness, computed with n-grams of varying lengths.

5.2.6 Normalization

All the features in Table 4 produce numerical scores, but the range of these scores vary to some extent. To make our final model more robust, we normalize all feature scores to have a range between 0.0 and 1.0. We do this by finding the maximum and minimum values for any given feature on a training set. Subsequently, instead of using the raw feature value of a feature f_d, we instead replace it with $(f_d - \min f_d) \,/\, (\max f_d - \min f_d)$.

6 Experimental Results

We train FRETS (Section 5) on the TabMCQ dataset (Section 4) using adaptive gradient descent with an L2 penalty of 1.0 and a mini-batch size of 500 training instances. We train two variants: one consisting of all the features from Table 4, the other – a compact model – consisting of the most important features (above a threshold) from the first model by feature-weight. These features are noted by ● in the final column of Table 4.

We run experiments on three 4th grade science exam MCQ datasets: the publicly available Regents dataset, the larger but unreleased dataset called Monarch, and a third even larger public dataset of Elementary School Science Questions (ESSQ)[9]. For the first two datasets we use the test splits only, since the training sets were directly studied to construct the Aristo Tablestore, which was in turn used to generate our TabMCQ training data. On ESSQ we use all the questions since they are independent of the tables. The Regents test set consists of 129 MCQs, the Monarch test set of 250 MCQs, and ESSQ of 855 MCQs.

Since we are investigating semi-structured models, we compare against two baselines. The first is an unstructured information retrieval method, which uses the Lucene search engine. To apply Lucene to the tables, we ignore their structure and simply use rows as plain-text sentences. The score for top retrieved hits are used to rank the different choices of MCQs. The second baseline is the highly-structured Markov-logic Network (MLN) model from Khot et al. (2015) as reported in Clark et al. (2016), who use the model as a baseline[10]. Note that Clark et al. (2016) achieve a score of 71.3 on Regents Test, which is higher than FRETS' scores (see Table 5), but their results are not comparable to ours because they use an ensemble of algorithms. In contrast, we use a single algorithm with a much smaller collection of knowledge. FRETS rivals the best individual algorithm from their work.

We primarily use the tables from the Aristo Tablestore as knowledge base data in three different settings: with only tables constructed for Regents (40 tables), with only supplementary tables constructed for Monarch (25 tables), and with all ta-

[9] http://aristo-public-data.s3.amazonaws.com/AI2-Elementary-NDMC-Feb2016.zip

[10] We do not re-implement the MLN, and therefore only cite results from previous work on part of our test suite.

Model	Data	Regents Test	Monarch Test	ESSQ
Lucene	Regents Tables	37.5	32.6	36.9
	Monarch Tables	28.4	27.3	27.7
	Regents+Monarch Tables	34.8	35.3	37.3
	Waterloo Corpus	55.4	51.8	54.4
MLN (Khot et al., 2015)	-	47.5	-	-
FRETS (Compact)	Regents Tables	**60.7**	47.2	51.0
	Monarch Tables	56.0	45.6	48.4
	Regents+Monarch Tables	59.9	47.6	50.7
FRETS	Regents Tables	59.1	**52.8**	54.4
	Monarch Tables	52.9	49.8	49.5
	Regents+Monarch Tables	59.1	52.4	**54.9**

Table 5: Evaluation results on three benchmark datasets using different sets of tables as knowledge bases. Best results on a dataset are highlighted in bold.

bles together (all 65 tables; see Section 3). For the Lucene baseline we also experiment with several orders of magnitude more data by indexing over the 5×10^{10} words Waterloo corpus compiled by Charles Clarke at the University of Waterloo. Data is not a variable for MLN, since we directly cite results from Clark et al. (2016).

The word vectors we used in soft matching feature variants (i.e., $\Diamond$ features from Table 4) for all our experiments were trained on 300 million words of Newswire English from the monolingual section of the WMT-2011 shared task data. These vectors were improved post-training by retrofitting (Faruqui et al., 2014) them to PPDB (Ganitkevitch et al., 2013).

The results of these experiments is presented in Table 5. All numbers are reported in percentage accuracy. We perform statistical significance testing on these results using Fisher's exact test with a p-value of 0.05 and report them in our discussions.

First, FRETS – in both full and compact form – consistently outperforms the baselines, often by large margins. For Lucene, the improvements over all but the Waterloo corpus baseline are statistically significant. Thus FRETS is able to capitalize on data more effectively and rival an unstructured model with access to orders of magnitude more data. For MLN, the improvements are statistically significant in the case of *Regents* and *Regents+Monarch* tables. FRETS is thus performing better than a highly structured model while making use of a much simpler data formalism.

Our models are able to effectively generalize. With Monarch tables, the Lucene baseline is little better than random (25%). But with the same knowledge base data, FRETS is competitive and sometimes scores higher than the best Lucene or MLN models (although this difference is statistically insignificant). These results indicate that our models are able to effectively capture both content and structure, reasoning approximately (and effectively) when the knowledge base may not even contain the relevant information to answer a question. The Monarch tables themselves seem to add little value, since results for Regents tables by themselves are just as good or better than Regents+Monarch tables. This is not a problem with FRETS, since the same phenomenon is witnessed with the Lucene baseline. It is noteworthy, however, that our models do not suffer from the addition of more tables, showing that our search procedure over table cells is robust.

Model	REG	MON	ESSQ
FRETS	59.1	**52.4**	**54.9**
w/o tab features	59.1	47.6	52.8
w/o row features	49.0	40.4	44.3
w/o col features	59.9	47.2	53.1
w/o cell features	25.7	25.0	24.9
w/o $\Diamond$ features	**62.2**	47.5	53.3

Table 6: Ablation study on FRETS, removing groups of features based on level of granularity. $\Diamond$ refers to the soft matching features from Table 4. Best results on a dataset are highlighted in bold.

Finally, dropping some features in the compact model doesn't always hurt performance, in comparison with the full model. This indicates that potentially higher scores are possible by a principled and detailed feature selection process. In these experiments the difference between the two FRETS models on equivalent data is statistically insignificant.

6.1 Ablation Study

To evaluate the contribution of different features we perform an ablation study, by individually removing groups of features from the full FRETS

model, and re-training. Evaluation of these partial models is given in Table 6. In this experiment we use all tables as knowledge base data.

Judging by relative score differential, cell features are by far the most important group, followed by row features. In both cases the drops in score are statistically significant. Intuitively, these results make sense, since row features are crucial in alignment to questions, while cell features capture the most fine-grained properties. It is less clear which among the other three feature groups is dominant, since the differences are not statistically significant. It is possible that cell features replicate information of other feature groups. For example, the cell answer-type entailment feature indirectly captures the same information as the header answer-type match feature (a column feature). Similarly, salience captures weighted statistics that are roughly equivalent to the coarse-grained table features. Interestingly, the success of these fine-grained features would explain our improvements over the Lucene baseline in Table 5, which is incapable of such fine-grained search.

7 Conclusions

We have presented tables as knowledge bases for question answering. We explored a connected framework in which tables are first used to guide the creation of MCQ data with alignment information to table elements, then jointly with this data are used in a feature-driven model to answer unseen MCQs. A central research question of this paper was the trade-off between the degree of structure in a knowledge base and its ability to be harvested or reasoned with. On three benchmark evaluation sets our consistently and significantly better scores over an unstructured and a highly-structured baseline strongly suggest that tables can be considered a balanced compromise in this trade-off. We also showed that our model is able to generalize from content to structure, thus reasoning about questions whose answer may not even be contained in the knowledge base.

We are releasing our dataset of more than 9000 MCQs and their alignment information, to the research community. We believe it offers interesting challenges that go beyond the scope of this paper – such as question parsing, or textual entailment – and are exciting avenues for future research.

Acknowledgement

We'd like to thank AI2 for funding this research and the creation of our MCQ dataset. The first and third authors of this paper were also supported in part by the following grants: NSF grant IIS-1143703, NSF award IIS-1147810, DARPA grant FA87501220342. Thanks also go to the anonymous reviewers, whose valuable comments helped to improve the quality of the paper.

References

Bahadir Ismail Aydin, Yavuz Selim Yilmaz, Yaliang Li, Qi Li, Jing Gao, and Murat Demirbas. 2014. Crowdsourcing for multiple-choice question answering. In *Twenty-Sixth IAAI Conference*.

Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, 1(1):538–549.

Peter Clark, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Turney, and Daniel Khashabi. 2016. Combining retrieval, statistics, and inference to answer elementary science questions. *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI-2016*.

Peter Clark. 2015. Elementary school science and math tests as a driver for ai: Take the aristo challenge. *Proceedings of IAAI, 2015*.

Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S Weld. 2008. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of NAACL-HLT*, pages 758–764, Atlanta, Georgia, June. Association for Computational Linguistics.

Heng Ji and Ralph Grishman. 2011. Knowledge base population: Successful approaches and challenges. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1148–1158. Association for Computational Linguistics.

Tushar Khot, Niranjan Balasubramanian, Eric Gribkoff, Ashish Sabharwal, Peter Clark, and Oren Etzioni. 2015. Exploring markov logic networks for question answering. *Proceedings of EMNLP, 2015*.

Dekang Lin and Patrick Pantel. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(04):343–360.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 87–93. Association for Computational Linguistics.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*, page 693. Association for Computational Linguistics.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*.

Rakesh Pimplikar and Sunita Sarawagi. 2012. Answering table queries on the web using column keywords. *Proceedings of the VLDB Endowment*, 5(10):908–919.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *EMNLP-CoNLL*, pages 12–21.

Rohini Srihari and Wei Li. 1999. Information extraction supported question answering. Technical report, DTIC Document.

Huan Sun, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web (to appear)*.

Zareen Syed, Tim Finin, Varish Mulwad, and Anupam Joshi. 2010. Exploiting a web of semantic data for interpreting tables. In *Proceedings of the Second Web Science Conference*.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM.

Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. 2011. Recovering semantics of tables on the web. *Proc. VLDB Endow.*, 4(9):528–538, June.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015a. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1301–1310. ACM.

Pengcheng Yin, Zhengdong Lu, Hang Li, and Ben Kao. 2015b. Neural enquirer: Learning to query tables. *arXiv preprint arXiv:1512.00965*.

Neural Summarization by Extracting Sentences and Words

Jianpeng Cheng **Mirella Lapata**
ILCC, School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
jianpeng.cheng@ed.ac.uk mlap@inf.ed.ac.uk

Abstract

Traditional approaches to extractive summarization rely heavily on human-engineered features. In this work we propose a data-driven approach based on neural networks and continuous sentence features. We develop a general framework for single-document summarization composed of a hierarchical document encoder and an attention-based extractor. This architecture allows us to develop different classes of summarization models which can extract sentences or words. We train our models on large scale corpora containing hundreds of thousands of document-summary pairs[1]. Experimental results on two summarization datasets demonstrate that our models obtain results comparable to the state of the art without any access to linguistic annotation.

1 Introduction

The need to access and digest large amounts of textual data has provided strong impetus to develop automatic summarization systems aiming to create shorter versions of one or more documents, whilst preserving their information content. Much effort in automatic summarization has been devoted to sentence extraction, where a summary is created by identifying and subsequently concatenating the most salient text units in a document.

Most extractive methods to date identify sentences based on human-engineered features. These include surface features such as sentence position and length (Radev et al., 2004), the words in the title, the presence of proper nouns, content features such as word frequency (Nenkova et al., 2006), and event features such as action nouns (Filatova and Hatzivassiloglou, 2004). Sentences are typically assigned a score indicating the strength of presence of these features. Several methods have been used in order to select the summary sentences ranging from binary classifiers (Kupiec et al., 1995), to hidden Markov models (Conroy and O'Leary, 2001), graph-based algorithms (Erkan and Radev, 2004; Mihalcea, 2005), and integer linear programming (Woodsend and Lapata, 2010).

In this work we propose a data-driven approach to summarization based on neural networks and continuous sentence features. There has been a surge of interest recently in repurposing sequence transduction neural network architectures for NLP tasks such as machine translation (Sutskever et al., 2014), question answering (Hermann et al., 2015), and sentence compression (Rush et al., 2015). Central to these approaches is an encoder-decoder architecture modeled by recurrent neural networks. The encoder reads the source sequence into a list of continuous-space representations from which the decoder generates the target sequence. An attention mechanism (Bahdanau et al., 2015) is often used to locate the region of focus during decoding.

We develop a general framework for single-document summarization which can be used to extract sentences or words. Our model includes a neural network-based hierarchical document reader or encoder and an attention-based content extractor. The role of the reader is to derive the meaning representation of a document based on its sentences and their constituent words. Our models adopt a variant of neural attention to extract sentences or words. Contrary to previous work where attention is an *intermediate* step used to blend hidden units of an encoder to a vector propagating additional information to the decoder, our model applies attention *directly* to select sentences or words of the input document as the output summary. Similar neural attention architectures have been previously used for geometry reasoning (Vinyals et al., 2015), under the name *Pointer Networks*.

[1]Resources are available for download at http://homepages.inf.ed.ac.uk/s1537177/resources.html

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 484–494,
Berlin, Germany, August 7-12, 2016.

One stumbling block to applying neural network models to extractive summarization is the lack of training data, i.e., documents with sentences (and words) labeled as summary-worthy. Inspired by previous work on summarization (Woodsend and Lapata, 2010; Svore et al., 2007) and reading comprehension (Hermann et al., 2015) we retrieve hundreds of thousands of news articles and corresponding highlights from the DailyMail website. Highlights usually appear as bullet points giving a brief overview of the information contained in the article (see Figure 1 for an example). Using a number of transformation and scoring algorithms, we are able to match highlights to document content and construct two large scale training datasets, one for sentence extraction and the other for word extraction. Previous approaches have used small scale training data in the range of a few hundred examples.

Our work touches on several strands of research within summarization and neural sequence modeling. The idea of creating a summary by extracting words from the source document was pioneered in Banko et al. (2000) who view summarization as a problem analogous to statistical machine translation and generate headlines using statistical models for selecting and ordering the summary words. Our word-based model is similar in spirit, however, it operates over continuous representations, produces multi-sentence output, and jointly selects summary words and organizes them into sentences. A few recent studies (Kobayashi et al., 2015; Yogatama et al., 2015) perform sentence extraction based on pre-trained sentence embeddings following an unsupervised optimization paradigm. Our work also uses continuous representations to express the meaning of sentences and documents, but importantly employs neural networks more directly to perform the actual summarization task.

Rush et al. (2015) propose a neural attention model for abstractive sentence compression which is trained on pairs of headlines and first sentences in an article. In contrast, our model summarizes documents rather than individual sentences, producing multi-sentential discourse. A major architectural difference is that our decoder selects output symbols from the document of interest rather than the entire vocabulary. This effectively helps us sidestep the difficulty of searching for the next output symbol under *a large vocabulary*, with *low-frequency words* and *named entities* whose representations can be challenging to learn. Gu et al. (2016) and Gulcehre et al. (2016) propose a similar "copy" mechanism in sentence compression and other tasks; their model can accommodate both generation and extraction by selecting which sub-sequences in the input sequence to copy in the output.

We evaluate our models both automatically (in terms of ROUGE) and by humans on two datasets: the benchmark DUC 2002 document summarization corpus and our own DailyMail news highlights corpus. Experimental results show that our summarizers achieve performance comparable to state-of-the-art systems employing hand-engineered features and sophisticated linguistic constraints.

2 Problem Formulation

In this section we formally define the summarization tasks considered in this paper. Given a document D consisting of a sequence of sentences $\{s_1, \cdots, s_m\}$ and a word set $\{w_1, \cdots, w_n\}$, we are interested in obtaining summaries at two levels of granularity, namely sentences and words.

Sentence extraction aims to create a summary from D by selecting a subset of j sentences (where $j < m$). We do this by scoring each sentence within D and predicting a label $y_L \in \{0, 1\}$ indicating whether the sentence should be included in the summary. As we apply supervised training, the objective is to maximize the likelihood of all sentence labels $\mathbf{y}_L = (y_L^1, \cdots, y_L^m)$ given the input document D and model parameters θ:

$$\log p(\mathbf{y}_L|D;\theta) = \sum_{i=1}^{m} \log p(y_L^i|D;\theta) \qquad (1)$$

Although extractive methods yield naturally grammatical summaries and require relatively little linguistic analysis, the selected sentences make for long summaries containing much redundant information. For this reason, we also develop a model based on **word extraction** which seeks to find a subset of words[2] in D and their optimal ordering so as to form a summary $\mathbf{y}_s = (w'_1, \cdots, w'_k), w'_i \in D$. Compared to sentence extraction which is a sequence labeling problem, this task occupies the middle ground between full abstractive summarization which can exhibit a wide range of rewrite operations and extractive

[2]The vocabulary can also be extended to include a small set of commonly-used (high-frequency) words.

AFL star blames vomiting cat for speeding

Adelaide Crows defender Daniel Talia has kept his driving license, telling a court he was speeding 36km over the limit because he was distracted by his sick cat.

The 22-year-old AFL star, who drove 96km/h in a 60km/h road works zone on the South Eastern expressway in February, said he didn't see the reduced speed sign because he was so distracted by his cat vomiting violently in the back seat of his car.

In the Adelaide magistrates court on Wednesday, Magistrate Bob Harrap fined Talia $824 for exceeding the speed limit by more than 30km/h.

He lost four demerit points, instead of seven, because of his significant training commitments.

- *Adelaide Crows defender Daniel Talia admits to speeding but says he didn't see road signs because his cat was vomiting in his car.*
- *22-year-old Talia was fined $824 and four demerit points, instead of seven, because of his 'significant' training commitments.*

Figure 1: DailyMail news article with highlights. Underlined sentences bear label 1, and 0 otherwise.

summarization which exhibits none. We formulate word extraction as a language generation task with an output vocabulary restricted to the original document. In our supervised setting, the training goal is to maximize the likelihood of the generated sentences, which can be further decomposed by enforcing conditional dependencies among their constituent words:

$$\log p(\mathbf{y}_s|D;\theta)=\sum_{i=1}^{k}\log p(w'_i|D,w'_1,\cdots,w'_{i-1};\theta) \quad (2)$$

In the following section, we discuss the data elicitation methods which allow us to train neural networks based on the above defined objectives.

3 Training Data for Summarization

Data-driven neural summarization models require a large training corpus of documents with labels indicating which sentences (or words) should be in the summary. Until now such corpora have been limited to hundreds of examples (e.g., the DUC 2002 single document summarization corpus) and thus used mostly for testing (Woodsend and Lapata, 2010). To overcome the paucity of annotated data for training, we adopt a methodology similar to Hermann et al. (2015) and create two large-scale datasets, one for sentence extraction and another one for word extraction.

In a nutshell, we retrieved[3] hundreds of thousands of news articles and their corresponding highlights from DailyMail (see Figure 1 for an example). The highlights (created by news editors) are genuinely abstractive summaries and therefore not readily suited to supervised training. To create the training data for **sentence extraction**, we reverse approximated the gold standard label of each document sentence given the summary based on their semantic correspondence (Woodsend and Lapata, 2010). Specifically, we designed a rule-based system that determines whether a document sentence matches a highlight and should be labeled with 1 (must be in the summary), and 0 otherwise. The rules take into account the position of the sentence in the document, the unigram and bigram overlap between document sentences and highlights, the number of entities appearing in the highlight and in the document sentence. We adjusted the weights of the rules on 9,000 documents with manual sentence labels created by Woodsend and Lapata (2010). The method obtained an accuracy of 85% when evaluated on a held-out set of 216 documents coming from the same dataset and was subsequently used to label 200K documents. Approximately 30% of the sentences in each document were deemed summary-worthy.

For the creation of the **word extraction** dataset, we examine the lexical overlap between the highlights and the news article. In cases where all highlight words (after stemming) come from the original document, the document-highlight pair constitutes a valid training example and is added to the word extraction dataset. For out-of-vocabulary (OOV) words, we try to find a semantically equivalent replacement present in the news article. Specifically, we check if a neighbor, represented

[3]The script for constructing our datasets is modified from the one released in Hermann et al. (2015).

by pre-trained[4] embeddings, is in the original document and therefore constitutes a valid substitution. If we cannot find any substitutes, we discard the document-highlight pair. Following this procedure, we obtained a word extraction dataset containing 170K articles, again from the DailyMail.

4 Neural Summarization Model

The key components of our summarization model include a neural network-based hierarchical document reader and an attention-based hierarchical content extractor. The hierarchical nature of our model reflects the intuition that documents are generated compositionally from words, sentences, paragraphs, or even larger units. We therefore employ a representation framework which reflects the same architecture, with global information being discovered and local information being preserved. Such a representation yields minimum information loss and is flexible allowing us to apply neural attention for selecting salient sentences and words within a larger context. In the following, we first describe the document reader, and then present the details of our sentence and word extractors.

4.1 Document Reader

The role of the reader is to derive the meaning representation of the document from its constituent sentences, each of which is treated as a sequence of words. We first obtain representation vectors at the sentence level using a single-layer convolutional neural network (CNN) with a max-over-time pooling operation (Kalchbrenner and Blunsom, 2013; Zhang and Lapata, 2014; Kim et al., 2016). Next, we build representations for documents using a standard recurrent neural network (RNN) that recursively composes sentences. The CNN operates at the word level, leading to the acquisition of sentence-level representations that are then used as inputs to the RNN that acquires document-level representations, in a hierarchical fashion. We describe these two sub-components of the text reader below.

Convolutional Sentence Encoder We opted for a convolutional neural network model for representing sentences for two reasons. Firstly, single-layer CNNs can be trained effectively (without any long-term dependencies in the model) and secondly, they have been successfully used for sentence-level classification tasks such as sentiment analysis (Kim, 2014). Let d denote the dimension of word embeddings, and s a document sentence consisting of a sequence of n words $(w_1, \cdots, w_n)$ which can be represented by a dense column matrix $\mathbf{W} \in \mathbb{R}^{n \times d}$. We apply a temporal narrow convolution between $\mathbf{W}$ and a kernel $\mathbf{K} \in \mathbb{R}^{c \times d}$ of width c as follows:

$$\mathbf{f}^i_j = \tanh(\mathbf{W}_{j:j+c-1} \otimes \mathbf{K} + b) \tag{3}$$

where $\otimes$ equates to the Hadamard Product followed by a sum over all elements. $\mathbf{f}^i_j$ denotes the j-th element of the i-th feature map $\mathbf{f}^i$ and b is the bias. We perform max pooling over time to obtain a *single* feature (the ith feature) representing the sentence under the kernel $\mathbf{K}$ with width c:

$$\mathbf{s}_{i,\mathbf{K}} = \max_j \mathbf{f}^i_j \tag{4}$$

In practice, we use multiple feature maps to compute a list of features that match the dimensionality of a sentence under each kernel width. In addition, we apply multiple kernels with different widths to obtain a set of different sentence vectors. Finally, we sum these sentence vectors to obtain the final sentence representation. The CNN model is schematically illustrated in Figure 2 (bottom). In the example, the sentence embeddings have six dimensions, so six feature maps are used under each kernel width. The blue feature maps have width two and the red feature maps have width three. The sentence embeddings obtained under each kernel width are summed to get the final sentence representation (denoted by green).

Recurrent Document Encoder At the document level, a recurrent neural network composes a sequence of sentence vectors into a document vector. Note that this is a somewhat simplistic attempt at capturing document organization at the level of sentence to sentence transitions. One might view the hidden states of the recurrent neural network as a list of partial representations with each focusing mostly on the corresponding input sentence given the previous context. These representations altogether constitute the document representation, which captures local and global sentential information with minimum compression.

The RNN we used has a Long Short-Term Memory (LSTM) activation unit for ameliorating the vanishing gradient problem when training long sequences (Hochreiter and Schmidhuber,

[4]We used the Python Gensim library and the 300-dimensional *GoogleNews* vectors.

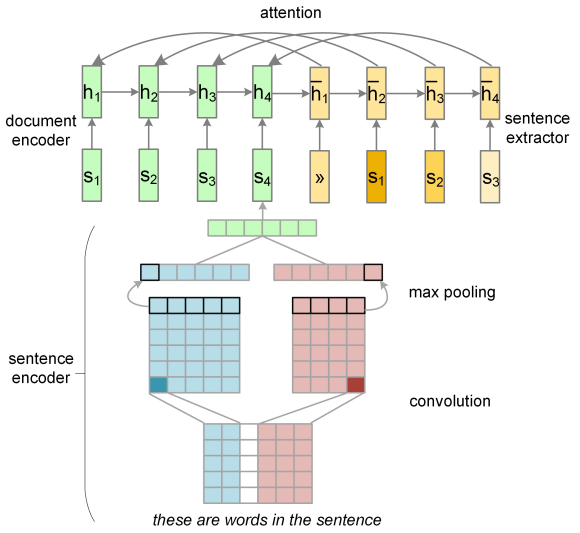


Figure 2: A recurrent convolutional document reader with a neural sentence extractor.

1997). Given a document $d = (s_1, \cdots, s_m)$, the hidden state at time step t, denoted by $\mathbf{h_t}$, is updated as:

$$\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \mathbf{W} \cdot \begin{bmatrix} \mathbf{h}_{t-1} \\ \mathbf{s}_t \end{bmatrix} \quad (5)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \quad (6)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (7)$$

where $\mathbf{W}$ is a learnable weight matrix. Next, we discuss a special attention mechanism for extracting sentences and words given the recurrent document encoder just described, starting from the sentence extractor.

4.2 Sentence Extractor

In the standard neural sequence-to-sequence modeling paradigm (Bahdanau et al., 2015), an attention mechanism is used as an intermediate step to decide which input region to focus on in order to generate the next output. In contrast, our sentence extractor applies attention to directly extract salient sentences after reading them.

The extractor is another recurrent neural network that labels sentences sequentially, taking into account not only whether they are individually relevant but also mutually redundant. The complete architecture for the document encoder and the sentence extractor is shown in Figure 2. As can be seen, the next labeling decision is made

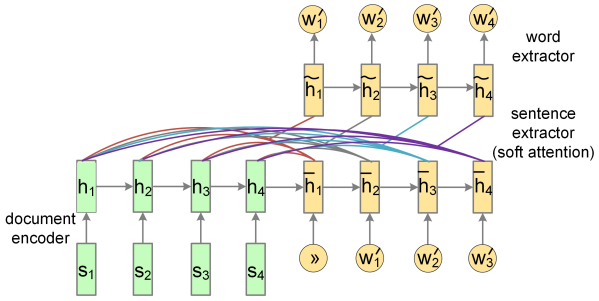


Figure 3: Neural attention mechanism for word extraction.

with both the encoded document and the previously labeled sentences in mind. Given encoder hidden states $(h_1, \cdots, h_m)$ and extractor hidden states $(\bar{h}_1, \cdots, \bar{h}_m)$ at time step t, the decoder attends the t-th sentence by relating its current decoding state to the corresponding encoding state:

$$\bar{\mathbf{h}}_t = \text{LSTM}(p_{t-1}\mathbf{s}_{t-1}, \bar{\mathbf{h}}_{t-1}) \quad (8)$$

$$p(y_L(t) = 1|D) = \sigma(\text{MLP}(\bar{\mathbf{h}}_t : \mathbf{h}_t)) \quad (9)$$

where MLP is a multi-layer neural network with as input the concatenation of $\bar{\mathbf{h}}_t$ and $\mathbf{h}_t$. p_{t-1} represents the degree to which the extractor believes the previous sentence should be extracted and memorized (p_{t-1}=1 if the system is certain; 0 otherwise).

In practice, there is a discrepancy between training and testing such a model. During training we know the true label p_{t-1} of the previous sentence, whereas at test time p_{t-1} is unknown and has to be predicted by the model. The discrepancy can lead to quickly accumulating prediction errors, especially when mistakes are made early in the sequence labeling process. To mitigate this, we adopt a curriculum learning strategy (Bengio et al., 2015): at the beginning of training when p_{t-1} cannot be predicted accurately, we set it to the true label of the previous sentence; as training goes on, we gradually shift its value to the predicted label $p(y_L(t-1) = 1|d)$.

4.3 Word Extractor

Compared to sentence extraction which is a purely sequence labeling task, word extraction is closer to a generation task where relevant content must be selected and then rendered fluently and grammatically. A small extension to the structure of the sequential labeling model makes it suitable for generation: instead of predicting a label for the next sentence at each time step, the model directly outputs the next word in the summary. The

model uses a *hierarchical* attention architecture: at time step t, the decoder softly[5] attends each document sentence and subsequently attends each word in the document and computes the probability of the next word to be included in the summary $p(w'_t = w_i|d, w'_1, \cdots, w'_{t-1})$ with a softmax classifier:

$$\bar{\mathbf{h}}_t = \text{LSTM}(\mathbf{w}'_{t-1}, \bar{\mathbf{h}}_{t-1})^6 \quad (10)$$

$$a^t_j = \mathbf{z}^{\text{T}} \tanh(\mathbf{W}_e \bar{\mathbf{h}}_t + \mathbf{W}_r \mathbf{h}_j), h_j \in D \quad (11)$$

$$b^t_j = \text{softmax}(a^t_j) \quad (12)$$

$$\tilde{\mathbf{h}}_t = \sum_{j=1}^{m} b^t_j \mathbf{h}_j \quad (13)$$

$$u^t_i = \mathbf{v}^{\text{T}} \tanh(\mathbf{W}_{e'} \tilde{\mathbf{h}}_t + \mathbf{W}_{r'} \mathbf{w}_i), w_i \in D \quad (14)$$

$$p(w'_t = w_i|D, w'_1, \cdots, w'_{t-1}) = \text{softmax}(u^t_i) \quad (15)$$

In the above equations, $\mathbf{w}_i$ corresponds to the vector of the i-th word in the input document, whereas $\mathbf{z}$, $\mathbf{W}_e$, $\mathbf{W}_r$, $\mathbf{v}$, $\mathbf{W}_{e'}$, and $\mathbf{W}_{r'}$ are model weights. The model architecture is shown in Figure 3.

The word extractor can be viewed as a conditional language model with a vocabulary constraint. In practice, it is not powerful enough to enforce grammaticality due to the lexical diversity and sparsity of the document highlights. A possible enhancement would be to pair the extractor with a neural language model, which can be pretrained on a large amount of unlabeled documents and then jointly tuned with the extractor during decoding (Gulcehre et al., 2015). A simpler alternative which we adopt is to use n-gram features collected from the document to rerank candidate summaries obtained via beam decoding. We incorporate the features in a log-linear reranker whose feature weights are optimized with minimum error rate training (Och, 2003).

5 Experimental Setup

In this section we present our experimental setup for assessing the performance of our summarization models. We discuss the datasets used for training and evaluation, give implementation details, briefly introduce comparison models, and explain how system output was evaluated.

Datasets We trained our sentence- and word-based summarization models on the two datasets created from DailyMail news. Each dataset was split into approximately 90% for training, 5% for validation, and 5% for testing. We evaluated the models on the DUC-2002 single document summarization task. In total, there are 567 documents belonging to 59 different clusters of various news topics. Each document is associated with two versions of 100-word[7] manual summaries produced by human annotators. We also evaluated our models on 500 articles from the DailyMail test set (with the human authored highlights as goldstandard). We sampled article-highlight pairs so that the highlights include a minimum of 3 sentences. The average byte count for each document is 278.

Implementation Details We trained our models with Adam (Kingma and Ba, 2014) with initial learning rate 0.001. The two momentum parameters were set to 0.99 and 0.999 respectively. We performed mini-batch training with a batch size of 20 documents. All input documents were padded to the same length with an additional mask variable storing the real length for each document. The size of word, sentence, and document embeddings were set to 150, 300, and 750, respectively. For the convolutional sentence model, we followed Kim et al. (2016)[8] and used a list of kernel sizes $\{1, 2, 3, 4, 5, 6, 7\}$. For the recurrent document model and the sentence extractor, we used as regularization dropout with probability 0.5 on the LSTM input-to-hidden layers and the scoring layer. The depth of each LSTM module was 1. All LSTM parameters were randomly initialized over a uniform distribution within [-0.05, 0.05]. The word vectors were initialized with 150 dimensional pre-trained embeddings.[9]

Proper nouns pose a problem for embedding-based approaches, especially when these are rare

[5] A simpler model would use hard attention to select a sentence first and then a few words from it as a summary, but this would render the system non-differentiable for training. Although hard attention can be trained with the REINFORCE algorithm (Williams, 1992), it requires sampling of discrete actions and could lead to high variance.

[6] We empirically found that feeding the previous sentence-level attention vector as additional input to the LSTM would lead to small performance improvements. This is not shown in the equation.

[7] According to the DUC2002 guidelines `http://www-nlpir.nist.gov/projects/duc/guidelines/2002.html`, the generated summary should be within 100 words.

[8] The CNN-LSTM architecture is publicly available at `https://github.com/yoonkim/lstm-char-cnn`.

[9] We used the *word2vec* (Mikolov et al., 2013) skip-gram model with context window size 6, negative sampling size 10 and hierarchical softmax 1. The model was trained on the *Google 1-billion* word benchmark (Chelba et al., 2014).

or unknown (e.g., at test time). Rush et al. (2015) address this issue by adding a new set of features and a log-linear model component to their system. As our model enjoys the advantage of generation by extraction, we can force the model to inspect the context surrounding an entity and its relative position in the sentence in order to discover extractive patterns, placing less emphasis on the meaning representation of the entity itself. Specifically, we perform named entity recognition with the package provided by Hermann et al. (2015) and maintain a set of randomly initialized entity embeddings. During training, the index of the entities is permuted to introduce some noise but also robustness in the data. A similar data augmentation approach has been used for reading comprehension (Hermann et al., 2015).

A common problem with extractive methods based on sentence labeling is that there is no constraint on the number of sentences being selected at test time. We address this by reranking the positively labeled sentences with the probability scores obtained from the softmax layer (rather than the label itself). In other words, we are more interested in is the relative ranking of each sentence rather than their exact scores. This suggests that an alternative to training the network would be to employ a ranking-based objective or a *learning to rank* algorithm. However, we leave this to future work. We use the three sentences with the highest scores as the summary (also subject to the word or byte limit of the evaluation protocol).

Another issue relates to the word extraction model which is challenging to batch since each document possesses a distinct vocabulary. We sidestep this during training by performing negative sampling (Mikolov et al., 2013) which trims the vocabulary of different documents to the same length. At each decoding step the model is trained to differentiate the true target word from 20 noise samples. At test time we still loop through the words in the input document (and a stop-word list) to decide which word to output next.

System Comparisons We compared the output of our models to various summarization methods. These included the standard baseline of simply selecting the "leading" three sentences from each document as the summary. We also built a sentence extraction baseline classifier using logistic regression and human engineered features. The classifier was trained on the same datasets as our neural network models with the following features: sentence length, sentence position, number of entities in the sentence, sentence-to-sentence cohesion, and sentence-to-document relevance. Sentence-to-sentence cohesion was computed by calculating for every document sentence its embedding similarity with every other sentence in the same document. The feature was the normalized sum of these similarity scores. Sentence embeddings were obtained by averaging the constituent word embeddings. Sentence-to-document relevance was computed similarly. We calculated for each sentence its embedding similarity with the document (represented as bag-of-words), and normalized the score. The word embeddings used in this baseline are the same as the pre-trained ones used for our neural models.

In addition, we included a neural abstractive summarization baseline. This system has a similar architecture to our word extraction model except that it uses an open vocabulary during decoding. It can also be viewed as a hierarchical document-level extension of the abstractive sentence summarizer proposed by Rush et al. (2015). We trained this model with negative sampling to avoid the excessive computation of the normalization constant.

Finally, we compared our models to three previously published systems which have shown competitive performance on the DUC2002 single document summarization task. The first approach is the phrase-based extraction model of Woodsend and Lapata (2010). Their system learns to produce highlights from parsed input (phrase structure trees and dependency graphs); it selects salient phrases and recombines them subject to length, coverage, and grammar constraints enforced via integer linear programming (ILP). Like ours, this model is trained on document-highlight pairs, and produces telegraphic-style bullet points rather than full-blown summaries. The other two systems, TGRAPH (Parveen et al., 2015) and URANK (Wan, 2010), produce more typical summaries and represent the state of the art. TGRAPH is a graph-based sentence extraction model, where the graph is constructed from topic models and the optimization is performed by constrained ILP. URANK adopts a unified ranking system for both single- and multi-document summarization.

Evaluation We evaluated the quality of the summaries automatically using ROUGE (Lin and Hovy, 2003). We report unigram and bigram over-

DUC 2002	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	43.6	21.0	40.2
LREG	43.8	20.7	40.3
ILP	45.4	21.3	42.8
NN-ABS	15.8	5.2	13.8
TGRAPH	48.1	**24.3**	—
URANK	**48.5**	21.5	—
NN-SE	47.4	23.0	**43.5**
NN-WE	27.0	7.9	22.8

DailyMail	ROUGE-1	ROUGE-2	ROUGE-L
LEAD	20.4	7.7	11.4
LREG	18.5	6.9	10.2
NN-ABS	7.8	1.7	7.1
NN-SE	**21.2**	**8.3**	**12.0**
NN-WE	15.7	6.4	9.8

Table 1: ROUGE evaluation (%) on the DUC-2002 and 500 samples from the DailyMail.

lap (ROUGE-1,2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

In addition, we evaluated the generated summaries by eliciting human judgments for 20 randomly sampled DUC 2002 test documents. Participants were presented with a news article and summaries generated by a list of systems. These include two neural network systems (sentence- and word-based extraction), the neural abstractive system described earlier, the lead baseline, the phrase-based ILP model[10] of Woodsend and Lapata (2010), and the human authored summary. Subjects were asked to rank the summaries from best to worst (with ties allowed) in order of informativeness (does the summary capture important information in the article?) and fluency (is the summary written in well-formed English?). We elicited human judgments using Amazon's Mechanical Turk crowdsourcing platform. Participants (self-reported native English speakers) saw 2 random articles per session. We collected 5 responses per document.

6 Results

Table 1 (top half) summarizes our results on the DUC 2002 test dataset using ROUGE. NN-SE represents our neural sentence extraction model, NN-WE our word extraction model, and NN-ABS the neural abstractive baseline. The table also includes results for the LEAD baseline, the logistic regression classifier (LREG), and three previously published systems (ILP, TGRAPH, and URANK).

[10]We are grateful to Kristian Woodsend for giving us access to the output of his system. Unfortunately, we do not have access to the output of TGRAPH or URANK for inclusion in the human evaluation.

Models	1^{st}	2^{nd}	3^{rd}	4^{th}	5^{th}	6^{th}	MeanR
LEAD	0.10	0.17	0.37	0.15	0.16	0.05	3.27
ILP	0.19	0.38	0.13	0.13	0.11	0.06	2.77
NN-SE	0.22	0.28	0.21	0.14	0.12	0.03	2.74
NN-WE	0.00	0.04	0.03	0.21	0.51	0.20	4.79
NN-ABS	0.00	0.01	0.05	0.16	0.23	0.54	5.24
Human	0.27	0.23	0.29	0.17	0.03	0.01	2.51

Table 2: Rankings (shown as proportions) and mean ranks given to systems by human participants (lower is better).

The NN-SE outperforms the LEAD and LREG baselines with a significant margin, while performing slightly better than the ILP model. This is an encouraging result since our model has only access to embedding features obtained from raw text. In comparison, LREG uses a set of manually selected features, while the ILP system takes advantage of syntactic information and extracts summaries subject to well-engineered linguistic constraints, which are not available to our models. Overall, our sentence extraction model achieves performance comparable to the state of the art without sophisticated constraint optimization (ILP, TGRAPH) or sentence ranking mechanisms (URANK). We visualize the sentence weights of the NN-SE model in the top half of Figure 4. As can be seen, the model is able to locate text portions which contribute most to the overall meaning of the document.

ROUGE scores for the word extraction model are less promising. This is somewhat expected given that ROUGE is *n*-gram based and not very well suited to measuring summaries which contain a significant amount of paraphrasing and may deviate from the reference even though they express similar meaning. However, a meaningful comparison can be carried out between NN-WE and NN-ABS which are similar in spirit. We observe that NN-WE consistently outperforms the purely abstractive model. As NN-WE generates summaries by picking words from the original document, decoding is easier for this model compared to NN-ABS which deals with an open vocabulary. The extraction-based generation approach is more robust for proper nouns and rare words, which pose a serious problem to open vocabulary mod-

sentence extraction:

a gang of at least three people poured gasoline on a car that stopped to fill up at *entity5* gas station early on Saturday morning and set the vehicle on fire

the driver of the car, who has not been identified, said he got into an argument with the suspects while he was pumping gas at a *entity13* in *entity14*

the group covered his white *entity16* in gasoline and lit it ablaze while there were two passengers inside

at least three people poured gasoline on a car and lit it on fire at a *entity14* gas station explosive situation

the passengers and the driver were not hurt during the incident but the car was completely ruined

the man's grandmother said the fire was lit after the suspects attempted to carjack her grandson, *entity33* reported

she said:' he said he was pumping gas and some guys came up and asked for the car

' they pulled out a gun and he took off running

' they took the gas tank and started spraying

' no one was injured during the fire , but the car 's entire front end was torched , according to *entity52*

the *entity53* is investigating the incident as an arson and the suspects remain at large

surveillance video of the incident is being used in the investigation

before the fire , which occurred at 12:15am on Saturday , the suspects tried to carjack the man hot case

the *entity53* is investigating the incident at the *entity67* station as an arson

word extraction:

gang poured gasoline in the car, *entity5* Saturday morning. the driver argued with the suspects. his grandmother said the fire was lit by the suspects attempted to carjack her grandson.

entities:

entity5:California *entity13*:76-Station *entity14*: South LA *entity16*:Dodge Charger *entity33*:ABC *entity52*:NBC *entity53*:LACFD *entity67*:LA76

Figure 4: Visualization of the summaries for a DailyMail article. The top half shows the relative attention weights given by the sentence extraction model. Darkness indicates sentence importance. The lower half shows the summary generated by the word extraction.

els. An example of the generated summaries for NN-WE is shown at the lower half of Figure 4.

Table 1 (lower half) also shows system results on the 500 DailyMail news articles (test set). In general, we observe similar trends to DUC 2002, with NN-SE performing the best in terms of all ROUGE metrics. Note that scores here are generally lower compared to DUC 2002. This is due to the fact that the gold standard summaries (aka highlights) tend to be more laconic and as a result involve a substantial amount of paraphrasing.

The results of our human evaluation study are shown in Table 2. Specifically, we show, proportionally, how often our participants ranked each system 1st, 2nd, and so on. Perhaps unsurprisingly, the human-written descriptions were considered best and ranked 1st 27% of the time, however closely followed by our NN-SE model which was ranked 1st 22% of the time. The ILP system was mostly ranked in 2nd place (38% of the time). The rest of the systems occupied lower ranks. We further converted the ranks to ratings on a scale of 1 to 6 (assigning ratings $6 \ldots 1$ to rank placements $1 \ldots 6$). This allowed us to perform Analysis of Variance (ANOVA) which revealed a reliable effect of system type. Specifically, post-hoc Tukey tests showed that NN-SE and ILP are significantly ($p < 0.01$) better than LEAD, NN-WE, and NN-ABS but do not differ significantly from each other or the human goldstandard.

7 Conclusions

In this work we presented a data-driven summarization framework based on an encoder-extractor architecture. We developed two classes of models based on sentence and word extraction. Our models can be trained on large scale datasets and learn informativeness features based on continuous representations without recourse to linguistic annotations. Two important ideas behind our work are the creation of hierarchical neural structures that reflect the nature of the summarization task and generation by extraction. The later effectively enables us to sidestep the difficulties of generating under a large vocabulary, essentially covering the entire dataset, with many low-frequency words and named entities.

Directions for future work are many and varied. One way to improve the word-based model would be to take structural information into account during generation, e.g., by combining it with a tree-based algorithm (Cohn and Lapata, 2009). It would also be interesting to apply the neural models presented here in a phrase-based setting similar to Lebret et al. (2015). A third direction would be to adopt an information theoretic perspective and devise a purely unsupervised approach that selects summary sentences and words so as to minimize information loss, a task possibly achievable with the dataset created in this work.

Acknowledgments

We would like to thank three anonymous reviewers and members of the ILCC at the School of Informatics for their valuable feedback. The support of the European Research Council under award number 681760 "Translating Multiple Modalities into Text" is gratefully acknowledged.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR 2015*, San Diego, California.

Michele Banko, Vibhu O. Mittal, and Michael J. Witbrock. 2000. Headline generation based on statistical translation. In *Proceedings of the 38th ACL*, pages 318–325, Hong Kong.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems 28*, pages 1171–1179. Curran Associates, Inc.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

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

Conroy and O'Leary. 2001. Text summarization via hidden Markov models. In *Proceedings of the 34th Annual ACL SIGIR*, pages 406–407, New Oleans, Louisiana.

Güneş Erkan and Dragomir R. Radev. 2004. Lexpagerank: Prestige in multi-document text summarization. In *Proceedings of the 2004 EMNLP*, pages 365–371, Barcelona, Spain.

Elena Filatova and Vasileios Hatzivassiloglou. 2004. Event-based extractive summarization. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 104–111, Barcelona, Spain.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th ACL*, Berlin, Germany. to appear.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Proceedings of the 54th ACL*, Berlin, Germany. to appear.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems 28*, pages 1684–1692. Curran Associates, Inc.

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

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

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of the 30th AAAI*, Phoenix, Arizon. to appear.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 EMNLP*, pages 1746–1751, Doha, Qatar.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hayato Kobayashi, Masaki Noguchi, and Taichi Yatsuka. 2015. Summarization based on embedding distributions. In *Proceedings of the 2015 EMNLP*, pages 1984–1989, Lisbon, Portugal.

Julian Kupiec, Jan O. Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *Proceedings of the 18th Annual International ACM SIGIR*, pages 68–73, Seattle, Washington.

Rémi Lebret, Pedro O Pinheiro, and Ronan Collobert. 2015. Phrase-based image captioning. In *Proceedings of the 32nd ICML*, Lille, France.

Chin-Yew Lin and Eduard H. Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of HLT NAACL*, pages 71–78, Edmonton, Canada.

Rada Mihalcea. 2005. Language independent extractive summarization. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 49–52, Ann Arbor, Michigan.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Ani Nenkova, Lucy Vanderwende, and Kathleen McKeown. 2006. A compositional context sensitive multi-document summarizer: exploring the factors that influence summarization. In *Proceedings of the 29th Annual ACM SIGIR*, pages 573–580, Washington, Seattle.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st ACL*, pages 160–167, Sapporo, Japan.

Daraksha Parveen, Hans-Martin Ramsl, and Michael Strube. 2015. Topical coherence for graph-based extractive summarization. In *Proceedings of the 2015 EMNLP*, pages 1949–1954, Lisbon, Portugal, September.

Dragomir Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Celebi, Stanko Dimitrov, Elliott Drabek, Ali Hakim, Wai Lam, Danyu Liu, et al. 2004. Mead-a platform for multidocument multilingual text summarization. Technical report, Columbia University Academic Commons.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 EMNLP*, pages 379–389, Lisbon, Portugal.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Krysta Svore, Lucy Vanderwende, and Christopher Burges. 2007. Enhancing single-document summarization by combining RankNet and third-party sources. In *Proceedings of the 2007 EMNLP-CoNLL*, pages 448–457, Prague, Czech Republic.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems 28*, pages 2674–2682. Curran Associates, Inc.

Xiaojun Wan. 2010. Towards a unified approach to simultaneous single-document and multi-document summarizations. In *Proceedings of the 23rd COLING*, pages 1137–1145.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Kristian Woodsend and Mirella Lapata. 2010. Automatic generation of story highlights. In *Proceedings of the 48th ACL*, pages 565–574, Uppsala, Sweden.

Dani Yogatama, Fei Liu, and Noah A. Smith. 2015. Extractive summarization by maximizing semantic volume. In *Proceedings of the 2015 EMNLP*, pages 1961–1966, Lisbon, Portugal.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of 2014 EMNLP*, pages 670–680, Doha, Qatar.

Neural Networks For Negation Scope Detection

Federico Fancellu and **Adam Lopez** and **Bonnie Webber**
School of Informatics
University of Edinburgh
11 Crichton Street, Edinburgh
f.fancellu[at]sms.ed.ac.uk, {alopez,bonnie}[at]inf.ed.ac.uk

Abstract

Automatic negation scope detection is a task that has been tackled using different classifiers and heuristics. Most systems are however 1) highly-engineered, 2) English-specific, and 3) only tested on the same genre they were trained on. We start by addressing 1) and 2) using a neural network architecture. Results obtained on data from the *SEM2012 shared task on negation scope detection show that even a simple feed-forward neural network using word-embedding features alone, performs on par with earlier classifiers, with a bi-directional LSTM outperforming all of them. We then address 3) by means of a specially-designed synthetic test set; in doing so, we explore the problem of detecting the negation scope more in depth and show that performance suffers from genre effects and differs with the type of negation considered.

1 Introduction

Amongst different extra-propositional aspects of meaning, negation is one that has received a lot of attention in the NLP community. Previous work have focused in particular on automatically detecting the *scope of negation*, that is, given a negative instance, to identify which tokens are affected by negation (§2). As shown in (1), only the first clause is negated and therefore we mark *he* and *the car*, along with the predicate *was driving* as inside the scope, while leaving the other tokens outside.

(1) He was **not** driving the car and she left to go home.

In the BioMedical domain there is a long line of research around the topic (e.g. Velldal et al. (2012) and Prabhakaran and Boguraev (2015)), given the importance of recognizing negation for information extraction from medical records. In more general domains, efforts have been more limited and most of the work centered around the *SEM2012 shared task on automatically detecting negation (§3), despite the recent interest (e.g. machine translation (Wetzel and Bond, 2012; Fancellu and Webber, 2014; Fancellu and Webber, 2015)).

The systems submitted for this shared task, although reaching good overall performance are highly feature-engineered, with some relying on heuristics based on English (Read et al. (2012)) or on tools that are available for a limited number of languages (e.g. Basile et al. (2012), Packard et al. (2014)), which do not make them easily portable across languages. Moreover, the performance of these systems was only assessed on data of the same genre (stories from Conan Doyle's *Sherlock Holmes*) but there was no attempt to test the approach on data of different genre.

Given these shortcomings, we investigate whether neural network based sequence-to-sequence models (§ 4) are a valid alternative. The first advantage of neural networks-based methods for NLP is that we could perform classification by means of unsupervised word-embeddings features only, under the assumption that they also encode structural information previous system had to explicitly represent as features. If this assumption holds, another advantage of continuous representations is that, by using a bilingual word-embedding space, we would be able to transfer the model cross-lingually, obviating the problem of the lack of annotated data in other languages.

The paper makes the following contributions:

1. *Comparable or better performance*: We show that neural networks perform on par with previously developed classifiers, with a bi-directional LSTM outperforming them

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 495–504,
Berlin, Germany, August 7-12, 2016.

when tested on data from the same genre.

2. *Better understanding of the problem*: We analyze in more detail the difficulty of detecting negation scope by testing on data of different genre and find that the performance of word-embedding features is comparable to that of more fine-grained syntactic features.

3. *Creation of additional resources*: We create a *synthetic* test set of negative sentences extracted from Simple English Wikipedia (§ 5) and annotated according to the guidelines released during the *SEM2012 shared task (Morante et al., 2011), that we hope will guide future work in the field.

2 The task

Before formalizing the task, we begin by giving some definitions. A *negative sentence n* is defined as a vector of words $\langle w_1, w_2...w_n \rangle$ containing one or more negation **cues**, where the latter can be a word (e.g. *not*), a morpheme (e.g. *im*-patient) or a multi-word expression (e.g. *by no means, no longer*) inherently expressing negation.

A word is a **scope token** if included in the scope of a negation cue. Following Blanco and Moldovan (2011), in the *SEM2012 shared task the negation **scope** is understood as part of a knowledge representation focused around a negated event along with its related semantic roles and adjuncts (or its head in the case of a nominal event). This is exemplified in (2) (from Blanco and Moldovan (2011)) where the scope includes both the negated event *eat* along the subject *the cow*, the object *grass* and the PP *with a fork*.

(2) The cow did **n't** eat grass with a fork.[1]

Each cue defines its own *negation instance*, here defined as a tuple *I*(*n*,c) where $c \in \{1,0\}^{|n|}$ is a vector of length *n* s.t. $c_i = 1$ if w_i is part of the cue and 0 otherwise. Given *I* the goal of automatic scope detection is to predict a vector $s \in \{O,I\}^{|n|}$ s.t. s_i = I (inside of the scope) if w_i is in the scope of the cue or O (outside) otherwise.

In (3) for instance, there are two cues, *not* and *no longer*, each one defining a separate negation instance, *I1(n,c1)* and *I2(n,c2)*, and each with its own scope, *s1* and *s2*. In both (3a) and (3b), *n* = [I, do, not, love, you, and, you, are, no, longer, invited]; in (3a), the vector *c1* is 1 only at index 3 (w_2='not'), while in (3b) *c2* is 1 at position 9, 10 (where w_9 w_{10} = 'no longer'); finally the vectors *s1* and *s2* are I only at the indices of the words underlined and O anywhere else.

(3) a. I do **not** love you and you are no longer invited

b. I do not love you and you are **no longer** invited

There are the two main challenges involved in detecting the scope of negation: 1) a sentence can contain multiple instances of negation, sometimes nested and 2) scope can be discontinuous. As for 1), the classifier must correctly classify each word as being inside or outside the scope and assign each word to the correct scope; in (4) for instance, there are two negation cues and therefore two scopes, one spanning the entire sentence (3a.) and the other the subordinate only (3b.), with the latter being nested in the former (given that, according to the guidelines, if we negate the event in the main, we also negate its cause).

(4) a. I did **not** drive to school because my wife was not feeling well .[2]

b. I did not drive to school because my wife was **not** feeling well .

In (5), the classifier should instead be able to capture the long range dependency between the subject and its negated predicate, while excluding the positive VP in the middle.

(5) Naomi went to visit her parents to give them a special gift for their anniversary but **never** came back .

In the original task, the performance of the classifier is assessed in terms of precision, recall and F_1 measure over the number of words correctly classified as part of the scope (*scope tokens*) and over the number of scopes predicted that exactly

[1] In the *SEM2012 shared task, negation is not considered as a downward monotone function and definite expressions are included in its scope.

[2] One might object that the scope only spans over the subordinate given that it is the part of the scope most likely to be interpreted as false (*It is not the case that I drove to school because my wife was not at home, but for other reasons*). In the *SEM2012 shared task however this is defined separately as the *focus* of negation and considered as part of the scope. One reason to distinguish the two is the high ambiguity of the focus: one can imagine for instance that if the speaker stresses the words *to school* this will be most likely considered the focus and the statement interpreted as *It is not the case that I drive to school because my wife was not feeling well* (but I drove to the hospital instead).

match the gold scopes (*exact scope match*). As for latter, recall is a measure of accuracy since we score how many scopes we fully predict (true positives) over the total number of scopes in our test set (true positives and false negatives); precision takes instead into consideration false positives, that is those negation instances that are predicted as having a scope but in reality don't have any. This is the case of the interjection *No* (e.g. '**No**, leave her alone') that never take scope.

3 Previous work

Table 1 summarizes the performance of systems previously developed to resolve the scope of negation in non-Biomedical texts.

In general, supervised classifiers perform better than rule-based systems, although it is a combination of hand-crafted heuristics and SVM rankers to achieve the best performance. Regardless of the approach used, the syntactic structure (either constituent or dependency-based) of the sentence is often used to detect the scope of negation. This is because the position of the cue in the tree along with the projection of its parent/governor are strong indicators of scope boundaries. Moreover, given that during training we basically learn which syntactic patterns the scope are likely to span, it is also possible to hypothesize that this system should scale well to other genre/domain, as long as we can have a parse for the sentence; this however was never confirmed empirically. Although informative, these systems suffers form three main shortcomings: 1) they are highly-engineered (as in the case of Read et al. (2012)) and syntactic features add up to other PoS, word and lemma n-gram features, 2) they rely on the parser producing a correct parse and 3) they are English specific.

Other systems (Basile et al., 2012; Packard et al., 2014) tried to traverse a semantic representation instead. Packard et al. (2014) achieves the best results so far, using hand-crafted heuristics to traverse the MRS (Minimal Recursion Semantics) structures of negative sentences. If the semantic parser cannot create a reliable representation for a sentence, the system 'backs-off' to the hybrid model of Read et al. (2012), which uses syntactic information instead. This system suffers however from the same shortcomings mentioned above, in particular, given that MRS representation can only be built for a small set of languages.

4 Scope detection using Neural Networks

In this paper, we experiment with two different neural networks architecture: a one hidden layer **feed-forward neural network** and a **bi-directional LSTM** (Long Short Term Memory, BiLSTM below) model. We chose to 'start simple' from a feed-forward network to investigate whether even a simple model can reach good performance using word-embedding features only. We then turned to a BiLSTM because a better fit for the task. BiLSTM are sequential models that operate both in forward and backwards fashion; the backward pass is especially important in the case of negation scope detection, given that a scope token can appear in a string before the cue and it is therefore important that we see the latter first to classify the former. We opted in this case for LSTM over RNN cells given that their inner composition is able to better retain useful information when backpropagating the error.[4]

Both networks take as input a single negative instance $I(n,c)$. We represent each word $w_i \in n$ as a d-dimensional word-embedding vector $\mathbf{x} \in \mathbb{R}^d$ (d=50). In order to encode information about the cue, each word is also represented by a *cue*-embedding vector $\mathbf{c} \in \mathbb{R}^d$ of the same dimensionality of $\mathbf{x}$. $\mathbf{c}$ can only take two representations, *cue*, if c_i=1, or *notcue* otherwise. We also define $\mathbf{E}_w^{v \times d}$ as the word-embedding matrix, where v is the vocabulary size, and $\mathbf{E}_c^{2 \times d}$ as the cue-embedding matrix.

In the case of a feed-forward neural network, the input for each word $w_i \in n$ is the concatenation of its representation with the ones of its neighboring words in a context window of length l. This is because feed-forward networks treat the input units as separate and information about how words are arranged as sequences must be explicitly encoded in the input. We define these concatenations $\mathbf{x}_{conc}$ and $\mathbf{c}_{conc}$ as $\mathbf{x}_{w_{i-l}}...\mathbf{x}_{w_{i-1}}$; $\mathbf{x}_{w_i}$; $\mathbf{x}_{w_{i+1}}...\mathbf{x}_{w_i+l}$ and $\mathbf{c}_{w_{i-l}}...\mathbf{c}_{w_{i-1}}$; $\mathbf{c}_{w_i}$; $\mathbf{c}_{w_{i+1}}...\mathbf{c}_{w_{i+l}}$ respectively. We chose the value of l after analyzing the negation scopes in the dev set. We found that although the furthest scope tokens are 23 and 31 positions away from the cue on the left and the right respectively, 95% of the scope tokens fall in a window of 9 tokens to the left and 15 to the right, these two values being the window sizes we con-

[4] For more details on LSTM and related mathematical formulations, we refer to reader to Hochreiter and Schmidhuber (1997)

			Method	Scope tokens[3]			Exact scope match		
				Prec.	*Rec.*	F_1	*Prec.*	*Rec.*	F_1
*SEM2012	Closed track	UiO1 (Read et al., 2012)	heuristics + SVM	81.99	88.81	85.26	87.43	61.45	72.17
		UiO2 (Lapponi et al., 2012)	CRF	86.03	81.55	83.73	85.71	62.65	72.39
		FBK (Chowdhury and Mahbub, 2012)	CRF	81.53	82.44	81.89	88.96	58.23	70.39
		UWashington (White, 2012)	CRF	83.26	83.77	83.51	82.72	63.45	71.81
		UMichigan (Abu-Jbara and Radev, 2012)	CRF	84.85	80.66	82.70	90.00	50.60	64.78
		UABCoRAL (Gyawali and Solorio, 2012)	SVM	85.37	68.86	76.23	79.04	53.01	63.46
	Open track	UiO2 (Lapponi et al., 2012)	CRF	82.25	82.16	82.20	85.71	62.65	72.39
		UGroningen (Basile et al., 2012)	rule-based	69.20	82.27	75.15	76.12	40.96	53.26
		UCM-1 (de Albornoz et al., 2012)	rule-based	85.37	68.53	76.03	82.86	46.59	59.64
		UCM-2 (Ballesteros et al., 2012)	rule-based	58.30	67.70	62.65	67.13	38.55	48.98
Packard et al. (2014)			heuristics + SVM	**86.1**	**90.4**	**88.2**	**98.8**	**65.5**	**78.7**

Table 1: Summary of previous work on automatic detection of negation scope.

sider for our input. The probability of a given input is then computed as follows:

$$\begin{aligned} \mathbf{h} &= \sigma(\mathbf{W}_x\mathbf{x}_{conc} + \mathbf{W}_c\mathbf{c}_{conc} + \mathbf{b}) \\ y &= g(\mathbf{W}_y\mathbf{h} + \mathbf{b}_y) \end{aligned} \tag{1}$$

where $\mathbf{W}$ and $\mathbf{b}$ the weight and biases matrices, $\mathbf{h}$ the hidden layer representation, σ the sigmoid activation function and g the softmax operation ($g(z_m) = e^{z_m} / \sum_k e^{z_k}$) to assign a probability to the input of belonging to either the inside (I) or outside (O) of the scope classes.

In the biLSTM, no concatenation is performed, given that the structure of the network is already sequential. The input to the network for each word w_i are the word-embedding vector $\mathbf{x}_{w_i}$ and the cue-embedding vector $\mathbf{c}_{w_i}$, where w_i constitutes a time step. The computation of the hidden layer at time t and the output can be represented as follows:

$$\begin{aligned} \mathbf{i}_t &= \sigma(\mathbf{W}_x^{(i)}\mathbf{x} + \mathbf{W}_c^{(i)}\mathbf{c} + \mathbf{W}_h^{(i)}\mathbf{h_{t-1}} + \mathbf{b}^{(i)}) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_x^{(f)}\mathbf{x} + \mathbf{W}_c^{(f)}\mathbf{c} + \mathbf{W}_h^{(f)}\mathbf{h_{t-1}} + \mathbf{b}^{(f)}) \\ \mathbf{o}_t &= \sigma(\mathbf{W}_x^{(o)}\mathbf{x} + \mathbf{W}_c^{(o)}\mathbf{c} + \mathbf{W}_h^{(o)}\mathbf{h_{t-1}} + \mathbf{b}^{(o)}) \\ \tilde{\mathbf{c}}_t &= tanh(\mathbf{W}_x^{(c)}\mathbf{x} + \mathbf{W}_c^{(c)}\mathbf{c} + \mathbf{W}_h^{(c)}\mathbf{h_{t-1}} + \mathbf{b}^{(c)}) \\ \mathbf{c}_t &= \mathbf{f}_t \cdot \tilde{\mathbf{c}}_{t-1} + \mathbf{i}_t \cdot \tilde{\mathbf{c}}_t \\ \mathbf{h}_{back/forw} &= \mathbf{o}_t \cdot tanh(\mathbf{c}_t) \\ y_t &= g(\mathbf{W}_y(\mathbf{h}_{back}; \mathbf{h}_{forw}) + \mathbf{b}_y) \end{aligned} \tag{2}$$

where the $\mathbf{W}$s are the weight matrices, $\mathbf{h}_{t-1}$ the hidden layer state a time t-1, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ the input, forget and the output gate at the time t and $\mathbf{h}_{back}$; $\mathbf{h}_{forw}$ the concatenation of the backward and forward hidden layers.

Finally, in both networks our training objective is to minimise, for each negative instance, the negative log likelihood *J(W,b)* of the correct predictions over gold labels:

$$\begin{aligned} J(W,b) = -\frac{1}{l}\sum_{i=1}^{l} & y^{(w_i)} \log h_\theta(x^{(w_i)}) \\ & + (1 - y^{(w_i)}) \log(1 - h_\theta(x^{(w_i)})) \end{aligned} \tag{3}$$

where l is the length of the sentence $n \in I$, $x^{(w_i)}$ the probability for the word w_i to belong to either the I or O class and $y^{(w_i)}$ its gold label.

An overview of both architectures is shown in Figure 1.

4.1 Experiments

Training, development and test set are a collection of stories from Conan Doyle's *Sherlock Holmes* annotated for cue and scope of negation and released in concomitance with the *SEM2012 shared task.[5] For each word, the correspondent lemma, POS tag and the constituent subtree it belongs to are also annotated. If a sentence contains multiple instances of negation, each is annotated separately.

Both training and testing is done on negative sentences only, i.e. those sentences with at least one cue annotated. Training and test size are of 848 and 235 sentences respectively. If a sentence contains multiple negation instances, we create as many copies as the number of instances. If the sentence contains a morphological cue (e.g. *im-patient*) we split it into affix (**im-**) and root (*patient*), and consider the former as cue and the latter as part of the scope.

Both neural network architectures are implemented using TensorFlow (Abadi et al., 2015) with a 200-units hidden layer (400 in total for two concatenated hidden layers in the BiLSTM), the Adam optimizer (Kingma and Ba, 2014) with a

[5]For the statistics regarding the data, we refer the reader to Morante and Blanco (2012).

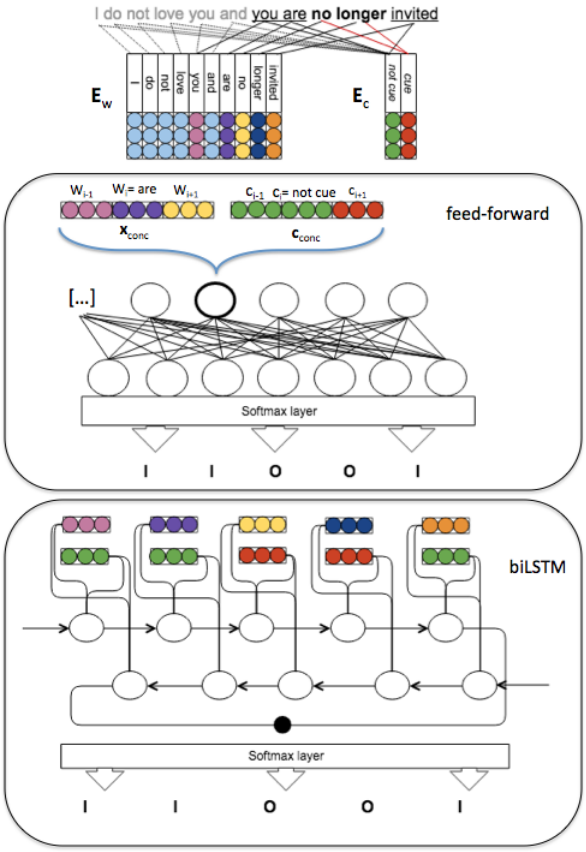


Figure 1: An example of scope detection using feed-forward and BiLSTM for the tokens '*you are* **no longer** *invited*' in the instance in ex. (3b).

starting learning rate of 0.0001, learning rate decay after 10 iterations without improvement and early stopping. In both cases we experimented with different settings:

1. **Simple baseline**: In order to understand how hard the task of negation scope detection is, we created a simple baseline by tagging as part of the scope all the tokens 4 words to the left and 6 to the right of the cue; these values were found to be the average span of the scope in either direction in the training data.

2. **Cue info (C)**: The word-embedding matrix is randomly initialised and updated relying on the training data only. Information about the cue is fed through another set of embedding vectors, as shown in 4. This resembles the 'Closed track' of the *SEM2012 shared task since no external resource is used.

3. **Cue info + external embeddings (E)**: This is the same as setting (2) except that the embeddings are pre-trained using external data. We experimented with both keeping the word-embedding matrix fixed and updating it during training but we found small or no difference between the two settings. To do this, we train a word-embedding matrix using Word2Vec (Mikolov et al., 2013) on 770 million tokens (for a total of 30 million sentences and 791028 types) from the 'One Billion Words Language Modelling' dataset [6] and the Sherlock Holmes data (5520 sentences) combined. The dataset was tokenized and morphological cues split into negation affix and root to match the Conan Doyle's data. In order to perform this split, we matched each word against an hand-crafted list of words containing affixal negation[7]; this method have an accuracy of 0.93 on the Conan Doyle test data.

4. **Adding PoS / Universal PoS information (PoS/uni PoS)**: This was mainly to assess whether we could get further improvement by adding additional information. For all the setting above, we also add an extra embedding input vector for the POS or Universal POS of each word w_i. As for the word and the cue embeddings, PoS-embedding information are fed to the hidden layer through a separate weight matrix. When pre-trained, the training data for the external PoS-embedding matrix is the same used for building the word embedding representation, except that in this case we feed the PoS / Universal PoS tag for each word. As in (3), we experimented with both updating the tag-embedding matrix and keeping it fixed but found again small or no difference between the two settings. In order to maintain consistency with the original data, we perform PoS tagging using the GENIA tagger (Tsuruoka et al., 2005)[8] and then map the resulting tags to universal POS tags.[9]

4.2 Results

The results for the scope detection task are shown in Table 2.

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

[7] The list was courtesy of Ulf Hermjakob and Nathan Schneider.

[8] `https://github.com/saffsd/geniatagger`

[9] Mapping available at `https://github.com/slavpetrov/universal-pos-tags`

Results for both architecture when word-embedding features *only* are used (C and C + E) show that neural networks are a valid alternative for scope detection, with bi-directional LSTM being able to outperform all previously developed classifiers on both scope token recognition and exact scope matching. Moreover, a bi-directional LSTM shows similar performance to the hybrid system of Packard et al. (2014) (rule-based + SVM as a back-off) in absence of any hand-crafted heuristics.

It is also worth noticing that although pre-training the word-embedding and PoS-embedding matrices on external data leads to a slight improvement in performance, the performance of the systems using internal data only is already competitive; this is a particularly positive result considering that the training data is relatively small.

Finally, adding universal POS related information leads to a better performance in most cases. The fact that the best system is built using language-independent features only is an important result when considering the portability of the model across different languages.

4.3 Error analysis

In order to understand the kind of errors our best classifier makes, we performed an error analysis on the held-out set.

First, we investigate whether the per-instance prediction accuracy correlates with scope-related (length of the scope to the left, to the right and combined; maximum length of the gap in a discontinuous scope) and cue-related (type of cue -one-word, prefixal, suffixal, multiword-) variables. We also checked whether the neural network is biased towards the words it has seen in the training(for instance, if it has seen the same token always labeled as O it will then classify it as O). For our best biLSTM system, we found only weak to moderate negative correlations with the following variables:

- *length of the gap*, if the scope is discontinuous (r=-0.1783, p = 0.004);
- *overall scope length* (r=-0.3529, $p < 0.001$);
- *scope length to the left and to the right* (r=-0.3251 and -0.2659 respectively with $p < 0.001$)
- *presence of a prefixal cue* (r=-0.1781, p = 0.004)
- *presence of a multiword cue* (r=-0.1868, p = 0.0023)

meaning that the variables considered are not strong enough to be considered as error patterns.

For this reason we also manually analyzed the 96 negation scopes that the best biLSTM system predicted incorrectly and noticed several error patterns:

- in 5 cases, the scope should only span on the subordinate but end up including elements from the main. In (6) for instance, where the system prediction is reported in curly brackets, the BiLSTM ends up including the main predicate with its subject in the scope.

  (6) You felt so strongly about it that {I knew you could} **not** {think of Beecher without thinking of that also} .

- in 5 cases, the system makes an incorrect prediction in presence of the syntactic inversion, where a subordinate appears before the main clause; in (7) for instance, the system extends the prediction to the main clause when the scope should instead span the subordinate only.

  (7) But {if she does} **not** {wish to shield him she would give his name}

- in 8 cases, where two VPs, one positive and one negative, are coordinated, the system ends up including in the scope the positive VP as well, as shown in (8). We hypothesized this is due to the lack of such examples in the training set.

  (8) Ah, {you do} **n't** {know Sarah 's temper or you would wonder no more} .

As in Packard et al. (2014), we also noticed that in 15 cases, the gold annotations do not follow the guidelines; in the case of a negated adverb in particular, as shown in (9a) and (9b) the annotations do not seem to agree on whether consider as scope only the adverb or the entire clause around it.

	Scope tokens							Exact scope match		
System	gold	tp	fp	fn	*Prec.*	*Rec.*	F_1	*Prec.*	*Rec.*	F_1
Baseline	1830	472	3031	1358	13.47	25.79	17.70	0.0	0.0	0.0
Best closed track: UiO1	N/A	N/A	N/A	N/A	81.99	88.81	85.26	87.43	61.45	72.17
Packard et al. (2014)	N/A	N/A	N/A	N/A	86.1	**90.4**	88.2	98.8	**65.5**	**78.7**
FF - C	1830	1371	273	459	83.39	74.91	78.92	93.61	34.10	50.00
FF - C + PoS	1830	1413	235	417	85.74	77.21	81.25	92.51	37.50	53.33
FF - C + Uni PoS	1830	1435	276	395	83.86	78.41	81.05	93.06	36.57	52.51
FF - C + E	1830	1455	398	375	78.52	79.50	79.01	89.53	30.19	45.16
FF - C + PoS + E	1830	1413	179	417	88.75	77.21	82.58	96.63	44.23	60.68
FF - C + Uni PoS + E	1830	1412	158	418	89.93	77.15	83.05	96.58	43.46	59.94
BiLSTM - C	1830	1583	175	247	90.04	86.50	88.23	98.71	58.77	73.68
BiLSTM - C + PoS	1830	1591	203	239	88.68	86.93	87.80	98.70	58.01	73.07
BiLSTM - C + Uni Pos	1830	**1592**	193	238	89.18	86.95	88.07	98.96	57.63	72.77
BiLSTM - C + E	1830	1570	157	260	90.90	85.79	88.27	99.37	60.83	75.47
BiLSTM - C + PoS + E	1830	1546	148	284	91.26	84.48	87.74	98.75	60.30	74.88
BiLSTM - C + Uni PoS + E	1830	1552	124	272	**92.62**	85.13	**88.72**	**99.40**	63.87	77.77

Table 2: Results for the scope detection task on the held-out set. Results are plotted against the simple baseline, the best system so far (Packard et al., 2014) and the system with the highest F_1 for *scope tokens* classification amongst the ones submitted for the *SEM2012 shared task. We also report the number of gold scope tokens, true positive (tp), false positives(fp) and false negatives(fn).

(9) a. [...] tossing rest**less**ly from side to side [..]

b. [...] glaring help**less**ly at the frightful thing which was hunting him down.

5 Evaluation on synthetic data set

5.1 Methodology

One question left unanswered by previous work is whether the performance of scope detection classifiers is robust against data of a different genre and whether different types of negation lead to difference in performance. To answer this, we compare two of our systems with the only original submission to the *SEM2012 we found available (White, 2012)[10]. We decided to use both our best system, BiLSTM+C+UniPoS+E and a sub-optimal systems, BiLSTM+C+E to also assess the robustness of non-English specific features.

The synthetic test set here used is built on sentences extracted from Simple Wikipedia and manually annotated for cue and scope according to the annotation guidelines released in concomitance with the *SEM2012 shared task (Morante et al., 2011). We created 7 different subsets to test different types of negative sentences:

Simple: we randomly picked 50 positive sentences, containing only one predicate, no dates and no named entities, and we made them negative by adding a negation cue (*do* support or minor morphological changes were added when required). If more than a lexical negation cue fit in the context, we used them all by creating more than one negative counterpart, as shown in (10). The sentences were picked to contain different kind of predicates (verbal, existential, nominal, adjectival).

(10) a. Many people disagree on the topic

b. Many people do **not** disagree on the topic

c. Many people **never** disagree on the topic

Lexical: we randomly picked 10 sentences[11] for each **lexical** (i.e. one-word) cue in training data (these are *not*, *no*, *none*, *nobody*, *never*, *without*)

Prefixal: we randomly picked 10 sentences for each prefixal cue in the training data (*un-*, *im-*, *in-*, *dis-*, *ir-*)

Suffixal: we randomly picked 10 sentences for the suffixal cue *-less*.

Multi-word: we randomly picked 10 sentences for each multi-word cue (*neither...nor*,*no longer*,*by no means*).

Unseen: we include 10 sentences for each of the negative prefixes *a-* (e.g. *a*-cyclic), *ab-* (e.g. *ab*-normal) *non-* (e.g. *non*-Communist) that are not annotated as cue in the Conan Doyle corpus,

[10] In order for the results to be comparable, we feed White's system with the cues from the gold-standard instead of automatically detecting them.

[11] In some cases, we ended up with more than 10 examples for some cues given that some of the sentences we picked contained more than a negation instance.

		Scope tokens							Exact scope match		
	Data	gold	tp	fp	fn	*Prec.*	*Rec.*	F_1	*Prec.*	*Rec.*	F_1
White (2012)	simple	850	**830**	0	20	100.00	97.65	**98.81**	100.00	93.98	**96.90**
	lexical	814	652	101	162	86.59	80.10	83.22	100.00	58.41	**73.75**
	prefixal	316	232	103	83	68.98	73.40	**71.12**	100.00	32.76	**49.35**
	suffixal	100	**78**	7	22	91.76	78.00	**84.32**	100.00	69.23	**81.82**
	multi-word	269	190	12	49	89.62	70.63	79.00	100.00	9.00	16.67
	unseen	220	138	40	82	77.53	62.73	69.35	100.00	38.89	**56.00**
	avg.	2569	**2120**	263	418	**85.74**	**77.08**	**80.97**	**100.00**	**50.37**	**62.41**
BiLSTM - C+ E	simple	850	827	0	23	100.00	97.29	98.62	100.00	88.72	94.02
	lexical	814	618	120	133	85.01	83.66	84.33	100.00	40.35	57.50
	prefixal	316	**235**	156	81	60.10	74.36	66.47	100.00	10.34	18.75
	suffixal	100	53	5	47	91.52	53.46	67.50	100.00	15.28	26.66
	multi-word	269	192	22	79	93.65	71.37	81.01	100.00	36.36	**53.00**
	unseen	220	151	79	69	66.09	69.05	67.54	100.00	22.22	36.36
	avg.	2569	2076	382	432	82.72	74.86	77.57	100.00	35.54	47.76
BiLSTM - C+ UniPos + E	simple	850	816	0	34	100.00	96	97.95	100.00	82.70	90.05
	lexical	814	**668**	97	146	87.32	82.06	**84.61**	100.00	42.10	59.25
	prefixal	316	231	128	85	64.34	73.10	68.44	100.00	20.68	34.28
	suffixal	100	54	3	47	94.73	53.46	68.35	100.00	38.46	55.55
	multi-word	269	**202**	19	67	91.40	75.09	**82.44**	100.00	27.27	42.85
	unseen	220	**152**	56	71	73.07	68.16	**70.53**	100.00	25.00	40.00
	avg.	2569	**2123**	303	449	85.14	74.64	78.72	100.00	39.36	53.66

Table 3: Results for the scope detection task on the synthetic test set.

to test whether the system can generalise the classification to unseen cues.[12]

5.2 Results

Table 3. shows the results for the comparison on the synthetic test set. The first thing worth noticing is that by using word-embedding features only it is possible to reach comparable performance with a classifier using syntactic features, with universal PoS generally contributing to a better performance; this is particularly evident in the *multi-word* and *lexical* sub-sets. In general, genre effects hinder both systems; however, considering that the training data is less than 1000 sentences, results are relatively good.

Performance gets worse when dealing with morphological cues and in particular in the case of our classifier, with suffixal cues; at a closer inspection however, the cause of such poor performance is attributable to a discrepancy between the annotation guidelines and the training data, already noted in §4.4. The guidelines state in fact that '*If the negated affix is attached to an adverb that is a complement of a verb, the negation scopes over the entire clause*'(Morante et al., 2011, p. 21) and we annotated suffixal negation in this way. However, 3 out of 4 examples of suffixal negation in adverbs in the training data (e.g. 9a.) mark the scope on the adverbial root only and that's what our classifiers learn to do.

Finally, it can be noticed that our system does worse at exact scope matching than the CRF classifier. This is because White (2012)'s CRF model is build on constituency-based features that will then predict scope tokens based on constituent boundaries (which, as we said, are good indicator of scope boundaries), while neural networks, basing the prediction only on word-embedding information, might extend the prediction over these boundaries or leave 'gaps' within.

6 Conclusion and Future Work

In this work, we investigated and confirmed that neural networks sequence-to-sequence models are a valid alternative for the task of detecting the scope of negation. In doing so we offer a detailed analysis of its performance on data of different genre and containing different types of negation, also in comparison with previous classifiers, and found that non-English specific continuous representation can perform batter than or on par with more fine-grained structural features.

Future work can be directed towards answering two main questions:

Can we improve the performance of our classifier? To do this, we are going to explore whether adding language-independent structural informa-

[12]The data, along with the code, is freely available at `https://github.com/ffancellu/NegNN`

tion (e.g. universal dependency information) can help the performance on exact scope matching.

Can we transfer our model to other languages? Most importantly, we are going to test the model using word-embedding features extracted from a bilingual embedding space.

Acknowledgments

This project was also founded by the European Unions Horizon 2020 research and innovation programme under grant agreement No 644402 (HimL).

The authors would like to thank Naomi Saphra, Nathan Schneider and Claria Vania for the valuable suggestions and the three anonymous reviewers for their comments.

References

M Abadi, A Agarwal, P Barham, E Brevdo, Z Chen, C Citro, GS Corrado, A Davis, J Dean, M Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems. *White paper, Google Research*.

Amjad Abu-Jbara and Dragomir Radev. 2012. Umichigan: A conditional random field model for resolving the scope of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 328–334. Association for Computational Linguistics.

Miguel Ballesteros, Alberto Díaz, Virginia Francisco, Pablo Gervás, Jorge Carrillo De Albornoz, and Laura Plaza. 2012. Ucm-2: a rule-based approach to infer the scope of negation via dependency parsing. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 288–293. Association for Computational Linguistics.

Valerio Basile, Johan Bos, Kilian Evang, and Noortje Venhuizen. 2012. Ugroningen: Negation detection with discourse representation structures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 301–309. Association for Computational Linguistics.

Eduardo Blanco and Dan I Moldovan. 2011. Some issues on detecting negation from text. In *FLAIRS Conference*, pages 228–233. Citeseer.

Md Chowdhury and Faisal Mahbub. 2012. Fbk: Exploiting phrasal and contextual clues for negation scope detection. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 340–346. Association for Computational Linguistics.

Jorge Carrillo de Albornoz, Laura Plaza, Alberto Díaz, and Miguel Ballesteros. 2012. Ucm-i: A rule-based syntactic approach for resolving the scope of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 282–287. Association for Computational Linguistics.

Federico Fancellu and Bonnie L Webber. 2014. Applying the semantics of negation to smt through n-best list re-ranking. In *EACL*, pages 598–606.

Federico Fancellu and Bonnie Webber. 2015. Translating negation: A manual error analysis. *ExProM 2015*, page 1.

Binod Gyawali and Thamar Solorio. 2012. Uabcoral: a preliminary study for resolving the scope of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 275–281. Association for Computational Linguistics.

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

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Emanuele Lapponi, Erik Velldal, Lilja Øvrelid, and Jonathon Read. 2012. Uio 2: sequence-labeling negation using dependency features. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 319–327. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Roser Morante and Eduardo Blanco. 2012. * sem 2012 shared task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference*

on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation, pages 265–274. Association for Computational Linguistics.

Roser Morante, Sarah Schrauwen, and Walter Daelemans. 2011. Annotation of negation cues and their scope: Guidelines v1. *Computational linguistics and psycholinguistics technical report series, CTRS-003*.

Woodley Packard, Emily M Bender, Jonathon Read, Stephan Oepen, and Rebecca Dridan. 2014. Simple negation scope resolution through deep parsing: A semantic solution to a semantic problem. In *ACL (1)*, pages 69–78.

Vinodkumar Prabhakaran and Branimir Boguraev. 2015. Learning structures of negations from flat annotations. *Lexical and Computational Semantics (* SEM 2015)*, page 71.

Jonathon Read, Erik Velldal, Lilja Øvrelid, and Stephan Oepen. 2012. Uio 1: Constituent-based discriminative ranking for negation resolution. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 310–318. Association for Computational Linguistics.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin-Dong Kim, Tomoko Ohta, John McNaught, Sophia Ananiadou, and Jun'ichi Tsujii. 2005. Developing a robust part-of-speech tagger for biomedical text. *Advances in informatics*, pages 382–392.

Erik Velldal, Lilja Øvrelid, Jonathon Read, and Stephan Oepen. 2012. Speculation and negation: Rules, rankers, and the role of syntax. *Computational linguistics*, 38(2):369–410.

Dominikus Wetzel and Francis Bond. 2012. Enriching parallel corpora for statistical machine translation with semantic negation rephrasing. In *Proceedings of the Sixth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 20–29. Association for Computational Linguistics.

James Paul White. 2012. Uwashington: Negation resolution using machine learning methods. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 335–339. Association for Computational Linguistics.

CSE: Conceptual Sentence Embeddings based on Attention Model

Yashen Wang, Heyan Huang*, Chong Feng, Qiang Zhou, Jiahui Gu and Xiong Gao
Beijing Engineering Research Center of High Volume Language Information Processing
and Cloud Computing Applications, School of Computer,
Beijing Institute of Technology, Beijing, P. R. China
{yswang,hhy63,fengchong,qzhou,gujh,gaoxiong}@bit.edu.cn

Abstract

Most sentence embedding models typically represent each sentence only using word surface, which makes these models indiscriminative for ubiquitous homonymy and polysemy. In order to enhance representation capability of sentence, we employ conceptualization model to assign associated concepts for each sentence in the text corpus, and then learn conceptual sentence embedding (CSE). Hence, this semantic representation is more expressive than some widely-used text representation models such as latent topic model, especially for short-text. Moreover, we further extend CSE models by utilizing a local attention-based model that select relevant words within the context to make more efficient prediction. In the experiments, we evaluate the CSE models on two tasks, text classification and information retrieval. The experimental results show that the proposed models outperform typical sentence embed-ding models.

1 Introduction

Many natural language processing applications require the input text to be represented as a fixed-length feature, of which sentence representation is very important. Perhaps the most common fixed-length vector representation for texts is the bag-of-words or bag-of-n-grams (Harris, 1970). However, they suffer severely from data sparsity and high dimensionality, and have very little sense about the semantics of words or the distances between the words. Recently, in sentence representation and classification, deep neural network (DNN) approaches have achieved state-of-the-art results (Le and Mikolov, 2014; Liu et al., 2015; Palangi et al., 2015; Wieting et al., 2015). Despite of their usefulness, recent sentence embeddings face several challenges: (i) Most sentence embedding models represent each sentence only using word surface, which makes these models indiscriminative for ubiquitous polysemy; (ii) For short-text, however, neither parsing nor topic modeling works well because there are simply not enough signals in the input; (iii) Setting window size of context words is very difficult. To solve these problems, we must derive more semantic signals from the input sentence, e.g., concepts. Besides, we should assigned different attention for different contextual word, to enhance the influence of words that are relevant for each prediction.

This paper proposed Conceptual Sentence Embedding (CSE), an unsupervised framework that learns continuous distributed vector representations for sentence. Specially, by innovatively introducing concept information, this concept-level vector representations of sentence are learned to predict the surrounding words or target word in contexts. Our research is inspired by the recent work in learning vector representations of words using deep learning strategy (Mikolov et al., 2013a; Le and Mikolov, 2014). More precisely, we first obtain concept distribution of the sentence, and generate corresponding concept vector. Then we concatenate or average the sentence vector, contextual word vectors with concept vector of the sentence, and predict the target word in the given context. All of the sentence vectors and word vectors are trained by the stochastic gradient descent and backpropagation (Rumelhart et al., 1986). At prediction time, sentence vectors are inferred by fixing the word vectors and observed sentence vectors, and training the new sentence vector until convergence.

In parallel, the concept of attention has gained

* The contact author.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 505–515,
Berlin, Germany, August 7-12, 2016.

popularity recently in neural natural language processing researches, which allowing models to learn alignments between different modalities (Bahdanau et al., 2014; Bansal et al., 2014; Rush et al., 2015). In this work, we further propose the extensions to CSE, which adds an attention model that considers contextual words differently depending on the word type and its relative position to the predicted word. The main intuition behind the extended model is that prediction of a word is mainly dependent on certain words surrounding it.

In summary, the basic idea of CSE is that, we allow each word to have different embeddings under different concepts. Taking word apple into consideration, it may indicate a fruit under the concept *food*, and indicate an IT company under the concept *information technology*. Hence, concept information significantly contributes to the discriminative of sentence vector. Moreover, an important advantage of the proposed conceptual sentence embeddings is that they could be learned from unlabeled data. Another advantage is that we take the word order into account, in the same way of n-gram model, while bag-of-n-grams model would create a very high-dimensional representation that tends to generalize poorly.

To summarize, this work contributes on the following aspects: We integrate concepts and attention-based strategy into basic sentence embedding representation, and allow the resulting conceptual sentence embedding to model different meanings of a word under different concept. The experimental results on text classification task and information retrieval task demonstrate that this concept-level sentence representation is robust. The outline of the paper is as follows. Section 2 surveys related researches. Section 3 formally de-scribes the proposed model of conceptual sentence embedding. Corresponding experimental results are shown in Section 4. Finally, we conclude the paper.

2 Related Works

Conventionally, one-hot sentence representation has been widely used as the basis of bag-of-words (BOW) text model. However, it can-not take the semantic information into consideration. Recently, in sentence representation and classification, deep neural network approaches have achieved state-of-the-art results (Le and Mikolov, 2014; Liu et al., 2015; Ma et al., 2015; Palangi et al., 2015; Wieting et al., 2015), most of which are inspired by word embedding (Mikolov et al., 2013a). (Le and Mikolov, 2014) proposed the paragraph vector (PV) that learns fixed-length representations from variable-length pieces of texts. Their model represents each document by a dense vector which is trained to predict words in the document. However, their model depends only on word surface, ignoring semantic information such as topics or concepts. In this paper, we extent PV by introducing concept information.

Aiming at enhancing discriminativeness for ubiquitous polysemy, (Liu et al., 2015) employed latent topic models to assign topics for each word in the text corpus, and learn topical word embeddings (TWE) and sentence embeddings based on both words and their topics. Besides, to combine deep learning with linguistic structures, many syntax-based embedding algorithms have been proposed (Severyn et al., 2014; Wang et al., 2015b) to utilize long-distance dependencies. However, short-texts usually do not observe the syntax of a written language, nor do they contain enough signals for statistical inference (e.g., topic model). Therefore, neither parsing nor topic modeling works well because there are simply not enough signals in the input, and we must derive more semantic signals from the input, e.g., concepts, which have been demonstrated effective in knowledge representation (Wang et al., 2015c; Song et al., 2015). Shot-text conceptualization, is an interesting task to infer the most likely concepts for terms in the short-text, which could help better make sense of text data, and extend the texts with categorical or topical information (Song et al., 2011). Therefore, our models utilize short-text conceptualization algorithm to discriminate concept-level sentence senses and provide a good performance on short-texts.

Recently, attention model has been used to improve many neural natural language pro-cessing researches by selectively focusing on parts of the source data (Bahdanau et al., 2014; Bansal et al., 2014; Wang et al., 2015a). To the best of our knowledge, there has not been any other work exploring the use of attentional mechanism for sentence embeddings.

3 Conceptual Sentence Embedding

This paper proposes four conceptual sentence embedding models. The first one is based on continu-

ous bag-of-word model (denoted as **CSE-1**) which have not taken word order into consideration. To overcome this drawback, its extension model (denoted as **CSE-2**), which is based on Skip-Gram model, is proposed. Based on the basic conceptual sentence embedding models above, we obtain their variants (**aCSE-1** and **aCSE-2**) by introducing attention model.

3.1 CBOW Model & Skip-Gram Model

As inspiration of the proposed conceptual sentence embedding models, we start by discussing previous models for learning word vectors (Mikolov et al., 2013a; Mikolov et al., 2013b) firstly.

Let us overview the framework of Continuous Bag-of-Words (CBOW) firstly, which is shown in Figure 1(a). Each word is typically mapped to an unique vector, represented by a column in a word matrix $\mathbf{W} \in \Re^{d*|V|}$. Wherein, V denotes the word vocabulary and d is embedding dimension of word. The column is indexed by position of the word in V. The concatenation or average of the vectors, the context vector w_t, is then used as features for predicting the target word in the current context. Formally, Given a sentence $S = \{w_1, w_2, \ldots, w_l\}$, the objective of CBOW is to maximize the average log probability:

$$\mathcal{L}(S)=\frac{1}{(l-2k-2)}\sum_{t=k+1}^{l-k}\log Pr(w_t|w_{t-k},\cdots,w_{t+k}) \quad (1)$$

Wherein, k is the context windows size of target word w_t. The prediction task is typically done via a softmax function, as follows:

$$Pr(w_t|w_{t-k},\cdots,w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_{w_i \in V} e^{y_{w_i}}} \quad (2)$$

Each of $y_(w_t)$ is an un-normalized log-probability for each target word w_t, as follows:

$$y_{w_t} = \mathbf{U}h(w_{t-k},\ldots,w_{t+k});\mathbf{W}) + b \quad (3)$$

Wherein, $\mathbf{U}$ and b are softmax parameters. And $h(\cdot)$ is constructed by a concatenation or average of word vectors $\{\mathbf{w}_{t-k},\ldots,\mathbf{w}_{t+k}\}$ extracted from word matrix $\mathbf{W}$ according to $\{w_{t-k},\ldots,w_{t+k}\}$. For illustration purposes, we utilize average here. On the condition of average, the context vector c_t is obtained by averaging the embeddings of each word, as follows:

$$\mathbf{c}_t = \frac{1}{2k}\sum_{-k\le c\le k, c\neq 0}\mathbf{w}_{t+c} \quad (4)$$

The framework of Skip-Gram (Figure 1(b)) aims to predict context words given a target word w_t in a sliding window, instead of predicting the current word based on its context. Formally, given a sentence $S = \{w_1, w_2, \ldots, w_l\}$, the objective of Skip-Gram is to maximize the following average log probability:

$$\mathcal{L}(S)=\frac{1}{(l-2k)}\sum_{t=k+1}^{l-k}\sum_{-k\le c\le k, c\neq 0}\log Pr(w_{t+c}|w_t) \quad (5)$$

Wherein, $\mathbf{w}_t$ and $\mathbf{w}_c$ are respectively the vector representations of the target word w_t and the context word w_c. Usually, during the training stage of CBOW and Skip-Gram: (i) in order to make the models efficient for learning, the techniques of hierarchical softmax and negative sampling are used to ensure the models efficient for learning (Morin and Bengio, 2005; Mikolov et al., 2013a); (ii) the word vectors are trained by using stochastic gradient descent where the gradient is obtained via backpropagation (Rumelhart et al., 1986). After the training stage converges, words with similar meaning are mapped to a similar position in the semantic vector space. e.g., 'powerful' and 'strong' are close to each other.

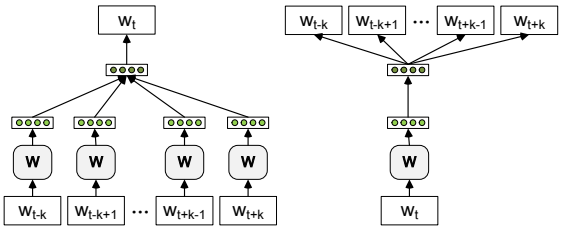


Figure 1: (a) CBOW model and (b) Skip-Gram model.

3.2 CSE based on CBOW Model

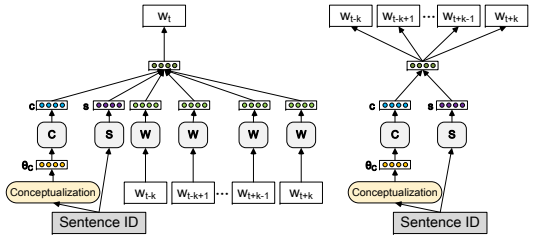
Intuitively, the proposed (attention-based) conceptual sentence embedding model for learning sentence vectors, is inspired by the methods for learning the word vectors. The inspiration is that, in researches of word embeddings: (i) The word vectors are asked to contribute to a prediction task about the target word or the surrounding words in the context; (ii) The word representation vectors are initialized randomly, however they could

finally capture precise semantics as an indirect result. Therefore, we will utilize this idea in our sentence vectors in a similar manner: The concept-associated sentence vectors are also asked to contribute to the prediction task of the target word or surrounding words in given contextual text windows. Furthermore, attention model will attribute different influence value to different contextual words.

We describe the first conceptual sentence embedding model, denoted as **CSE-1**, which is based on CBOW. In the framework of **CSE-1** (Figure 2(a)), each sentence, denoted by sentence ID, is mapped to a unique vector **s**, represented by a column in matrix **S**. And its concept distribution θ_C are generated from a knowledge-based text conceptualization algorithm (Wang et al., 2015c). Moreover, similar to word embedding methods, each word w_i is also mapped to a unique vector $\mathbf{w}_i$, represented by a column in matrix W. The surrounding words in contextual text window $\{w_{t-k}, \ldots, w_{t+k}\}$, sentence ID and concept distribution θ_C corresponding to this sentence are the inputs. Besides, **C** is a fixed linear operator similar to the one used in (Huang et al., 2013) that converts the concept distribution θ_C to a concept vector, denoted as **c**. Note that, this makes our model very different from (Le and Mikolov, 2014) where no concept information is used, and experimental results demonstrate the efficiency of introducing concept information. It is clear that **CSE-1** also does not take word order into consideration just like CBOW.

Afterward, the sentence vector **s**, surrounding word vectors $\{\mathbf{w}_{t-k}, \ldots, \mathbf{w}_{t+k}\}$ and the concept vector **c** are concatenated or averaged to predict the target word w_t in current context. In reality, the only change in this model compared to the word embedding method is in Eq. 3, where $h(\cdot)$ is constructed from not only **W** but also **C** and **S**. Note that, the sentence vector is shared across all contexts generated from the same sentence but not across sentences. Wherein, the contexts are fixed-length (length is $2k$) and sampled from a sliding window over the current sentence. However, the word matrix **W** is shared across sentences.

In summary, the procedure of **CSE-1** itself is described as follows. A probabilistic conceptualization algorithm (Wang et al., 2015c) is employed here to obtain the corresponding concepts about given sentence: Firstly, we preprosess and segment the given sentence into a set of words; Then, based on a probabilistic lexical knowledge-base Probase (Wu et al., 2012), the heterogeneous semantic graph for these words and their corresponding concepts are constructed (Figure 3 shows an example); Finally, we utilize a simple iterative process to identify the most likely mapping from words to concepts. After efforts above, we could conceptualize words in given sentence, and access the concepts and corresponding probabilities, which is the concept distribution θ_C mentioned before. Note that, the concept distribution yields an important influence on the entire framework of conceptual sentence embedding, by contributing greatly to the semantic representation.



(a) (b)

Figure 2: **CSE-1** model (a) and **CSE-2** model (b). Green circles indicate word embeddings, blue circles indicate concept embeddings, and purple circles indicate sentence embeddings. Besides, orange circles indicate concept distribution θ_C generated by knowledge-based text conceptualization algorithm.

During the training stage, we aim at obtaining word matrix **W**, sentence matrix **S**, and softmax weights $\{\mathbf{U}, b\}$ on already observed sentences. The techniques of hierarchical softmax and negative sampling are used to make the model efficient for learning. **W** and **S** are trained using stochastic gradient descent: At each step of stochastic gradient descent, we sample a fixed-length context from the given sentence, compute the error gradient which is obtained via backpropagation, and then use the gradient to update the parameters. During the inferring stage, we get sentence vectors for new sentences (unobserved before) by adding more columns in **S** and gradient descending on **S** while holding **W**, **U** and b fixed. Finally, we use **S** to make a prediction about multi-labels by using a standard classifier in output layer.

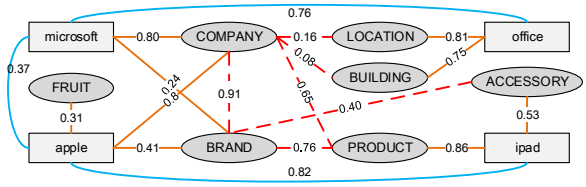


Figure 3: Semantic graph of example sentence microsoft unveils office for apples ipad. Rectangles indicate terms occurred in given sentence, and ellipses indicate concept defined in knowledge-base (e.g., Probase). Bule solid links indicate *isA* relationship between terms and concepts, and red dashed lines indicate correlation relationship between two concepts. Numerical values on the line is corresponding probabilities.

3.3 CSE based on Skip-Gram Model

The above method considers the combination of the sentence vector with the surrounding word vectors and concept vector to predict the target word in given text window. However, it loss information about word order somehow, just like CBOW. In fact, there exists another for modeling the prediction procedure: we could ignore the context words in the input, but force the model to predict words randomly sampled from the fix-length contexts in the output. As is shown in Figure 2 (b), only sentence vector **s** and concept vector **c** are used to predict the next word in a text window. That means, contextual words are no longer used as inputs, whereas they become what the output layer predict. Hence, this model is similar to the Skip-Gram model in word embedding (Mikolov et al., 2013b). In reality, what this means is that at each iteration of stochastic gradient descent, we sample a text window $\{w_{t-k}, \ldots, w_{t+k}\}$, then sample a random word from this text window and form a classification task given the sentence vector **s** and corresponding concept vector **c**.

We denote this sort of conceptual sentence embedding model as **CSE-2**. The scheme of **CSE-2** is similar to that of **CSE-1** as described above. In addition to being conceptually simple, **CSE-2** requires to store less data. We only need to store {**U**,b,**S**} as opposed to {**U**,b,**S**,**W**} in **CSE-1**.

3.4 CSE based on Attention Model

As mentioned above, setting a good value for contextual window size k is difficult. Because a larger value of k may introduce a degenerative behavior in the model, and more effort is spent predicting words that are conditioned on unrelated words, while a smaller value of k may lead to cases where the window size is not large enough include words that are semantically related (Bansal et al., 2014; Wang et al., 2015a). To solve these problems , we extend the proposed models by introducing attention model (Bahdanau et al., 2014; Rush et al., 2015), by allowing it to consider contextual words within the window in a non-uniform way. For illustration purposes, we extend **CSE-1** here with attention model. Following (Wang et al., 2015a), we rewrite Eq.(4) as follows:

$$\mathbf{c}_t = \frac{1}{2k} \sum_{-k \le c \le k, c \ne 0} a_{t+c}(w_{t+c}) \mathbf{w}_{t+c} \qquad (6)$$

Wherein we replace the average of the surrounding word vectors in Eq.(4) with a weighted sum of the these vectors. That means, each contextual word w_{t+c} is attributed a different attention level, representing how much the attention model believes whether it is important to look at in order to predict the target word w_t. The attention factor $a_i(w_i)$ for word w_i in position i is formulated as a softmax function over contextual words (Bahdanau et al., 2014), as follows:

$$a_i(w) = \frac{e^{d_{w,i}} + r_i}{\sum_{-k \le c \le k, c \ne 0} e^{d_{w,c}} + r_c} \qquad (7)$$

Wherein, $d_{w,i}$ is an element of matrix $\mathbf{D} \in \Re^{|V| * 2k}$, which is a set of parameters determining the importance of each word type in each relative position i (distance to the left/right of target word w_t). Moreover, r_i, an element of $\mathbf{R} \in \Re^{2k}$, is a bias, which is conditioned only on the relative position i. Note that, attention models have been reported expensive for large tables in terms of storage and performance (Bahdanau et al., 2014; Wang et al., 2015a). Nevertheless the computation consumption here is simple, and compute the attention of all words in the input requires $2k$ operations, as it simply requires retrieving on value from the lookup-matrix **D** for each word and one value from the bias vector **R** for each word in the context. Although this strategy may not be the best approach and there exist more elaborate attention models (Bahdanau et al., 2014; Luong et al., 2015), the proposed attention model is a proper balance of computational efficiency and complexity.

Thus, besides {**W**,**C**,**S**} in CSE models, **D** and **R** are added into parameter set which relates to

gradients of the loss function Eq.(1). All parameters are computed with backpropagation and updated after each training instance using a fixed learning rate. We denote the attention-based **CSE-1** model above as **aCSE-1**. With limitation of space, attention variant of **CSE-2**, denoted as **aCSE-2**, is not described here, however the principle is similar to **aCSE-1**.

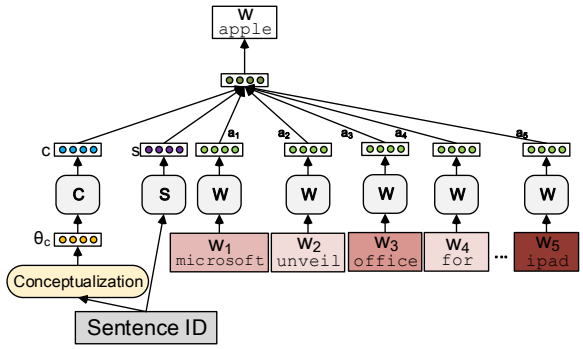


Figure 4: **aCSE-1** model. The illustration of example sentence 'mcrosoft unveils office for apple's ipad' for predicting word 'apple'.

Taking example 'microsoft unveils office for apple's ipad' into consideration. The prediction of the polysemy word 'apple' by **CSE-1** is shown in Figure 4, and darker cycle cell indicate higher attention value. We could observe that preposition word 'for' tend to be attributed very low attention, while context words, especially noun-words which contribute much to conceptualization (such as 'ipad', 'office', and 'microsoft') are attributed higher weights as these word own more predictive power. Wherein, 'ipad' is assigned the highest attention value as it close to the predicted word and co-occurs with it more frequently.

As described before, concept distribution θ_C yields a considerable influence on conceptual sentence embedding. This is because, each dimensionality of this distribution denotes the probability of the concept (topic or category) this sentence is respect to. In other words, the concept distribution is a solid semantic representation of the sentence. Nevertheless, the information in each dimensionality of sentence (or word) vector makes no sense. Hence, there exist a linear operator in **CSE-1**, **CSE-2**, **aCSE-1**, and **aCSE-2**, which transmit the concept distribution into word vector and sentence vector, as shown in Figure 2 and Figure 3.

4 Experiments and Results

In this section, we show experiments on two text understanding problems, text classification and information retrieval, to evaluate related models in several aspects. These tasks are always used to evaluate the performance of sentence embedding methods (Liu et al., 2015; Le and Mikolov, 2014). The source codes and datasets of this paper are publicly available[1].

4.1 Datasets

We utilize four datasets for training and evaluating. For text classification task, we use three datasets: **NewsTile**, **TREC** and **Twitter**. Dataset **Tweet11** is used for evaluation in information retrieval task. Moreover, we construct dataset **Wiki** to fully train topic model-based models.

NewsTitle: The news articles are extracted from a large news corpus, which contains about one million articles searched from Web pages. We organize volunteers to classify these news articles manually into topics according its article content (Song et al., 2015), and we select six topics: company, health, entertainment, food, politician, and sports. We randomly select 3,000 news articles in each topic, and only keep its title and its first one line of article. The average word count of titles is 9.41.

TREC: It is the corpus for question classification on TREC (Li and Roth, 2002), which is widely used as benchmark in text classification task. There are 5,952 sentences in the entire dataset, classified into the 6 categories as follows: person, abbreviation, entity, description, location and numeric.

Tweet11: This is the official tweet collections used in TREC Microblog Task 2011 and 2012 (Ounis et al., 2011; Soboroff et al., 2012). Using the official API, we crawled a set of local copies of the corpus. Our local Tweets11 collection has a sample of about 16 million tweets, and a set of 49 (TMB2011) and 60 (TMB2012) timestamped topics.

Twitter: This dataset is constructed by manually labeling the previous dataset Tweet11. Similar to dataset NewsTitle, we ask our volunteers to label these tweets. After manually labeling, the dataset contains 12,456 tweets which are in four

[1] http://hlipca.org/index.php/2014-12-09-02-55-58/2014-12-09-02-56-24/58-acse

categories: company, country, entertainment, and device. The average length of the tweets is 13.16 words. Because of its noise and sparsity, this social media dataset is very challenging for the comparative models.

Moreover, we also construct a Wikipedia dataset (denoted as **Wiki**) for training. We preprocess the Wikipedia articles[2] with the following rules. First, we remove the articles less than 100 words, as well as the articles less than 10 links. Then we remove all the category pages and disambiguation pages. Moreover, we move the content to the right redirection pages. Finally we obtain about 3.74 million Wikipedia articles for indexing and training.

4.2 Alternative Algorithms

We compare the proposed models with the following comparative algorithms.

BOW: It is a simple baseline which represents each sentence as bag-of-words, and uses TF-IDF scores (Salton and Mcgill, 1986) as features to generate sentence vector.

LDA: It represents each sentence as its topic distribution inferred by latent dirichlet allocation (Blei et al., 2003). We train this model in two ways: (i) on both Wikipedia articles and the evaluation datasets above, and (ii) only on the evaluation datasets. We report the better of the two.

PV: Paragraph Vector models are variable-length text embedding models, including the distributed memory model (**PV-DM**) and the distributed bag-of-words model (**PV-DBOW**). It has been reported to achieve the state-of-the-art performance on task of sentiment classification (Le and Mikolov, 2014), however it only utilizes word surface.

TWE: By taking advantage of topic model, it overcome ambiguity to some extent (Liu et al., 2015). Typically, TWE learn topic models on training set. It further learn topical word embeddings using the training set, then generate sentence embeddings for both training set and testing set. (Liu et al., 2015) proposed three models for topical word embedding, and we present the best results here. Besides, We also train TWE in two ways like **LDA**.

[2] `http://en.wikipedia.org/wiki/Wikipedia:Databasedown-load`

4.3 Experiment Setup

The details about parameter settings of the comparative algorithms are described in this section, respectively. For **TWE**, **CSE-1**, **CSE-2** and their attention variants **aCSE-1**, and **aCSE-2**, the structure of the hierarchical softmax is a binary Huffman tree (Mikolov et al., 2013a; Mikolov et al., 2013b), where short codes are assigned to frequent words. This is a good speedup trick because common words are accessed quickly (Le and Mikolov, 2014).We set the dimensions of sentence, word, topic and concept embeddings as 5,000, which is like the number of concept clusters in Probase (Wu et al., 2012; Wang et al., 2015c). Meanwhile, we have done many experiments on choosing the context window size (k). We perform experiments on increasing windows size from 3 to 11, and different size works differently on different dataset with different average length of short-texts. And we choose the result of windows size of 5 present here, because it performs best in almost datasets. Usually, in project layer, the sentence vector, the context vector and the concept vectors could be averaged or concatenated for combination to predict the next word in a context. We perform experiments following these two strategies respectively, and report the better of the two. In fact, the concatenation performs better since averaging different types of vectors may cause loss of information somehow.

For **BOW** and **LDA**, we remove stop words by using InQuery stop-word list. For **BOW**, we select top 50,000 words according to TF-IDF scores as features. For both **LDA** and **TWE**, in the text classification task, we set the topic number to be the cluster number or twice, and report the better of the two; while in the information retrieval task, we experimented with a varying number of topics from 100 to 500, which gives similar performance, and we report the final results of using 500 topics.

In summary, we use the sentence vectors generated by each algorithm as features and run a linear classifier using Liblinear (Fan et al., 2010) for evaluation.

4.4 Text Classification

In this section, we run the multi-class text classification experiments on the dataset **NewsTitle**, **Twitter**, and **TREC**. We report precision, recall and F-measure for comparison (as shown in Table 1). Statistical t-test are employed here. To de-

	NewsTitle			Twitter			TREC		
Model	**P**	**R**	**F**	**P**	**R**	**F**	**P**	**R**	**F**
BOW	0.782	0.791	0.786	0.437	0.429	0.433	0.892	0.891	0.891
LDA	0.717	0.705	0.711	0.342	0.308	0.324	0.813	0.809	0.811
PV-DBOW	0.725	0.719	0.722	0.413	0.408	0.410	0.824	0.819	0.821
PV-DM	0.748	0.740	0.744	0.426	0.424	0.425	0.836	0.825	0.830
TWE	0.811^{β}	0.803^{β}	0.807^{β}	0.459^{β}	0.438	0.448^{β}	0.898^{β}	0.886^{β}	0.892^{β}
CSE-1	0.815	0.809	0.812	0.461	**0.449**	0.454	0.896	0.890	0.893
CSE-2	**0.827**	**0.817**	**0.822**	**0.475**	0.447	**0.462**	**0.901**	**0.895**	**0.898**
aCSE-1	0.824	0.818	0.821	0.471	**0.454**	0.462	0.901	0.897	0.899
aCSE-2	$\mathbf{0.831}^{\alpha\beta}$	$\mathbf{0.820}^{\alpha\beta}$	$\mathbf{0.825}^{\alpha\beta}$	$\mathbf{0.477}^{\alpha\beta}$	$0.450^{\alpha\beta}$	$\mathbf{0.463}^{\alpha\beta}$	$\mathbf{0.909}^{\alpha\beta}$	$\mathbf{0.904}^{\alpha\beta}$	$\mathbf{0.906}^{\alpha\beta}$

Table 1: Evaluation results of multi-class text classification task.

cide whether the improvement by method A over method B is significant, the t-test calculates a value p based on the performance of A and B. The smaller p is, the more significant is the improvement. If the p is small enough ($p < 0.05$), we conclude that the improvement is statistically significant. In Table 1, the superscript α and β respectively denote statistically significant improvements over **TWE** and **PV-DM**.

Without regard to attention-based model firstly, we could conclude that **CSE-2** outperforms all the baselines significantly (expect for recall in **Twitter**). This fully indicates that the proposed model could capture more precise semantic information of sentence as compared to topic model-based models and other embedding models. Because the concepts we obtained contribute significantly to the semantic representation of sentence, meanwhile suffer slightly from texts noisy and sparsity. Moreover, as compared to **BOW**, **CSE-1** and **CSE-2** manage to reduce the feature space by 90 percent, while among them, **CSE-2** needs to store less data comparing with **CSE-1**. By introducing attention model, performances of CSE models are entirely promoted, as compared **aCSE-2** with original **CSE-2**, which demonstrates the advantage of attention model.

PV-DM and **PV-DBOW** are reported as the state-of-the-art model for sentence embedding. From the results we can also see that, the proposed model **CSE-2** and **aCSE-2** significantly outperforms **PV-DBOW**. As expected, **LDA** performs worst, even worse than **BOW**, because it is trained on very sparse short-texts (i.e., question and social media text), where there is no enough statistical information to infer word co-occurrence and word topics, and latent topic model suffer extremely from the sparsity of the short-text. Besides, the number of topics slightly impacts the performance of **LDA**. In future, we may conduct more experiments to explore genuine reasons. As described in section 3, **aCSE-2** (**CSE-2**) performs better than **aCSE-1** (**CSE-1**), because the former one take word order into consideration. Based on Skip-Gram similarly, **CSE-2** outperforms **TWE**. Although **TWE** aims at enhancing sentence representation by using topic model, neither parsing nor topic modeling would work well because short-texts lack enough signals for inference. Whats more, sentence embeedings are generated by simple aggregating over all topical word embeddings of each word in this sentence in **TWE**, which limits its capability of semantic representation.

Overall, nearly all the alternative algorithms perform worse on **Twitter**, especially **LDA** and **TWE**. This is mainly because that data in **Twitter** are more challenging for topic model as short-texts are noisy, sparse, and ambiguous. Although the training on larger corpus, i.e., way (i), contributes greatly to improving the performance of these topic-model based algorithms, they only have similar performance to **CSE-1** and could not transcend the attention-based variants. Certainly, we could also train **TWE** (even **LDA**) on a very larger corpus, and could expect a letter better results. However, training latent topic model on very large dataset is very slow, although many fast algorithms of topic models are available (Smola and Narayanamurthy, 2010; Ahmed et al., 2012). Whats more, from the complexity analysis, we could conclude that, compared with **PV**, **CSE** only need a little more space to store look-ups matrix **D**

and **R**; while compared with **CSE** and **PV**, **TWE** require more parameters to store more discriminative information for word embedding.

4.5 Information Retrieval

The information retrieval task is also utilized to evaluate the proposed models, and we want to examine whether a sentence should be retrieved given a query. Specially, we mainly focus on short-text retrieval by utilizing official tweet collection **Tweet11**, which is the benchmark dataset for microblog retrieval. We index all tweets in this collection by using Indri toolkit, and then perform a general relevance-pseudo feedback procedure, as follows: (i) Given a query, we firstly obtain associated tweets, which are before query issue time, via preliminary retrieval as feedback tweets. (ii) We generate the sentence representation vector of both original query and these feedback tweets by the alternative algorithms above. (iii) With efforts above, we compute cosine scores between query vector and each tweet vector to measure the semantic similarity between the query and candidate tweets, and then re-rank the feedback tweets with descending cosine scores.

We utilize the official metric for the TREC Microblog track, i.e., Precision at 30 (P@30), and Mean Average Precision (MAP), for evaluating the ranking performance of different algorithms. Experimental results for this task are shown in Table 2. Besides, we also operate a query-by-query analysis and conduct t-test to demonstrate the improvements on both metrics are statistically significant. In Table 2, the superscript α and β respectively denote statistically significant improvements over **TWE** and **PV-DM** ($p < 0.05$).

As shown in Table 2, the **CSE-2** significantly outperforms all these models, and exceeds the best baseline model (**TWE**) by 11.9% in MAP and 4.5% in P@30, which is a statistically significant improvement. Without regard to attention-based model firstly, such an improvement comes from the **CSE-2**'s ability to embed the contextual and semantic information of the sentences into a finite dimension vector. Topic model based algorithms (e.g., **LDA** and **TWE**) suffer extremely from the sparsity and noise of tweet collection. For the twitter data, since we are not able to find appropriate long texts, latent topic models are not performed.

We could observe that attention-based CSE model (**aCSE-1** and **aCSE-2**) improves over original CSE model (**CSE-1** and **CSE-2**). However, attention model promotes **CSE-1** significantly, while **aCSE-2** obtain similar results compared to **CSE-2**, indicating that attention model leads to small improvement for Skip-Gram based CSE model. We argue that it is because Skip-Gram itself gives less weight to the distant words by sampling less from those words, which is essentially similar to attention model somehow.

	TMB2011		TMB2012	
Model	**MAP**	**P@30**	**MAP**	**P@30**
BOW	0.304	0.412	0.321	0.494
LDA	0.281	0.409	0.311	0.486
PV-DBOW	0.285	0.412	0.324	0.491
PV-DM	0.327	0.431	0.340	0.524
TWE	0.331	0.446^{β}	0.347^{β}	0.511
CSE-1	0.337	0.451	0.344	0.512
CSE-2	**0.367**	**0.461**	**0.360**	**0.517**
aCSE-1	0.342	0.459	0.351	0.516
aCSE-2	$\mathbf{0.370}^{\alpha\beta}$	$\mathbf{0.464}^{\alpha\beta}$	$\mathbf{0.364}^{\alpha\beta}$	$\mathbf{0.522}^{\alpha\beta}$

Table 2: Results of information retrieval.

5 Conclusion

By inducing concept information, the proposed conceptual sentence embedding maintains and enhances the semantic information of sentence embedding. Furthermore, we extend the proposed models by introducing attention model, which allows it to consider contextual words within the window in a non-uniform way while maintaining the efficiency. We compare them with different algorithms, including bag-of-word models, topic model-based model and other state-of-the-art sentence embedding models. The experimental results demonstrate that the proposed method performs the best and shows improvement over the compared methods, especially for short-texts.

Acknowledgments

The work was supported by National Natural Science Foundation of China (Grant No. 61132009), National Basic Research Program of China (973 Program, Grant No. 2013CB329303), and National Hi-Tech Research & Development Program (863 Pro-gram, Grant No. 2015AA015404).

References

Amr Ahmed, Moahmed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander J. Smola. 2012. Scalable inference in latent variable models. In *International Conference on Web Search and Web Data Mining, WSDM 2012, Seattle, Wa, Usa, February*, pages 123–132.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Eprint Arxiv*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Meeting of the Association for Computational Linguistics*, pages 809–815.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Rongen Fan, Kaiwei Chang, Cho Jui Hsieh, Xiangrui Wang, and Chih Jen Lin. 2010. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9(12):1871–1874.

Zellig S. Harris. 1970. *Distributional Structure*. Springer Netherlands.

Posen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *ACM International Conference on Conference on Information and Knowledge Management*, pages 2333–2338.

Quoc V. Le and Tomas. Mikolov. 2014. Distributed representations of sentences and documents. *Eprint Arxiv*, 4:1188–1196.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26:3111–3119.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. *Aistats*.

Iadh Ounis, Craig MacDonald, Jimmy Lin, and Ian Soboroff. 2011. Overview of the trec-2011 microblog track.

H Palangi, L Deng, Y Shen, J Gao, X He, J Chen, X Song, and R Ward. 2015. Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. *Arxiv*, 24(4):694–707.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Gerard Salton and Michael J. Mcgill. 1986. *Introduction to modern information retrieval*. McGraw-Hill,.

Aliaksei Severyn, Alessandro Moschitti, Manos Tsagkias, Richard Berendsen, and Maarten De Rijke. 2014. A syntax-aware re-ranker for microblog retrieval. In *SIGIR*, pages 1067–1070.

Alexander Smola and Shravan Narayanamurthy. 2010. An architecture for parallel topic models. *Proceedings of the Vldb Endowment*, 3(1):703–710.

Ian Soboroff, Iadh Ounis, Craig MacDonald, and Jimmy Lin. 2012. Overview of the trec-2012 microblog track. In *TREC*.

Yangqiu Song, Haixun Wang, Zhongyuan Wang, Hongsong Li, and Weizhu Chen. 2011. Short text conceptualization using a probabilistic knowledgebase. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, pages 2330–2336.

Yangqiu Song, Shusen Wang, and Haixun Wang. 2015. Open domain short text conceptualization: a generative + descriptive modeling approach. In *Proceedings of the 24th International Conference on Artificial Intelligence*.

Ling Wang, Tsvetkov Yulia, Amir Silvio, Fermandez Ramon, Dyer Chris, Black Alan W, Trancoso Isabel, and Lin Chu-Cheng. 2015a. Not all contexts are created equal: Better word representations with variable attention. In *Conference on Empirical Methods in Natural Language Processing*, pages 1367–1372.

Mingxuan Wang, Zhengdong Lu, Hang Li, and Qun Liu. 2015b. Syntax-based deep matching of short texts. *Computer Science*.

Zhongyuan Wang, Kejun Zhao, Haixun Wang, Xiaofeng Meng, and Ji-Rong Wen. 2015c. Query understanding through knowledge-based conceptualization. In *Proceedings of the 24th International Conference on Artificial Intelligence*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Towards universal paraphrastic sentence embeddings. *Computer Science*.

Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. 2012. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 481–492.

DocChat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents

Zhao Yan[†] [*], **Nan Duan**[‡] , **Junwei Bao**[+] , **Peng Chen**[§] , **Ming Zhou**[‡] ,
Zhoujun Li[†] , **Jianshe Zhou**[¶]

[†]State Key Laboratory of Software Development Environment, Beihang University
[‡]Microsoft Research [+]Harbin Institute of Technology
[§]Microsoft Search Technology Center [¶]BAICIT, Capital Normal University
[†]{yanzhao, lizj}@buaa.edu.cn [+]baojunwei001@gmail.com
[‡§]{nanduan, peche, mingzhou}@microsoft.com [¶]zhoujs@cnu.edu.cn

Abstract

Most current chatbot engines are designed to reply to user utterances based on existing utterance-response (or Q-R)[1] pairs. In this paper, we present DocChat, a novel information retrieval approach for chatbot engines that can leverage unstructured documents, instead of Q-R pairs, to respond to utterances. A learning to rank model with features designed at different levels of granularity is proposed to measure the relevance between utterances and responses directly. We evaluate our proposed approach in both English and Chinese: (i) For English, we evaluate DocChat on WikiQA and QASent, two answer sentence selection tasks, and compare it with state-of-the-art methods. Reasonable improvements and good adaptability are observed. (ii) For Chinese, we compare DocChat with XiaoIce[2], a famous chitchat engine in China, and side-by-side evaluation shows that DocChat is a perfect complement for chatbot engines using Q-R pairs as main source of responses.

1 Introduction

Building chatbot engines that can interact with humans with natural language is one of the most challenging problems in artificial intelligence. Along with the explosive growth of social media, like community question answering (CQA) websites (e.g., Yahoo Answers and WikiAnswers) and social media websites (e.g., Twitter and Weibo), the amount of utterance-response (or Q-R) pairs has experienced massive growth in recent years, and such a corpus greatly promotes the emergence of various data-driven chatbot approaches.

Instead of multiple rounds of conversation, we only consider a much simplified task, short text conversation (STC) in which the response $\mathcal{R}$ is a short text and only depends on the last user utterance $\mathcal{Q}$. Previous methods for the STC task mostly rely on Q-R pairs and fall into two categories: **Retrieval-based methods** (e.g., Ji et al., 2014). This type of methods first retrieve the most possible $\langle \hat{\mathcal{Q}}, \hat{\mathcal{R}} \rangle$ pair from a set of existing Q-R pairs, which best matches current utterance $\mathcal{Q}$ based on semantic matching models, then take $\hat{\mathcal{R}}$ as the response $\mathcal{R}$. One disadvantage of such a method is that, for many specific domains, collecting such Q-R pairs is intractable. **Generation based methods** (e.g., Shang et al., 2015). This type of methods usually uses an encoder-decoder framework which first encode $\mathcal{Q}$ as a vector representation, then feed this representation to decoder to generate response $\mathcal{R}$. Similar to retrieval-based methods, such approaches also depend on existing Q-R pairs as training data. Like other language generation tasks, such as machine translation and paraphrasing, the fluency and naturality of machine generated text is another drawback.

To overcome the issues mentioned above, we present a novel response retrieval approach, **DocChat**, to find responses based on unstructured documents. For each user utterance, instead of looking for the best Q-R pair or generating a word sequence based on language generation techniques, our method selects a sentence from given documents directly, by ranking all possible sentences based on features designed at different levels of granularity. On one hand, using documents rather than Q-R pairs greatly improve the adapt-

[*]Contribution during internship at Microsoft Research.

[1]For convenience sake, we denote all utterance-response pairs (either QA pairs or conversational exchanges from social media websites like Twitter) as Q-R pairs in this paper.

[2]http://www.msxiaoice.com

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 516–525,
Berlin, Germany, August 7-12, 2016.

ability of chatbot engines on different chatting topics. On the other hand, all responses come from existing documents, which guarantees their fluency and naturality. We also show promising results in experiments, on both QA and chatbot scenarios.

2 Task Description

Formally, given an utterance $\mathcal{Q}$ and a document set $\mathcal{D}$, the document-based chatbot engine retrieves response $\mathcal{R}$ based on the following three steps:

- *response retrieval*, which retrieves response candidates $\mathcal{C}$ from $\mathcal{D}$ based on $\mathcal{Q}$:

$$\mathcal{C} = Retrieve(\mathcal{Q}, \mathcal{D})$$

  Each $\mathcal{S} \in \mathcal{C}$ is a sentence existing in $\mathcal{D}$.

- *response ranking*, which ranks all response candidates in $\mathcal{C}$ and selects the most possible response candidate as $\hat{\mathcal{S}}$:

$$\hat{\mathcal{S}} = \arg\max_{\mathcal{S} \in \mathcal{C}} Rank(\mathcal{S}, \mathcal{Q})$$

- *response triggering*, which decides whether it is confident enough to response $\mathcal{Q}$ using $\hat{S}$:

$$\mathcal{I} = Trigger(\hat{\mathcal{S}}, \mathcal{Q})$$

  where $\mathcal{I}$ is a binary value. When $\mathcal{I}$ equals to $true$, let the response $\mathcal{R} = \hat{\mathcal{S}}$ and output $\mathcal{R}$; otherwise, output nothing.

In the following three sections, we will describe solutions of these three components one by one.

3 Response Retrieval

Given a user utterance $\mathcal{Q}$, the goal of response retrieval is to efficiently find a small number of sentences from $\mathcal{D}$, which have high possibility to contain suitable sentences as $\mathcal{Q}$'s response. Although it is not necessarily true that a good response always shares more words with a given utterance, this measurement is still helpful in finding possible response candidates (Ji et al., 2014).

In this paper, the BM25 term weighting formulas (Jones et al., 2000) is used to retrieve response candidates from documents. Given each document $\mathcal{D}_k \in \mathcal{D}$, we collect a set of sentence triples $\langle \mathcal{S}^{prev}, \mathcal{S}, \mathcal{S}^{next} \rangle$ from $\mathcal{D}_k$, where $\mathcal{S}$ denotes a sentence in $\mathcal{D}_k$, $\mathcal{S}^{prev}$ and $\mathcal{S}^{next}$ denote $\mathcal{S}$'s previous sentence and next sentence respectively. Two special tags, $\langle BOD \rangle$ and $\langle EOD \rangle$, are added at the beginning and end of each passage, to make sure that such sentence triples can be extracted for every sentence in the document. The reason for indexing each sentence together with its context sentences is intuitive: If a sentence within a document can respond to an utterance, then its context should be revelent to the utterance as well.

4 Response Ranking

Given a user utterance $\mathcal{Q}$ and a response candidate $\mathcal{S}$, the ranking function $Rank(\mathcal{S}, \mathcal{Q})$ is designed as an ensemble of individual matching features:

$$Rank(\mathcal{S}, \mathcal{Q}) = \sum_k \lambda_k \cdot h_k(\mathcal{S}, \mathcal{Q})$$

where $h_k(\cdot)$ denotes the k-th feature function, λ_k denotes $h_k(\cdot)$'s corresponding weight.

We design features at different levels of granularity to measure the relevance between S and Q, including *word-level*, *phrase-level*, *sentence-level*, *document-level*, *relation-level*, *type-level* and *topic-level*, which will be introduced below.

4.1 Word-level Feature

We define three word-level features in this work: (1) $h_{WM}(\mathcal{S}, \mathcal{Q})$ denotes a word matching feature that counts the number (weighted by the *IDF* value of each word in $\mathcal{S}$) of non-stopwords shared by $\mathcal{S}$ and $\mathcal{Q}$. (2) $h_{W2W}(\mathcal{S}, \mathcal{Q})$ denotes a word-to-word translation-based feature that calculates the IBM model 1 score (Brown et al., 1993) of $\mathcal{S}$ and $\mathcal{Q}$ based on word alignments trained on 'question-related question' pairs using GIZA++ (Och and Ney, 2003). (3) $h_{W2V}(\mathcal{S}, \mathcal{Q})$ denotes a word embedding-based feature that calculates the average cosine distance between word embeddings of all non-stopword pairs $\langle v_{\mathcal{S}_j}, v_{\mathcal{Q}_i} \rangle$. $v_{\mathcal{S}_j}$ represent the word vector of j^{th} word in $\mathcal{S}$ and $v_{\mathcal{Q}_j}$ represent the word vector of i^{th} word in $\mathcal{Q}$.

4.2 Phrase-level Feature

4.2.1 Paraphrase

We first describe how to extract phrase-level paraphrases from an existing SMT (statistical machine translation) phrase table.

$PT = \{\langle s_i, t_i, p(t_i|s_i), p(s_i|t_i) \rangle\}$[3] is a phrase table, which is extracted from a bilingual corpus, where s_i (or t_i) denotes a phrase, in source

[3]We omit lexical weights that are commonly used in phrase tables, as they are not useful in paraphrase extraction.

(or target) language, $p(t_i|s_i)$ (or $p(s_i|t_i)$) denotes the translation probability from s_i (or t_i) to t_i (or s_i). We follow Bannard and Callison-Burch (2005) to extract a paraphrase table $PP = \{\langle s_i, s_j, score(s_j; s_i)\rangle\}$. s_i and s_j denote two phrases in source language, $score(s_j; s_i)$ denotes a confidence score that s_i can be paraphrased to s_j, which is computed based on PT:

$$score(s_j; s_i) = \sum_t \{p(t|s_i) \cdot p(s_j|t)\}$$

The underlying idea of this approach is that, two source phrases that are aligned to the same target phrase trend to be paraphrased.

We then define a paraphrase-based feature as:

$$h_{PP}(\mathcal{S}, \mathcal{Q}) = \frac{\sum_{n=1}^{N} \frac{\sum_{j=0}^{|\mathcal{S}|-n} Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})}{|\mathcal{S}|-n+1}}{N}$$

where $\mathcal{S}_j^{j+n-1}$ denotes the consecutive word sequence (or phrase) in $\mathcal{S}$, which starts from $\mathcal{S}_j$ and ends with $\mathcal{S}_{j+n-1}$, N denotes the maximum n-gram order (here is 3). $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})$ is computed based on the following rules:

- If $\mathcal{S}_j^{j+n-1} \in \mathcal{Q}$, then $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 1$;
- Else, if $\langle \mathcal{S}_j^{j+n-1}, s, score(s; \mathcal{S}_j^{j+n-1})\rangle \in PP$ and $\mathcal{S}_j^{j+n-1}$'s paraphrase s occurs in $\mathcal{Q}$, then $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = score(s; \mathcal{S}_j^{j+n-1})$
- Else, $Count_{PP}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 0$.

4.2.2 Phrase-to-Phrase Translation

Similar to $h_{PP}(\mathcal{S}, \mathcal{Q})$, a phrase translation-based feature based on a phrase table PT is defined as:

$$h_{PT}(\mathcal{S}, \mathcal{Q}) = \frac{\sum_{n=1}^{N} \frac{\sum_{j=0}^{|\mathcal{S}|-n} Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})}{|\mathcal{S}|-n+1}}{N}$$

where $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q})$ is computed based on the following rules:

- If $\mathcal{S}_j^{j+n-1} \in \mathcal{Q}$, then $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 1$;
- Else, if $\langle \mathcal{S}_j^{j+n-1}, s, p(\mathcal{S}_j^{j+n-1}|s), p(s|\mathcal{S}_j^{j+n-1})\rangle \in PT$ and $\mathcal{S}_j^{j+n-1}$'s translation $s \in \mathcal{Q}$, then $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = p(\mathcal{S}_j^{j+n-1}|s) \cdot p(s|\mathcal{S}_j^{j+n-1})$
- Else, $Count_{PT}(\mathcal{S}_j^{j+n-1}, \mathcal{Q}) = 0$

We train a phrase table based on 'question-answer' pairs crawled from community QA websites.

4.3 Sentence-level Feature

We first present an attention-based sentence embedding method based on a convolution neural network (CNN), whose input is a sentence pair and output is a sentence embedding pair. Two features will be introduced in Section 4.3.1 and 4.3.2, which are designed based on two sentence embedding models trained using different types of data.

In the input layer, given a sentence pair $\langle \mathcal{S}_X, \mathcal{S}_Y\rangle$, an attention matrix $\mathcal{A} \in R^{|\mathcal{S}_X|\times|\mathcal{S}_Y|}$ is generated based on pre-trained word embeddings of $\mathcal{S}_X$ and $\mathcal{S}_Y$, where each element $\mathcal{A}_{i,j} \in \mathcal{A}$ is computed as:

$$\mathcal{A}_{i,j} = cosine(v_i^{\mathcal{S}_X}, v_j^{\mathcal{S}_Y})$$

where $v_i^{\mathcal{S}_X}$ (or $v_j^{\mathcal{S}_Y}$) denotes the embedding vector of the i^{th} (or j^{th}) word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$).

Then, column-wise and row-wise max-pooling are applied to $\mathcal{A}$ to generate two attention vectors $V^{\mathcal{S}_X} \in R^{|\mathcal{S}_X|}$ and $V^{\mathcal{S}_Y} \in R^{|\mathcal{S}_Y|}$, where the k^{th} elements of $V^{\mathcal{S}_X}$ and $V^{\mathcal{S}_Y}$ are computed as:

$$V_k^{\mathcal{S}_X} = \max_{1<l<|\mathcal{S}_Y|} \{\mathcal{A}_{k,l}\} \; and \; V_k^{\mathcal{S}_Y} = \max_{1<l<|\mathcal{S}_X|} \{\mathcal{A}_{l,k}\}$$

$V_k^{\mathcal{S}_X}$ (or $V_k^{\mathcal{S}_Y}$) can be interpreted as the *attention score* of the k^{th} word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$) with regard to all words in $\mathcal{S}_Y$ (or $\mathcal{S}_X$).

Next, two *attention distributions* $D^{\mathcal{S}_X} \in R^{|\mathcal{S}_X|}$ and $D^{\mathcal{S}_Y} \in R^{|\mathcal{S}_Y|}$ are generated for $\mathcal{S}_X$ and $\mathcal{S}_Y$ based on $V^{\mathcal{S}_X}$ and $V^{\mathcal{S}_Y}$ respectively, where the k^{th} elements of $D^{\mathcal{S}_X}$ and $D^{\mathcal{S}_Y}$ are computed as:

$$D_k^{\mathcal{S}_X} = \frac{e^{V_k^{\mathcal{S}_X}}}{\sum_{l=1}^{|\mathcal{S}_X|} e^{V_l^{\mathcal{S}_X}}} \; and \; D_k^{\mathcal{S}_\mathcal{Y}} = \frac{e^{V_k^{\mathcal{S}_Y}}}{\sum_{l=1}^{|\mathcal{S}_Y|} e^{V_l^{\mathcal{S}_Y}}}$$

$D_k^{\mathcal{S}_\mathcal{X}}$ (or $D_k^{\mathcal{S}_\mathcal{Y}}$) can be interpreted as the *normalized attention score* of the k^{th} word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$) with regard to all words in $\mathcal{S}_Y$ (or $\mathcal{S}_X$).

Last, we update each pre-trained word embedding $v_k^{\mathcal{S}_X}$ (or $v_k^{\mathcal{S}_Y}$) to $\hat{v}_k^{\mathcal{S}_X}$ (or $\hat{v}_k^{\mathcal{S}_Y}$), by multiplying every value in $v_k^{\mathcal{S}_X}$ (or $v_k^{\mathcal{S}_Y}$) with $D_k^{\mathcal{S}_X}$ (or $D_k^{\mathcal{S}_Y}$). The underlying intuition of updating pre-trained word embeddings is to *re-weight the importance of each word in $\mathcal{S}_X$ (or $\mathcal{S}_Y$) based on $\mathcal{S}_Y$ (or $\mathcal{S}_X$), instead of treating them in an equal manner.*

In the convolution layer, we first derive an input matrix $Z_{\mathcal{S}_X} = \{l_1, ..., l_{|\mathcal{S}_X|}\}$, where l_t is the concatenation of a sequence of $m = 2d-1$[4] updated word embeddings $[\hat{v}_{t-d}^{\mathcal{S}_X}, ..., \hat{v}_t^{\mathcal{S}_X}, ..., \hat{v}_{t+d}^{\mathcal{S}_X}]$, centralized in the t^{th} word in $\mathcal{S}_X$. Then, the convo-

[4] In this paper, m is set to 3.

lution layer performs sliding window-based feature extraction to project each vector representation $l_t \in Z_{\mathcal{S}_X}$ to a contextual feature vector $h_t^{\mathcal{S}_X}$:

$$h_t^{\mathcal{S}_X} = tanh(W_c \cdot l_t)$$

where W_c is the convolution matrix, $tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$ is the activation function. The same operation is performed to $\mathcal{S}_Y$ as well.

In the pooling layer, we aggregate local features extracted by the convolution layer from $\mathcal{S}_X$, and form a sentence-level global feature vector with a fixed size independent of the length of the input sentence. Here, *max-pooling* is used to force the network to retain the most useful local features by $l_p^{\mathcal{S}_X} = [v_1^{\mathcal{S}_X}, ..., v_K^{\mathcal{S}_X}]$, where:

$$v_i^{\mathcal{S}_X} = \max_{t=1,...,|\mathcal{S}_X|} \{h_t^{\mathcal{S}_X}(i)\}$$

$h_t^{\mathcal{S}_X}(i)$ denotes the i^{th} value in the vector $h_t^{\mathcal{S}_X}$. The same operation are performed to $\mathcal{S}_Y$ as well.

In the output layer, one more non-linear transformation is applied to $l_p^{\mathcal{S}_X}$:

$$y(\mathcal{S}_X) = tanh(W_s \cdot l_p^{\mathcal{S}_X})$$

W_s is the semantic projection matrix, $y(\mathcal{S}_X)$ is the final sentence embedding of $\mathcal{S}_X$. The same operation is performed to $\mathcal{S}_Y$ to obtain $y(\mathcal{S}_Y)$.

We train model parameters W_c and W_s by minimizing the following ranking loss function:

$$L = \max\{0, M - cosine(y(\mathcal{S}_X), y(\mathcal{S}_Y)) + cosine(y(\mathcal{S}_X), y(\mathcal{S}_Y^-))\}$$

where M is a constant, $\mathcal{S}_Y^-$ is a negative instance.

4.3.1 Causality Relationship Modeling

We train the first attention-based sentence embedding model based on a set of 'question-answer' pairs as input sentence pairs, and then design a causality relationship-based feature as:

$$h_{SCR}(\mathcal{S}, \mathcal{Q}) = cosine(y_{SCR}(\mathcal{S}), y_{SCR}(\mathcal{Q}))$$

$y_{SCR}(\mathcal{S})$ and $y_{SCR}(\mathcal{Q})$ denote the sentence embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively. We expect this feature captures the *causality relationship* between questions and their corresponding answers, and works on *question-like utterances*.

4.3.2 Discourse Relationship Modeling

We train the second attention-based sentence embedding model based on a set of 'sentence-next sentence' pairs as input sentence pairs, and then design a discourse relationship-based feature as:

$$h_{SDR}(\mathcal{S}, \mathcal{Q}) = cosine(y_{SDR}(\mathcal{S}), y_{SDR}(\mathcal{Q}))$$

$y_{SDR}(\mathcal{S})$ and $y_{SDR}(\mathcal{Q})$ denote the sentence embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively. We expect this feature learns and captures the *discourse relationship* between sentences and their next sentences, and works on *statement-like utterances*. Here, a large number of 'sentence-next sentence' pairs can be easily obtained from documents.

4.4 Document-level Feature

We take document-level information into consideration to measure the semantic similarity between $\mathcal{Q}$ and $\mathcal{S}$, and define two context features as:

$$h_{DM}(\mathcal{S}^*, \mathcal{Q}) = cosine(y_{SCR}(\mathcal{S}^*), y_{SCR}(\mathcal{Q}))$$

where $\mathcal{S}^*$ can be $\mathcal{S}^{prev}$ and $\mathcal{S}^{next}$ that denote previous and next sentences of $\mathcal{S}$ in the original document. The sentence embedding model trained based on 'question-answer' pairs (in Section 4.3.1) is directly used to generate context embeddings for $h_{DM}(\mathcal{S}^{prev}, \mathcal{Q})$ and $h_{DM}(\mathcal{S}^{next}, \mathcal{Q})$. So no further training data is needed for this feature.

4.5 Relation-level Feature

Given a structured knowledge base, such as Freebase[5], a single relation question $\mathcal{Q}$ (in natural language) with its answer can be first parsed into a fact formatted as $\langle e_{sbj}, rel, e_{obj} \rangle$, where e_{sbj} denotes a subject entity detected from the question, rel denotes the relationship expressed by the question, e_{obj} denotes an object entity found from the knowledge base based on e_{sbj} and rel. Then we can get $\langle \mathcal{Q}, rel \rangle$ pairs. This rel can help for modeling semantic relationships between $\mathcal{Q}$ and $\mathcal{R}$. For example, the Q-A pair $\langle$*What does Jimmy Neutron do? – inventor*$\rangle$ can be parsed into $\langle$*Jimmy Neutron, fictional_character_occupation, inventor*$\rangle$ where the rel is *fictional_character_occupation*.

Similar to Yih et al. (2014), We use $\langle \mathcal{Q}, rel \rangle$ pairs as training data, and learn a *rel*-CNN model, which can encode each question $\mathcal{Q}$ (or each relation rel) into a relation embedding. For a given question $\mathcal{Q}$, the corresponding relation rel^+ is

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

treated as a positive example, and randomly selected other relations are used as negative examples rel^-. The posterior probability of rel^+ given $\mathcal{Q}$ is computed as:

$$P(rel^+|\mathcal{Q}) = \frac{e^{cosine(y(rel^+),y(\mathcal{Q}))}}{\sum_{rel^-} e^{cosine(y(rel^-),y(\mathcal{Q}))}}$$

$y(rel)$ and $y(\mathcal{Q})$ denote relation embeddings of rel and $\mathcal{Q}$ based on *rel*-CNN. *rel*-CNN is trained by maximizing the log-posterior.

We then define a relation-based feature as:

$$h_{RE}(\mathcal{S}, \mathcal{Q}) = cosine(y_{RE}(\mathcal{Q}), y_{RE}(\mathcal{S}))$$

$y_{RE}(\mathcal{S})$ and $y_{RE}(\mathcal{Q})$ denote relation embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively, coming from *rel*-CNN.

4.6 Type-level Feature

We extend each $\langle \mathcal{Q}, e_{sbj}, rel, e_{obj} \rangle$ in the SimpleQuestions data set to $\langle \mathcal{Q}, e_{sbj}, rel, e_{obj}, type \rangle$, where $type$ denotes the type name of e_{obj} based on Freebase. Thus, we obtain $\langle \mathcal{Q}, type \rangle$ pairs.

Similar to *rel*-CNN, we use $\langle \mathcal{Q}, type \rangle$ pairs to train another CNN model, denoted as *type*-CNN. Based on which, we define a type-based feature as:

$$h_{TE}(\mathcal{S}, \mathcal{Q}) = cosine(y_{TE}(\mathcal{Q}), y_{TE}(\mathcal{S}))$$

$y_{TE}(\mathcal{S})$ and $y_{TE}(\mathcal{Q})$ denote type embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively, coming from *type*-CNN.

4.7 Topic-level Feature

4.7.1 Unsupervised Topic Model

As the assumption that Q-R pair should share similar topic distribution, We define an unsupervised topic model-based feature h_{UTM} as the average cosine distance between topic vectors of all non-stopword pairs $\langle v_{\mathcal{S}_j}, v_{\mathcal{Q}_i} \rangle$, where $v_w = [p(t_1|w), ..., p(t_N|w)]^T$ denotes the topic vector of a given word w. Given a corpus, various topic modeling methods, such as pLSI (probabilistic latent semantic indexing) and LDA (latent Dirichlet allocation), can be used to estimate $p(t_i|w)$, which denotes the probability that w belongs to a topic t_i.

4.7.2 Supervised Topic Model

One shortcoming of the unsupervised topic model is that, the topic size is pre-defined, which might not reflect the truth on a specific corpus. In this paper, we explore a supervised topic model approach as well, based on 'sentence-topic' pairs.

We crawl a large number of $\langle \mathcal{S}, topic \rangle$ pairs from Wikipedia documents, where $\mathcal{S}$ denotes a sentence, $topic$ denotes the content name of the section that $\mathcal{S}$ extracted from. Such content names are labeled by Wikipedia article editors, and can be found in the *Contents* fields.

Similar to *rel*-CNN and *type*-CNN, we use the $\langle \mathcal{S}, topic \rangle$ pairs to train another CNN model, denoted as *topic*-CNN. Based on which, we define a supervised topic model-based feature as:

$$h_{STM}(\mathcal{S}, \mathcal{Q}) = cosine(y_{STM}(\mathcal{S}), y_{STM}(\mathcal{Q}))$$

$y_{STM}(\mathcal{S})$ and $y_{STM}(\mathcal{Q})$ denote topic embeddings of $\mathcal{S}$ and $\mathcal{Q}$ respectively, coming from *topic*-CNN.

4.8 Learning to Ranking Model

We employ a regression-based learning to rank method (Nallapati, 2004) to train response ranking model, based on a set of labeled $\langle \mathcal{Q}, \mathcal{C} \rangle$ pairs, Feature weights in the ranking model are trained by SGD based on the training data that consists of a set of $\langle \mathcal{Q}, \mathcal{C} \rangle$ pairs, where $\mathcal{Q}$ denotes a user utterance and $\mathcal{C}$ denotes a set of response candidates. Each candidate $\mathcal{S}$ in $\mathcal{C}$ is labeled by $+$ or $-$, which indicates whether $\mathcal{S}$ is a suitable response of $\mathcal{Q}$ $(+)$, or not $(-)$.

As manually labeled data, such as WikiQA (Yang et al., 2015), needs expensive human annotation effort, we propose an automatic way to collect training data. First, 'question-answer' (or Q-A) pairs $\{Q_i, A_i\}_{i=1}^M$ are crawled from community QA websites. Q_i denotes a question. A_i denotes Q_i's answer, which includes one or more sentences $A_i = \{s_1, ..., s_K\}$. Then, we index answer sentences of all questions. Next, for each question Q_i, we run response retrieval to obtain answer sentence candidates $\mathcal{C}_i = \{s_1^{'}, ..., s_N^{'}\}$. Last, if we know the correct answer sentences of each question Q_i, we can then label each candidate in $\mathcal{C}_i$ as $+$ or $-$. In experiments, manually labeled data (WikiQA) is used in open domain question answering scenario, and automatically generated data is used in chatbot scenario.

5 Response Triggering

There are two types of utterances, *chit-chat utterances* and *informative utterances*. The former should be handled by chit-chat engines, and the latter is more suitable to our work, as documents usually contain formal and informative contents. Thus, we have to respond to informative utterances only. Response retrieval cannot always guarantee to return a candidate set that contains

at least one suitable response, but response ranking will output the best possible candidate all the time. So, we have to decide which responses are confident enough to be output, and which are not.

In this paper, we define *response triggering* as a function that decides whether a response candidate $\mathcal{S}$ has enough confidence to be output:

$$\begin{aligned}\mathcal{I} &= Trigger(\mathcal{S}, \mathcal{Q}) \\ &= \mathcal{I}_U(\mathcal{Q}) \wedge \mathcal{I}_{Rank}(\mathcal{S}, \mathcal{Q}) \wedge \mathcal{I}_R(\mathcal{S})\end{aligned}$$

where $Trigger(\mathcal{Q}, \mathcal{S})$ returns $true$, if and only if all its three sub-functions return $true$.

$\mathcal{I}_U(\mathcal{Q})$ returns $true$, if $\mathcal{Q}$ is an informative query. We collect and label chit-chat queries based on conversational exchanges from social media websites to train the classifier.

$\mathcal{I}_{Rank}(\mathcal{S}, \mathcal{Q})$ returns $true$, if the score $s(\mathcal{S}, \mathcal{Q})$ exceeds an empirical threshold τ:

$$s(\mathcal{S}, \mathcal{Q}) = \frac{1}{1 + e^{-\alpha \cdot Rank(\mathcal{S}, \mathcal{Q})}}$$

where α is the scaling factor that controls the distribution of $s(\cdot)$ smooth or sharp. Both α and τ are selected based on a separated development set.

$\mathcal{I}_R(\mathcal{S})$ returns $true$, if (i) the length of $\mathcal{S}$ is less than a pre-defined threshold, and (ii) $\mathcal{S}$ does not start with a phrase that expresses a progressive relation, such as *but also*, *besides*, *moreover* and etc., as the contents of sentences starting with such phrases usually depend on their context sentences, and they are not suitable for responses.

6 Related Work

For modeling dialogue. Previous works mainly focused on rule-based or learning-based approaches (Litman et al., 2000; Schatzmann et al., 2006; Williams and Young, 2007). These methods require efforts on designing rules or labeling data for training, which suffer the coverage issue.

For short text conversation. With the fast development of social media, such as microblog and CQA services, large scale conversation data and data-driven approaches become possible. Ritter et al. (2011) proposed an SMT based method, which treats response generation as a machine translation task. Shang et al. (2015) presented an RNN based method, which is trained based on a large number of single round conversation data. Grammatical and fluency problems are the biggest issue for such generation-based approaches. Retrieval-based methods selects the most suitable response to the current utterance from the large number of Q-R pairs. Ji et al. (2014) built a conversation system using learning to rank and semantic matching techniques. However, collecting enough Q-R pairs to build chatbots is often intractable for many domains. Compared to previous methods, DocChat learns internal relationships between utterances and responses based on statistical models at different levels of granularity, and relax the dependency on Q-R pairs as response sources. These make DocChat as a general response generation solution to chatbots, with high adaptation capability.

For answer sentence selection. Prior work in measuring the relevance between question and answer is mainly in word-level and syntactic-level (Wang and Manning, 2010; Heilman and Smith, 2010; Yih et al., 2013). Learning representation by neural network architecture (Yu et al., 2014; Wang and Nyberg, 2015; Severyn and Moschitti, 2015) has become a hot research topic to go beyond word-level or phrase-level methods. Compared to previous works we find that, (i) Large scale existing resources with noise have more advantages as training data. (ii) Knowledge-based semantic models can play important roles.

7 Experiments

7.1 Evaluation on QA (English)

Take into account response ranking task and answer selection task are similar, we first evaluate DocChat in a QA scenario as a simulation. Here, response ranking is treated as the *answer selection* task, and response triggering is treated as the *answer triggering* task.

7.1.1 Experiment Setup

We select *WikiQA*[6] as the evaluation data, as it is precisely constructed based on natural language questions and Wikipedia documents, which contains 2,118 'question-document' pairs in the training set, 296 'question-document' pairs in development set, and 633 'question-document' pairs in testing set. Each sentence in the document of a given question is labeled as 1 or 0, where 1 denotes the current sentence is a correct answer sentence, and 0 denotes the opposite meaning. Given a question, the task of WikiQA is to select answer sentences from all sentences in a question's corresponding document. The training data settings of response ranking features are described below.

[6] http://aka.ms/WikiQA

F_w denotes 3 word-level features, h_{WM}, h_{W2W} and h_{W2V}. For h_{W2W}, GIZA++ is used to train word alignments on 11.6M 'question-related question' pairs (Fader et al., 2013) crawled from WikiAnswers.[7]. For h_{W2V}, Word2Vec (Mikolov et al., 2013) is used to train word embedding on sentences from Wikipedia in English.

F_p denotes 2 phrase-level features, h_{PP} and h_{PT}. For h_{PP}, bilingual data[8] is used to extract a phrase-based translation table (Koehn et al., 2003), from which paraphrases are extracted (Section 4.2.1). For h_{PT}, GIZA++ trains word alignments on 4M 'question-answer' pairs[9] crawled from Yahoo Answers[10], and then a phrase table is extracted from word alignments using the *intersect-diag-grow* refinement.

F_s denotes 2 sentence-level features, h_{SCR} and h_{SDR}. For h_{SCR}, 4M 'question-answer' pairs (the same to h_{PT}) is used to train the CNN model. For h_{SDR}, we randomly select 0.5M 'sentence-next sentence' pairs from English Wikipedia.

F_d denotes document-level feature h_{DM}. Here, we didn't train a new model. Instead, we just reuse the CNN model used in h_{SCR}.

F_r and F_{ty} denote relation-level feature h_{RE} and type-level feature h_{TE}. Bordes et al. (2015) released the *SimpleQuestions* data set[11], which consists of 108,442 English questions. Each question (e.g., *What does Jimmy Neutron do?*) is written by human annotators based on a triple in Freebase which formatted as $\langle e_{sbj}, rel, e_{obj} \rangle$ (e.g., $\langle$*Jimmy Neutron, fictional_character_occupation, inventor*$\rangle$) Here, as described in Section 4.5 and 4.6, 'question-relation' pairs and 'question-type' pairs based upon *SimpleQuestions* data set are used to train h_{RE} and h_{TE}.

F_{to} denotes 2 topic-level features, h_{UTM} and h_{STM}. For h_{UTM}, we run LightLDA (Yuan et al., 2015) on sentences from English Wikipedia, where the topic is set to 1,000. For h_{STM}, 4M 'sentence-topic' pairs are extracted from English Wikipedia (Section 4.7.2), where the most frequent 25,000 content names are used as topics.

[7]http://wiki.answers.com

[8]We use 0.5M Chinese-English bilingual sentences in phrase table extraction, i.e., LDC2003E07, LDC2003E14, LDC2005T06, LDC2005T10, LDC2005E83, LDC2006E26, LDC2006E34, LDC2006E85 and LDC2006E92.

[9]For each question, we only select the first sentence in its answer to construct a 'question-answer' pair, as it contains more causality information than sentences in other positions.

[10]https://answers.yahoo.com

[11]https://research.facebook.com/research/-babi/

Features	MAP	MRR
F_w	60.25%	61.70%
F_p	61.31%	62.61%
F_s	61.99%	64.32%
F_d	59.15%	61.17%
F_r	46.95%	45.89%
F_{ty}	45.67%	43.37%
F_{to}	58.34%	59.96%

Table 1: Impacts of features at different levels.

#	Methods	MAP	MRR
(1)	Yih et al. (2013)	59.93%	60.68%
(2)	Yang et al. (2015)	65.20%	66.52%
(3)	Miao et al. (2015)	68.86%	70.69%
(4)	Yin et al. (2015)	69.21%	71.08%
(5)	DocChat	68.25%	70.73%
(6)	DocChat+(2)	**70.08%**	**72.22%**

Table 2: Evaluation of AS task on WikiQA.

7.1.2 Results on Answer Selection (AS)

The performance of answer selection is evaluated by *Mean Average Precision (MAP)* and *Mean Reciprocal Rank (MRR)*. Among all 'question-document' pairs in WikiQA, only one-third of documents contain answer sentences to their corresponding questions. Similar to previous work, questions without correct answers in the candidate sentences are not taken into account.

We first evaluate the impact of features at each level, and show results in Table 1. F_w, F_p, and F_s perform best among all features, which makes sense, as they can capture lexical features. F_r and F_{ty} perform not very good, but make sense, as the training data (i.e. SimpleQuestions) are based on Freebase instead of Wikipedia. Interestingly, we find that F_{to} and F_d can achieve comparable results as well. We think the reason is that, their training data come from Wikipedia, which fit the WikiQA task very well.

We evaluate the quality of DocChat on WikiQA, and show results in Table 2. The first four rows in Table 2 represent four baseline methods, including: (1) Yih et al. (2013), which makes use of rich lexical semantic features; (2) Yang et al. (2015), which uses a bi-gram CNN model with average pooling; (3) Miao et al. (2015), which uses an enriched LSTM with a latent stochastic attention mechanism to model similarity between Q-R pairs; and (4) Yin et al. (2015), which adds the attention mechanism to the CNN architecture.

Table 2 shows that, without using WikiQA's training set (only development set for ranking weights), DocChat can achieve comparable per-

Methods	MAP	MRR
CNN$_{WikiQA}$	0.6575	0.7534
CNN$_{QASent}$	**0.6951**	0.7633
DocChat	0.6896	**0.7688**

Table 3: Evaluation of AS on QASent.

formance with state-of-the-art baselines. Furthermore, by combining the CNN model proposed by Yang et al. (2015) and trained on WikiQA training set, we achieve the best result on both metrics.

Compared to previous methods, we think DocChat has the following two advantages: First, our feature models depending on existing resources are readily available (such as Q-Q pairs, Q-A pairs, 'sentence-next sentence' pairs, and etc.), instead of requiring manually annotated data (such as WikiQA and QASent). Training of the response ranking model does need labeled data, but the size demanded is acceptable. Second, as the training data used in our approach come from open domain resources, we can expect a high adaptation capability and comparable results on other WikiQA-like tasks, as our models are task-independent.

To verify the second advantage, we evaluate DocChat on another answer selection data set, QASent (Wang et al., 2007), and list results in Table 3. CNN_{WikiQA} and CNN_{QASent} refer to the results of Yang et al. (2015)'s method, where the CNN models are trained on WikiQA's training set and QASent's training set respectively. All these three methods train feature weights using QASent's development set. Table 3 tells, DocChat outperforms CNN_{WikiQA} in terms of MAP and MRR, and achieves comparable results compared to CNN_{QASent}. The comparisons results show a good adaptation capability of DocChat.

Table 4 evaluates the contributions of features at different levels of granularity. To highlight the differences, we report the percent deviation by removing different features at the same level from DocChat. From Table 4 we can see that, 1) Each feature group is indispensable to DocChat; 2) Features at sentence-level are most important than other feature groups; 3) Compared to results in Table 1, combining all features can significantly promote the performance.

7.1.3 Evaluation of Answer Triggering (AT)

In both QA and chatbot, response triggering is important. Similar to Yang et al. (2015), we also evaluate answer triggering using Precision, Recall, and F1 score as metrics. We use the WikiQA development set to tune the scaling factor α and trigger threshold τ that are described in Section 5, where α is set to 0.9 and τ is set to 0.5.

Models	MAP	Change	MRR	Change
DocChat	68.25%		70.73%	
DocChat - F_w	66.06%	-2.19	67.99%	-2.74
DocChat - F_p	66.80%	-1.45	68.66%	-2.07
DocChat - F_s	65.49%	**-2.76**	67.27%	**-3.46**
DocChat - F_d	68.02%	-0.23	69.79%	-0.94
DocChat - F_r	67.00%	-1.25	69.07%	-1.66
DocChat - F_{ty}	67.09%	-1.16	69.28%	-1.45
DocChat - F_{to}	66.85%	-1.40	68.96%	-1.77

Table 4: Impacts of different feature groups.

Methods	Precision	Recall	F1
Yang et al. (2015)	28.34	35.80	31.64
DocChat	**28.95**	**44.44**	**35.06**

Table 5: Evaluation of AT on WikiQA.

Table 5 shows the evaluation results compare to Yang et al. (2015). We think the improvements come from the fact that our response ranking model are more discriminative, as more semantic-level features are leveraged.

7.2 Evaluation on Chatbot (Chinese)

XiaoIce is a famous Chinese chatbot engine, which can be found in many platforms including WeChat official accounts (like business pages on Facebook Messenger). The documents that each official account maintains and post to their followers can be easily obtained from the Web. Meanwhile, a WeChat official account can choose to authorize XiaoIce to respond to its followers' utterances. We design an interesting evaluation below to compare DocChat with XiaoIce, based on the publicly available documents.

7.2.1 Experiment Setup

For ranking features, 17M 'question-related questions' pairs crawled from Baidu Zhidao are used to train word alignments for h_{W2W}; sentences from Chinese Wikipedia are used to train word embeddings for h_{W2V} and a topic model for h_{UTM}; the same bilingual phrase table described in last experiment is also used to extract a Chinese paraphrase table for h_{PP} which use Chinese as the source language; 5M 'question-answer' pairs crawled from Baidu Zhidao are used for h_{PT}, h_{SCR} and h_{DM}; 0.5M 'sentence-next sentence' pairs from Chinese Wikipedia are used for h_{SDR}; 1.3M 'sentence-topic pairs' crawled from Chinese Wikipedia are used to train $topic-$CNN for

Utterance	Response
你知道北京的历史么？ (Do you know the history of Beijing?)	[XiaoIce Response]: 我的历史课学的不太好。 (I am not good at history class)
	[DocChat Response]:北京历史悠久，可以追溯到3000年前。 (Beijing is a historical city that can be traced back to 3,000 years ago.)

Table 6: XiaoIce response is more colloquial, as it comes from Q-R pairs; while DocChat response is more formal, as it comes from documents.

h_{STM}. As there is no knowledge base based labeled data for Chinese, we ignore relation-level feature h_{RE} and type-level feature h_{TE}.

For ranking weights, we generate 90,321 $\langle \mathcal{Q}, \mathcal{C} \rangle$ pairs based on Baidu Zhidao Q-A pairs by the automatic method described in Section 4.8. This data set is used to train the learning to rank model feature weights $\{\lambda_k\}$ by SGD.

For documents, we randomly select 10 WeChat official accounts, and index their documents separately. The average number of documents is 600.

Human annotators are asked to freely issue 100 queries to each official account to get XiaoIce response. Thus, we obtain 100 ⟨query, XiaoIce response⟩ pairs for each official account. We also send the same 100 queries of each official account to DocChat based on official account's corresponding document index, and obtain another 100 ⟨query, DocChat response⟩ pairs. Given these 1,000 ⟨query, XiaoIce response, DocChat response⟩ triples, we let human annotators do a side-by-side evaluation, by asking them which response is better for each query. Note that, the source of each response is masked during evaluation procedure. Table 6 gives an example.

7.2.2 DocChat v.s. XiaoIce

Table 7 shows the results. **Better** (or **Worse**) denotes a DocChat response is better (or worse) than a XiaoIce response, **Tie** denotes a DocChat response and a XiaoIce response are equally good or bad. From Table 7 we observe that: (1) 156 DocChat responses (58+47+51) out of 1,000 queries are triggered. The trigger rate of DocChat is 15.6%. We check un-triggered queries, and find most of them are chitchat, such as "hi", "hello", "who are you?". (2) Better cases are more than worse cases. Most queries in better cases are non-chitchat ones, and their contents are highly related to the domain of their corresponding WeChat official accounts. (3) Our proposed method is a perfect complement for chitchat engines on informative utterances. The reasons for bad cases are two-fold: First, a DocChat response overlaps with a query, but cannot actually response it. For this issue, we need to refine the capability of our response ranking model on measuring causality relationships. Second, we wrongly send a chitchat query to DocChat, as currently, we only use a white list of chitchat queries for chitchat/non-chitchat classification (Section 5).

	Better	Worse	Tie
Compare to XiaoIce	58	47	51

Table 7: Chatbot side-by-side evaluation.

8 Conclusion

This paper presents a response retrieval method for chatbot engines based on unstructured documents. We evaluate our method on both question answering and chatbot scenarios, and obtain promising results. We leave better triggering component and multiple rounds of conversation handling to be addressed in our future work.

Acknowledgments

This paper is supported by Beijing Advanced Innovation Center for Imaging Technology (No.BAICIT-2016001), the National Natural Science Foundation of China (Grant Nos. 61170189, 61370126), National High Technology Research and Development Program of China under grant (No.2015AA016004), the Fund of the State Key Laboratory of Software Development Environment (No.SKLSDE-2015ZX-16).

References

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 597–604.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

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

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*.

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

Zongcheng Ji, Zhengdong Lu, and Hang Li. 2014. An information retrieval approach to short text conversation. *arXiv preprint arXiv:1408.6988*.

K Sparck Jones, Steve Walker, and Stephen E. Robertson. 2000. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information Processing & Management*, 36(6):809–840.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *Proceedings of Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 1:48–54.

Diane Litman, Satinder Singh, Michael Kearns, and Marilyn Walker. 2000. Njfun: a reinforcement learning spoken dialogue system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 17–20.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural variational inference for text processing. *arXiv preprint arXiv:1511.06038*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, pages 3111–3119.

Ramesh Nallapati. 2004. Discriminative models for information retrieval. In *Proceedings of the international ACM SIGIR conference on Research and development in information retrieval*, pages 64–71.

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

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 583–593.

Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(02):97–126.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1577–1586.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 1164–1172.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 707–712.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 7, pages 22–32.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2013–2018.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1744–1753.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 643–648.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *NIPS Deep Learning and Representation Learning Workshop*.

Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. 2015. Lightlda: Big topic models on modest computer clusters. In *Proceedings of the Annual International Conference on World Wide Web (WWW)*, pages 1351–1361.

Investigating the Sources of Linguistic Alignment in Conversation

Gabriel Doyle
Department of Psychology
Stanford University
Stanford, CA 94305
gdoyle@stanford.edu

Michael C. Frank
Department of Psychology
Stanford University
Stanford, CA 94305
mcfrank@stanford.edu

Abstract

In conversation, speakers tend to "accommodate" or "align" to their partners, changing the style and substance of their communications to be more similar to their partners' utterances. We focus here on "linguistic alignment," changes in word choice based on others' choices. Although linguistic alignment is observed across many different contexts and its degree correlates with important social factors such as power and likability, its sources are still uncertain. We build on a recent probabilistic model of alignment, using it to separate out alignment attributable to words versus word categories. We model alignment in two contexts: telephone conversations and microblog replies. Our results show evidence of alignment, but it is primarily lexical rather than categorical. Furthermore, we find that discourse acts modulate alignment substantially. This evidence supports the view that alignment is shaped by strategic communicative processes related to the ongoing discourse.

1 Introduction

In conversation, people tend to adapt to one another across a broad range of behaviors. This adaptation behavior is collectively known as "communication accommodation" (Giles et al., 1991). Linguistic alignment, the use of similar words to a conversational partner, is one prominent form of accommodation. Alignment is found robustly across many settings, including in-person, computer-mediated, and web-based conversation (Danescu-Niculescu-Mizil et al., 2012; Giles et al., 1979; Niederhoffer and Pennebaker, 2002). In addition, the strength of alignment to conversational partners varies with relevant sociological factors, such as the power of the partners, their social network centrality, and their likability. Potentially, this alignment could be used to infer these factors in situations where they are difficult to observe directly.

Although linguistic alignment appears to reflect important social dynamics, the mechanisms underlying alignment are still not well-understood. One particular question is whether alignment is supported by relatively automatic priming mechanisms, or higher-level, discourse and communicative strategies. The Interactive Alignment Model proposes that conversational partners prime each other, causing alignment via the primed reuse of structures ranging from individual lexical items to syntactic abstractions (Pickering and Garrod, 2004). In contrast, Accommodation Theory emphasizes the relatively more communicative and strategic nature of alignment (Giles et al., 1991).

Relative to this theoretical landscape, a number of questions have emerged. First, does alignment occur at structural levels? If alignment is driven by interactive priming of structures, effects of alignment should be expected not only at the lexical level but also for structural elements or categories as well. In contrast, if alignment is primarily communicative, then alignment strength might differ and be greater for specific words that serve particular conversational or discourse functions in a particular situation.

Second, does alignment vary with conversational goals? If alignment is driven primarily by priming, it should be relatively consistent across different aspects of a discourse. In contrast, from a strategic or communicative perspective, alignment – in which preceding words and concepts are reused – must be balanced against a need to move the conversation forward by introducing new words and concepts. Thus, on a communica-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 526–536,
Berlin, Germany, August 7-12, 2016.

tive account, alignment should be modulated by the speaker's discourse act, reflecting whether the balance of the concern is convergence on a current focus or conveyal of new information.

Our goal in the current work is to investigate these questions. We make use of a recent probabilistic model of linguistic alignment, modifying it to operate robustly over corpora with highly varying distributional structures and to consider both lexical and category-based alignment. We use two corpora of spontaneous conversations, the Switchboard Corpus and a corpus of Twitter conversations, to perform two experiments. First, in both datasets we measure alignment across different levels of representation and find very limited evidence for category-level alignment. Second, we make use of annotations in Switchboard to measure alignment across different discourse acts, finding that the level of alignment depends on the discourse actions that are included in the analysis. Taken together, these findings are consistent with the idea that alignment arises from discourse-level, strategic processes that operate primarily over lexical items.

2 Previous Work

2.1 Why does alignment matter?

Linguistic alignment, like other kinds of accommodation, can be a critical part of achieving social goals. Performance in cooperative decision-making tasks is positively related to the participants' linguistic convergence (Fusaroli et al., 2012; Kacewicz et al., 2013). Romantically, match-making in speed dating and stability in established relationships have both been linked to increased alignment (Ireland et al., 2011). Alignment can also improve perceived persuasiveness, encouraging listeners to follow good health practices (Kline and Ceropski, 1984) or to leave larger tips (van Baaren et al., 2003).

Alignment is also important as an indicator of implicit sociological variables. Less powerful conversants generally accommodate to more to powerful conversants. Prominent examples include interviews and jury trials (Willemyns et al., 1997; Gnisci, 2005; Danescu-Niculescu-Mizil et al., 2012). A similar effect is found for network structure: speakers align more to more network-central speakers (Noble and Fernández, 2015). Additionally, factors such as gender, likability, respect, and attraction all interact with the magnitude of accommodation (Bilous and Krauss, 1988; Natale, 1975).

2.2 Sources of linguistic alignment

Despite the important outcomes associated with alignment, its sources are not clear. The most prominent strand of work on alignment has focused on the level of word categories, looking at how interlocutors change their frequency of using, for instance, pronouns or quantitative words (Danescu-Niculescu-Mizil et al., 2012; Ireland et al., 2011). These results show alignment effects at the category level, but it is in principle possible that these effects arose purely from alignment on individual words (and that conclusion would not be inconsistent with the interpretation of that work).

Syntactic alignment is one area in which theoretical predictions have been tested, though results have been somewhat equivocal. The Interactive Alignment Model has generally been taken to suggest that there should be cross-person priming of syntactic categories and structures (Pickering and Garrod, 2004). But while some studies have found support for syntactic priming (Gries, 2005; Dubey et al., 2005), others have found negative or null alignment (Healey et al., 2014; Reitter et al., 2006). In one particularly thorough study, Healey et al. (2014) found across two corpora that speakers syntactically *diverged* from their interlocutors once lexical alignment was accounted for.

Furthermore, positive alignment is generally regarded as a good conversational tactic, but there is clearly a limit to its virtues, at least when it comes to content words. Alignment is inherently backward-looking, while the general goal of a conversation is to exchange information that is not already known by both parties, an inherently forward-looking goal. Perhaps because of this, some recent work finding positive alignment has limited itself to "non-topical" word categories, which are less contentful (Danescu-Niculescu-Mizil et al., 2011; Doyle et al., 2016). And suggestively, alignment within a task-relevant syntactic category was a better predictor of decision-making performance than overall lexical alignment (Fusaroli et al., 2012).

In sum, although individual studies do bear on the sources of alignment, the picture is still not clear. Because most work on alignment has been done either on categories of words or aggregating

across the lexicon, we do not have a good sense of whether there are systematic differences in alignment at different levels of representation. A further complication is that there is no standard measure of alignment; we turn to this issue next.

2.3 Measures of alignment

The metrics used in previous work fall into two basic categories: distributional and conditional. Distributional methods such as Linguistic Style Matching (LSM) (Niederhoffer and Pennebaker, 2002; Ireland et al., 2011) or the Zelig Quotient (Jones et al., 2014) calculate the similarity between the conversation participants over their frequencies of word or word category use in all utterances within the conversation. In contrast, conditional metrics, such as Local Linguistic Alignment (LLA) (Fusaroli et al., 2012; Wang et al., 2014) and the metric used by Danescu-Nicolescu-Mizil et al. (2011), look at how a message conditions its reply, with alignment indicated by elevated word use in the reply when that word was in the preceding message.

While distributional methods have been popular, a major weakness of such methods is that they do not necessarily show true alignment, only similarity. A high level of distributional similarity does not imply that two conversational partners have aligned to one another, because they might instead have been similar to begin with. In contrast, conditional measures allow for stronger inferences about the temporal sequence of alignment (even though they cannot guarantee any causal interpretation). Thus, we focus here on conditional measures exclusively.

By-message conditional methods Several existing conditional methods have started from the simplified representation that messages either do or do not contain particular words ("markers"), irrespective of message length or marker count. (Danescu-Niculescu-Mizil et al., 2012; Doyle et al., 2016). We refer to these as "by-message" methods. Consider the following example of conditional alignment, using pronouns as the marker: Bob aligns to Alice if his replies are more likely to contain a pronoun when in response to a message from Alice that contains a pronoun.

	Bob's reply	
Alice's message	has pronoun	no pronoun
has pronoun	8	2
no pronoun	5	5

Here, Alice sends 10 messages that contain at least one pronoun, and 8 of Bob's replies contain at least one pronoun. But Alice also sends 10 messages that don't contain any pronouns, and only 5 of Bob's replies to these contain pronouns. This increased likelihood of a pronoun-containing reply to a pronoun-containing message is the conditional alignment.

Different models quantify this conditional alignment slightly differently. Danescu-Niculescu-Mizil et al. (2011) proposed a subtractive conditional probability model, where alignment is the difference between the likelihood of a pronoun-containing reply B to a pronoun-containing message A and the probability of a pronoun-containing reply to any message:

$$align_{SCP} = p(B|A) - p(B) \quad (1)$$

Doyle et al. (2016) showed that this measure can be affected by the overall frequency of the category being aligned on, though. To correct this issue, they proposed a Hierarchical Alignment Model (HAM), which defines alignment as a linear effect on the log-odds of a reply containing the relevant marker (e.g., a pronoun), similar to a linear predictor in a logistic regression.[1]

$$align_{HAM} \approx logit^{-1}(p(B|A)) - logit^{-1}(p(B|\neg A)) \quad (2)$$

These binary conditional methods depend on the assumption that all messages have similar, and small, numbers of words, however. The probability that a message contains at least one of any marker of interest is dependent on the message's length, so if messages vary substantially in their length, these alignment values can be at least noisy, if not biased. They are also not robust as messages increase in length, since the likelihood that a message contains any marker approaches 1 as message length increases.

By-word conditional methods A solution to the problem of variable message lengths is simply to shift from binarized data to count data. Instead of counting how many times Bob's replies contain at least one pronoun, we can count what proportion of his replies' word tokens are pronouns.

[1] Because the HAM estimated this quantity via Bayesian inference, the inferred alignment value depends on the prior and number of messages observed, so unlike the other measures, this equality is only approximate.

Some existing measures use a related quantity, the proportion of the preceding message that appears in its reply, to estimate alignment, notably Local Linguistic Alignment (LLA) (Fusaroli et al., 2012; Wang et al., 2014) and the lexical similarity (LS) measure of Healey et al. (2014). LLA is defined as the number of word tokens (w_i) that appear in both the message (M_a) and the reply (M_b), divided by the product of the total number of word tokens in the message and reply:

$$align_{LLA} = \frac{\sum_{w_i \in M_b} \delta(w_i \in M_a)}{length(M_a) length(M_b)} \quad (3)$$

These measures have an aspect of conditionality, as they only count words that appear in both the message and the reply. But they nevertheless fail to control for the baseline frequency of the initial marker, and hence may be biased in measurements across words or categories of different frequencies (Doyle et al., 2016). They also can be affected by reply length, as the maximum alignment estimate is only possible when the reply is shorter than the message.

All of these by-word conditional models treat the reply as a bag of words, without order information. The by-word models, including the WHAM model we propose, are agnostic about reply length effects, correcting for the artifactual length effects of by-message models, but assuming that all messages have similar alignment strengths independent of length. This is in contrast to models that explicitly model priming effects as decaying over time (Reitter et al., 2006; Reitter, 2008), which predict higher alignment in shorter replies. Future by-word alignment models could infer a discounting for words that occur later in the reply, similar to the beta value on the log-distance from the prime proposed in Reitter et al. (2006).

Our goal in this work is to create a model that combines the benefits of the existing by-message conditional models with the length-robustness of a by-word conditional method. We present WHAM, a modification of the HAM model that satisfies this goal.

3 The Word-Based Hierarchical Alignment Model (WHAM)

We propose the Word-Based Hierarchical Alignment Model (WHAM). Like HAM, WHAM assumes that word use in replies is shaped by whether the preceding message contained the marker of interest. But WHAM uses marker token frequencies within replies, so that a 40-word reply with two instances of the marker is represented differently from a 3-word reply containing one instance.

For each marker, WHAM treats each reply as a series of token-by-token independent draws from a binomial distribution. The binomial probability μ is dependent on whether the preceding message did (μ^{align}) or did not (μ^{base}) contain the marker, and the inferred alignment value is the difference between these probabilities in log-odds space (η^{align}). The graphical model is shown in Figure 1.

For a set of message-reply pairs between a speaker-replier dyad (a, b), we first separate the replies into two sets based on whether the preceding message contained the marker m (the "alignment" set) or not (the "baseline" set). All replies within a set are then aggregated in a single bag-of-words representation, with marker token counts $C^{align}_{m,a,b}$ and $C^{base}_{m,a,b}$, and total token counts $N^{base}_{m,a,b}$ and $N^{base}_{m,a,b}$, the observed variables on the far right of the model. Moving from right to left, these counts are assumed to come from binomial draws with probability $\mu^{align}_{m,a,b}$ or $\mu^{base}_{m,a,b}$. The μ values are generated from η values in log-odds space by an inverse-logit transform, similar to linear predictors in logistic regression.

The η^{base} variables are representations of the baseline frequency of a marker in log-odds space, and μ^{base} is simply a conversion of η^{base} to probability space, the equivalent of an intercept term in a logistic regression. η^{align} is an additive value, with $\mu^{align} = \text{logit}^{-1}(\eta^{base} + \eta^{align})$, the equivalent of a binary feature coefficient in a logistic regression. Alignment is then the change in log-odds of the replier using m above baseline usage, given that the initial message uses m.

The remainder of the model is a hierarchy of normal distributions that allow social and word category structure to be integrated into the analysis. In the present work, we have three levels in the hierarchy: category level, marker level,[2] and conversational dyad level. All of these normal distributions have identical standard deviations $\sigma^2 = .25$.[3] A $Cauchy(0, 2.5)$ distribution

[2] In the lexical and category-not-word alignment models, these markers are words within a category. The category alignment model does not include this level, since all words in a category are treated identically.

[3] This value was chosen as a good balance between rea-

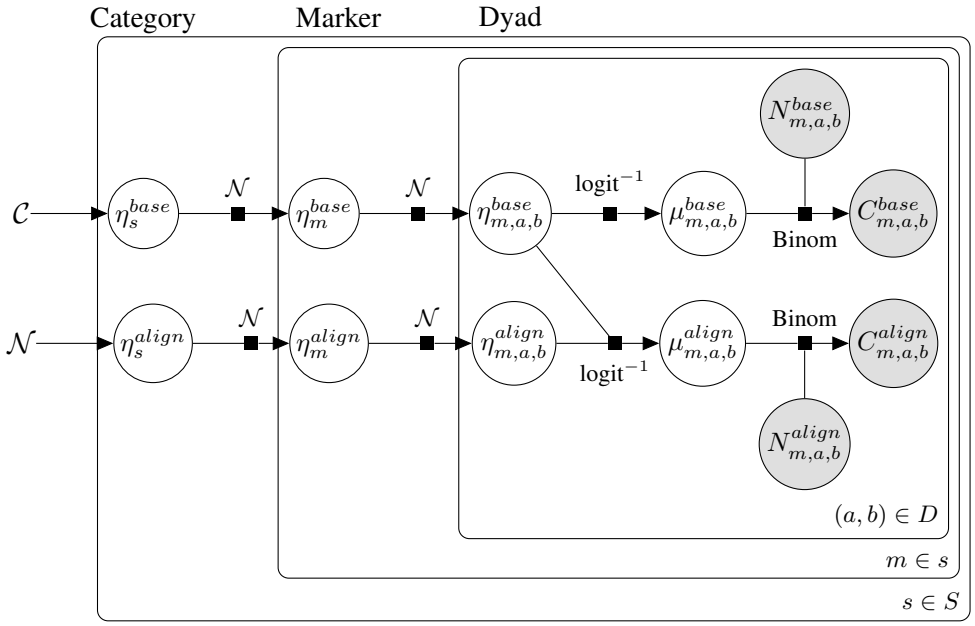


Figure 1: The Word-Based Hierarchical Alignment Model (WHAM). A chain of normal distributions generates a linear predictor η, which is converted into a probability μ for binomial draws of the words in each reply.

gives a relatively uninformative prior for the baseline marker frequency (Gelman et al., 2008). The alignment hierarchy is headed by a normal distribution centered at 0, biasing the model equally in favor of positive and negative alignments.

For our marker set, we adopt the Linguistic Inquiry and Word Count (LIWC) system to categorize words (Pennebaker et al., 2007). We use a set of 11 categories that have shown alignment effects in previous work (Danescu-Niculescu-Mizil et al., 2011). These can be loosely grouped into a set of five syntactic categories (articles, conjunctions, prepositions, pronouns, and quantifiers) and six conceptual categories (certainty, discrepancy, exclusion, inclusion, negation, and tentative). Categories and example elements are shown in Table 1. We manually lemmatized all words in each category. We implemented WHAM in RStan (Carpenter, 2015), with code available at `http://github.com/langcog/disc_align`.

Category	Examples	Size	Swbd Prob	Twit Prob
Article	*a, the*	2	.053	.047
Certainty	*always, never*	17	.014	.015
Conjunction	*but, and, though*	18	.077	.051
Discrepancy	*should, would*	21	.015	.019
Exclusive	*without, exclude*	77	.038	.028
Inclusive	*with, include*	57	.057	.028
Negation	*not, never*	12	.020	.023
Preposition	*to, in, by, from*	97	.097	.091
Pronoun	*it, you*	55	.17	.16
Quantifier	*few, many*	23	.028	.025
Tentative	*maybe, perhaps*	28	.033	.025

Table 1: Marker categories for linguistic alignment, with examples, number of distinct word lemmas, and token probability of in a reply in Switchboard and Twitter.

3.1 Validating WHAM

A major goal of our by-word alignment model, WHAM, is to fix the length issues discussed in Section 2.3. We test WHAM and the by-message HAM model on simulated data, using a method similar to Simulation 2 in Doyle et al. (2016), to see how robust they are to different reply lengths. We generate 500 speaker-replier dyads, each exchanging an average of 5 message pairs (drawn from a geometric distribution). Each message pair consists of a message whose length in words is drawn from a uniform distribution $[1, 25]$, and a reply of length L. Because our goal is to test the effect of length on the models' performances, we create separate simulated datasets for different values of L, and see whether the model correctly estimates the alignment value η^{align}. Three independent simulations were run for each alignment-length pair. We present data here for a simulated

sonable parameter convergence (improved by smaller σ^2) and good model log-probability (improved by larger σ^2).

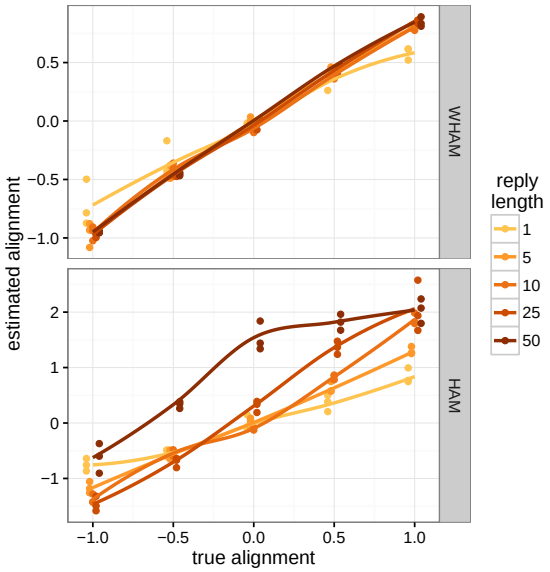


Figure 2: Actual versus estimated alignment on simulated data. Lines are loess-fit curves; colors represent the reply length in the simulation run. WHAM estimates alignment accurately regardless of reply length; HAM is highly affected by length.

word category with a baseline frequency of 0.1, around the middle of the attested category frequency range (see Table 1).

Figure 2 plots the true alignment value in the simulations against the model-estimated alignment values. Different colors represent different reply lengths L, ranging from single-word replies (light yellow) to 50-word replies (dark orange). The WHAM model shows consistently accurate alignment estimates over the range of simulated alignment values and reply lengths. The HAM model estimates the alignment far less accurately, and the reply length biases its estimates.

4 Data

Moving on to real data, we use two corpora for our experiments. The first is a collection of Twitter conversations collected by Doyle & Frank (2015) to examine information density in conversation. This corpus focuses on conversations within a set of 14 mostly distinct sub-communities on Twitter, and contains 63,673 conversation threads, covering 228,923 total tweets. We divide these conversations into message pairs, also called conversational turns, which are two consecutive tweets within a conversation thread. The second tweet is always in reply to the first (according to the Twitter API), although this does not necessarily mean that the content of the reply is a response to the preceding tweet. Retweets (including explicit retweets and some common manual retweet methods) were removed automatically. This processing leaves us with 122,693 message pairs, spanning 2,815 users. The tweets were parsed into word tokens using the Twokenizer (Owoputi et al., 2013).

The second corpus is the SwDA version of the Switchboard corpus (Godfrey et al., 1992; Jurafsky et al., 1997).[4] This corpus is a collection of transcribed telephone conversations, with each utterance labeled with the discourse act it is performing (e.g., statement of opinion, signal of non-understanding). It contains 221,616 total utterances in 1,155 conversations. We combine consecutive utterances by the same speaker without interruption from the listener into a single message and treat consecutive pairs of messages from different speakers as conversation turns, resulting in 110,615 message pairs.

5 Experiment 1: Lexical- and Category-Level Alignment

Our first experiment examines how alignment differs across the lexical and categorical levels. We use the WHAM framework to infer alignment on word and category counts, and also introduce a measure to estimate the influence of one word in a category on other words in its category, "category-not-word" alignment. We include this last type of alignment because it is possible that the category alignment effects in previous work are the result of lexical alignment on the individual words in the category, without any influence across words in the category. If categorical alignment is a real effect over and above lexical alignment, as an interactive-priming source for alignment would suggest, then the presence of a word in a message should not only increase the chance of seeing that word in the reply, but also other words in its category.

5.1 Category-not-word-alignment model

Assessing the amount of alignment triggered across words in a category (which we call "category-not-word alignment" or CNW) is not trivial, as there are a variety of interactions between lexical items within a category that can cause the lexical alignment to actually be less than

[4] Available courtesy of Christopher Potts at `http://compprag.christopherpotts.net/swda.html`.

	Reply		
Message	$\emptyset$	*he*	*she*
$\emptyset$	25	25	25
he	20	50	10
she	20	10	50

Table 2: A theoretical case where lexical alignment surpasses categorical alignment due to negative CNW between the words.

the category alignment. Table 2 illustrates this with a theoretical distribution over the pronouns *he* and *she*; one use of the pronoun *he* makes another use more likely (*A: Did he like the movie? B: Yeah, he loved it.*) while also reducing the likelihood of *she*, since the topic of conversation is now a male, and vice versa for *she*. For both *he* and *she*, the lexical alignment is approximately $logit^{-1}(p(B|A) - p(B|\neg A)) = logit^{-1}(\frac{50}{80} - \frac{25}{75}) \approx 1.2$, but categorical alignment is approximately $logit^{-1}(\frac{120}{160} - \frac{50}{75}) \approx 0.4$. On the other hand, the pronouns *you* and *I* might trigger each other more than themselves (*A: Did you like the movie? B: Yeah, I loved it.*).

The differences between lexical, categorical, and CNW alignment are also relevant to discussions of "lexical boosts" in the syntactic priming literature, an increased priming effect at the categorical level when there is lexical repetition. Lexicalist residual activation accounts (Pickering and Branigan, 1998) predict such a boost, while implicit learning accounts do not (Bock and Griffin, 2000; Chang et al., 2006). In the context of this experiment, such a lexical boost could make lexical and categorical alignment appear elevated and closer together, but would not have a substantial effect on CNW alignment.[5]

To investigate CNW alignment, we look at a subset of the data: for each word w, exclude all messages that contain a word from that category (S) that is not w. This limits the category alignment influence on the reply to the single word w. Then, instead of looking at how often w appears in the reply, we look at how often *all other words* in category S appear in the reply. The model then infers the influence of w on the other words in the category independent of their lexical alignment. Within the WHAM model, we change the count variables $C^{\cdot}$ and $N^{\cdot}$ so that C^{align} is the number of tokens of $\{S - w\}$ in replies to messages containing w but not $\{S - w\}$. C^{base} is then the number in replies to messages not containing any words in S. Similarly, N^{align} is the total token counts over replies containing w but not any other words in S, and N^{base} the total token counts over replies containing no words in S.

5.2 Methods

We conducted three sets of simulations, fitting the model with marker categories, individual words, and with the CNW scheme described above. In each, the model was fit with two chains of 200 iterations of the sampler for each dataset. We then extracted alignment estimates from each of the final 100 samples, and we report 95% highest posterior density intervals on η_S^{align}.

5.3 Results

Figure 3 shows the alignment on each marker category in the Twitter and Switchboard corpora. There were substantial differences in the overall rate of alignment between the corpora: Mean category alignment on Twitter was .19, while Switchboard category alignment was $-.051$. These differences may reflect the nature of the two discourse contexts: Replies on Twitter are composed while looking at the preceding message, encouraging the replier to take more account of the other tweeter's words, and a replier can draft and edit their reply to make it better fit the conversation. Messages on Switchboard, on the other hand, are evanescent, so a replier must compose a reply without looking back at the message, without editing, and in real-time. Differences in the discourse structure of these corpora may also be contributing, an effect we will consider in Experiment 2.

Despite the difference in reply construction in the two corpora, the results across levels of alignment were similar. Alignment was found primarily at the lexical – rather than the category – level. Lexical and category alignment were not significantly different from each other, but the strength of lexical alignment was significantly larger than the CNW alignment, according to a t-test over categories (Twitter: $t(10) = .21, p < .001$; Swbd: $t(10) = .12, p = .003$). CNW alignment was significantly negative on Switchboard ($t(10) = -.11, p = .01$) and not significantly different from zero on Twitter ($t(10) = .009, p = .79$).

[5]The categories being investigated in our work contain mostly non-topical, closed-class words, which have not exhibited lexical boosts in past research (Bock, 1989; Pickering and Branigan, 1998; Hartsuiker et al., 2008), but such boosting may be detectable in estimates on topical categories.

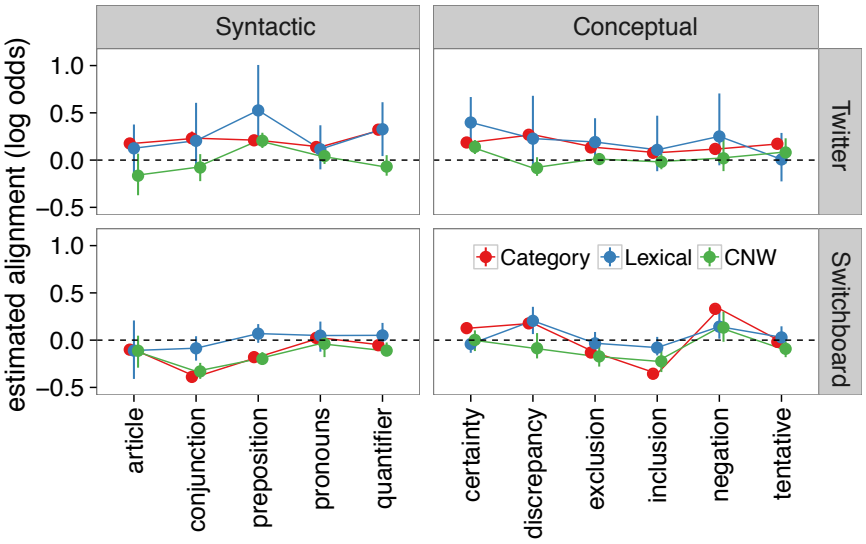


Figure 3: Categorical (red), lexical (blue), and CNW (green) alignments plotted by category, on the Twitter (left) and Switchboard (right) datasets. 95% HPD intervals from WHAM shown.

WHAM – unlike other previous measures – provides estimates of alignment that are unbiased by either marker frequency or message length, but we still observed modest alignment on Twitter, replicating previous work (Doyle et al., 2016; Danescu-Niculescu-Mizil et al., 2011). Alignment was smaller in Switchboard, and in both cases there were no category effects. Thus, the categorical alignment results may result primarily from lexical alignment, inconsistent with the predictions of interactive priming accounts of alignment.

6 Experiment 2: Discourse Acts and Alignment

Messages within a discourse can serve a very wide range of purposes. This variety has effects for both linguistic structure and the relationship to neighboring messages. For example, a simple yes/no question is likely to receive a short, constrained reply, while a statement of an opinion is more likely to yield a longer reply. In addition, different types of messages can either introduce new information to the conversation (e.g., statements, questions, offers) or look back at existing information (e.g., acknowledgments, reformulations, yes/no answers). We hypothesize that alignment will be substantially different depending on the discourse act, as speakers' conversational goals vary. Thus, our second experiment examines how alignment differs depending on discourse act.

We focus on a particular kind of discourse act, the backchannel (Yngve, 1970). Backchannels are extremely common in Switchboard, accounting for almost 20% of utterances, and include utterances such as single words signaling understanding or misunderstanding (*yeah, uh-huh, no*) or simple messages expressing empathy without trying to take a full conversational turn (*It must have been tough*). Backchannels are a particularly interesting case because their short and constrained nature makes it difficult to align on some categories (e.g., backchannels rarely contain quantifiers or prepositions), while the purpose of giving feedback to the speaker makes it important to align on others (e.g., matching the positive/negative tone or certainty of a speaker). In addition, backchannels are primarily restricted to spoken corpora. Twitter conversations contain far fewer backchannels than Switchboard, which may account for some of their alignment differences—especially as the results of this experiment suggest that backchannels reduce overall alignment.

6.1 Methods

We use the discourse-annotated Switchboard corpus to compare alignment in conversations containing backchannels with those whose backchannels have been removed. We make this comparison by creating a second corpus, removing every utterance classified as a backchannel from the corpus prior to parsing the utterances into conversation turns as before.

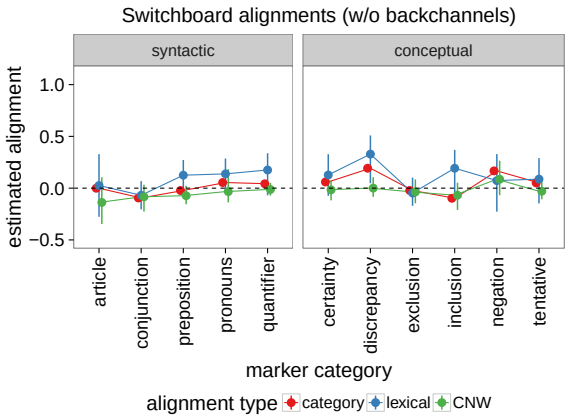


Figure 4: Categorical (red), lexical (blue), and CNW (green) alignments on the Switchboard dataset with backchannels removed. 95% HPD intervals from WHAM shown.

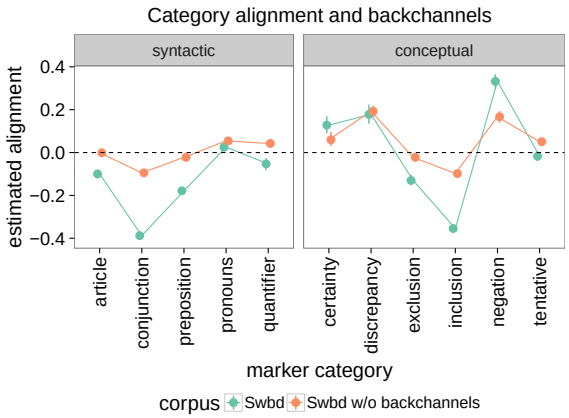


Figure 5: Comparing categorical alignment on the Switchboard dataset with and without backchannels. 95% HPD intervals from WHAM shown.

6.2 Results

Alignment values for the Switchboard corpus without backchannels are shown in Figure 4. As expected, alignment is on average higher without the backchannels ($p = .09$ for category, $p < .05$ for lexical and CNW), reflecting the constrained nature of backchannels. Lexical alignment is significantly higher than category alignment ($t(10) = -.08, p = .03$), consistent with the findings of Experiment 1. The mean category alignment without backchannels is .029.

Figure 5 compares the category alignments for the full Switchboard corpus (green) and Switchboard without backchannels (orange). Alignment on the full corpus is lower for all but two categories, exhibiting the reduced opportunity for alignment provided by backchannels. Syntactic category alignment is especially affected by backchannels, whose constrained forms provide very little ability to align syntactically.

Interestingly, the two categories that do show greater alignment when backchannels are included are certainty and negation. These categories are both important for backchannels; a negative backchannel is generally inappropriate in reply to a non-negative message, and similarly a confident backchannel would often be out of place in reply to an uncertain message. These influences of discourse acts on alignment are more consistent with a discourse-strategic origin for alignment than a priming-based account.

7 Discussion

Linguistic alignment is a prominent type of communicative accommodation, but its sources are unclear. We presented WHAM, a length-robust extension of a probabilistic alignment model. Using this model, we find evidence that linguistic alignment is primarily lexical, and that it is strongly affected by at least some aspects of the discourse goal of a message.

This combination of a primarily-lexical origin for linguistic alignment and its variation by word category and discourse act suggest that alignment is primarily a higher-level discourse strategy rather than a low-level priming-based mechanism. This set of results is consistent with both Accommodation Theory and the set of findings, reviewed above, that sociological factors affect the level of observed alignment. The effect of discourse acts on alignment further suggests that alignment is not a completely automatic process but rather one of many discourse strategies that speakers use to achieve their conversational goals.

Acknowledgments

We wish to thank Dan Yurovsky, Aaron Chuey, and Jake Prasad for their work on and discussion of earlier versions of the model, Herb Clark for discussions of potential effects of message length, and, of course, the reviewers. The authors were funded by NSF BCS 1528526, NSF BCS 1456077, and a grant from the Stanford Data Science Initiative.

References

Frances R. Bilous and Robert M. Krauss. 1988. Dominance and accommodation in the conversational behaviours of same-and mixed-gender dyads. *Language & Communication*.

Kay Bock and Zenzi M. Griffin. 2000. The persistence of structural priming: Transient activation or implicit learning. *Journal of Experimental Psychology: General*, 129:177–192.

Kay Bock. 1989. Closed-class immanence in sentence production. *Cognition*, 31:163–186.

Bob Carpenter. 2015. Stan: A Probabilistic Programming Language. *Journal of Statistical Software*.

Franklin Chang, Gary S. Dell, and Kay Bock. 2006. Becoming syntactic. *Psychological Review*, 113:234–272.

Cristian Danescu-Niculescu-Mizil, Michael Gamon, and Susan Dumais. 2011. Mark my words!: linguistic style accommodation in social media. In *Proceedings of the 20th international conference on World Wide Web - WWW '11*, page 745, New York, New York, USA. ACM Press.

Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang, and Jon Kleinberg. 2012. Echoes of power: Language effects and power differences in social interaction. In *Proceedings of the 21st international conference on World Wide Web - WWW '12*, page 699.

Gabriel Doyle and Michael C. Frank. 2015. Audience size and contextual effects on information density in Twitter conversations. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*.

Gabriel Doyle, Dan Yurovsky, and Michael C. Frank. 2016. A robust framework for estimating linguistic alignment in Twitter conversations. In *WWW 2016*.

Amit Dubey, Patrick Sturt, and Frank Keller. 2005. Parallelism in coordination as an instance of syntactic priming: Evidence from corpus-based modeling. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 827–834. Association for Computational Linguistics.

Riccardo Fusaroli, Bahador Bahrami, Karsten Olsen, Andreas Roepstorff, Geraint Rees, Chris Frith, and Kristian Tylén. 2012. Coming to Terms: Quantifying the Benefits of Linguistic Coordination. *Psychological Science*, 23(8):931–939.

Andrew Gelman, Aleks Jakulin, Maria Grazia Pittau, and Yu-Sung Su. 2008. A weakly informative default prior distribution for logistic and other regression models. *The Annals of Applied Statistics*.

Howard Giles, Klaus R. Scherer, and Donald M. Taylor. 1979. Speech markers in social interaction. In Klaus R. Scherer and Howard Giles, editors, *Social markers in speech*, pages 343–81. Cambridge University Press, Cambridge.

Howard Giles, Nikolas Coupland, and Justine Coupland. 1991. Accommodation theory: Communication, context, and consequences. In Howard Giles, Justine Coupland, and Nikolas Coupland, editors, *Contexts of accommodation: Developments in applied sociolinguistics*. Cambridge University Press, Cambridge.

Augusto Gnisci. 2005. Sequential strategies of accommodation: A new method in courtroom. *British Journal of Social Psychology*, 44(4):621–643.

John J. Godfrey, Edward C. Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *1992 IEEE International Conference on Acoustics, Speech, and Signal Processing.*, volume 1, pages 517–520. IEEE.

Stefan Th Gries. 2005. Syntactic priming: A corpus-based approach. *Journal of psycholinguistic research*, 34(4):365–399.

Robert J. Hartsuiker, Sarah Bernolet, Sofie Schoonbaert, Sara Speybroeck, and Dieter Vanderelst. 2008. Syntactic priming persists while the lexical boost decays: Evidence from written and spoken dialogue. *Journal of Memory and Language*, 58:214–238.

Patrick G. T. Healey, Matthew Purver, and Christine Howes. 2014. Divergence in dialogue. *PloS one*, 9(6):e98598.

Molly E. Ireland, Richard B. Slatcher, Paul W. Eastwick, Lauren E. Scissors, Eli J. Finkel, and James W. Pennebaker. 2011. Language style matching predicts relationship initiation and stability. *Psychological Science*, 22:39–44.

Simon Jones, Rachel Cotterill, Nigel Dewdney, Kate Muir, and Adam Joinson. 2014. Finding Zelig in text: A measure for normalising linguistic accommodation. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics*, pages 455–465.

Dan Jurafsky, Elizabeth Shriberg, and Debra Biasca. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*, pages 97–102.

Ewa Kacewicz, James W. Pennebaker, Matthew Davis, Moongee Jeon, and Arthur C. Graesser. 2013. Pronoun use reflects standings in social hierarchies. *Journal of Language and Social Psychology*, 33(2):125–143.

Susan L. Kline and Janet M. Ceropski. 1984. Person-centered communication in medical practice. In *Human Decision-Making*, pages 120–141. SIU Press, Carbondale.

Michael Natale. 1975. Convergence of mean vocal intensity in dyadic communication as a function of social desirability. *Journal of Personality and Social Psychology*, 32(5):790–804.

Kate G. Niederhoffer and James W. Pennebaker. 2002. Linguistic style matching in social interaction. *Journal of Language and Social Psychology*, 21(4):337–360.

Bill Noble and Raquel Fernández. 2015. Centre Stage: How Social Network Position Shapes Linguistic Coordination. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah Smith. 2013. Improved Part-of-Speech Tagging for Online Conversational Text with Word Clusters. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–391.

James W. Pennebaker, Roger J. Booth, and Martha E. Francis. 2007. Linguistic Inquiry and Word Count: LIWC.

Martin J. Pickering and H. P. Branigan. 1998. The representation of verbs: Evidence from syntactic priming in language production. *Journal of Memory and Language*, 39:633–651.

Martin J. Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and brain sciences*, 27(2):169–190.

David Reitter, Johanna D. Moore, and Frank Keller. 2006. Priming of syntactic rules in task-oriented dialogue and spontaneous conversation. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society*.

David Reitter. 2008. *Context Effects in Language Production: Models of Syntactic Priming in Dialogue Corpora*. Ph.D. thesis, U. of Edinburgh.

Rick B. van Baaren, Rob W. Holland, Bregje Steenaert, and Ad van Knippenberg. 2003. Mimicry for money: Behavioral consequences of imitation. *Journal of Experimental Social Psychology*, 39(4):393–398.

Yafei Wang, David Reitter, and John Yen. 2014. Linguistic Adaptation in Conversation Threads: Analyzing Alignment in Online Health Communities. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Michael Willemyns, Cynthia Gallois, Victor Callan, and Jeffrey Pittam. 1997. Accent accommodation in the employment interview. *Journal of Language and Social Psychology*, 15(1):3–22.

Victor Yngve. 1970. On getting a word in edgewise. In *Papers from the Sixth Regional Meeting of the Chicago Linguistics Society*, pages 567–577.

Entropy Converges Between Dialogue Participants: Explanations from an Information-Theoretic Perspective

Yang Xu and **David Reitter**
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA 16802, USA
yang.xu@psu.edu, reitter@psu.edu

Abstract

The applicability of entropy rate constancy to dialogue is examined on two spoken dialogue corpora. The principle is found to hold; however, new entropy change patterns within the topic episodes of dialogue are described, which are different from written text. Speaker's dynamic roles as *topic initiators* and *topic responders* are associated with decreasing and increasing entropy, respectively, which results in local convergence between these speakers in each topic episode. This implies that the sentence entropy in dialogue is conditioned on different contexts determined by the speaker's roles. Explanations from the perspectives of grounding theory and interactive alignment are discussed, resulting in a novel, unified information-theoretic approach of dialogue.

1 Introduction

Information in written text and speech is strategically distributed. It has been claimed to be ordered such that the rate of information is not only close to the channel capacity, but also approximately constant (Genzel and Charniak, 2002, 2003; Jaeger, 2010); these results were developed within the framework of Information Theory (Shannon, 1948). In these studies, the per-word cross-entropy of a sentence is used to model the amount of information transmitted. Language is treated as a series of random variables of words.

Most existing work examined written text as opposed to speech. Spoken dialogue is different from written text in many ways. For example, dialogue contains more irregular or ungrammatical components, such as incomplete utterances, disfluencies etc. (Jurafsky and Martin, 2014, ch 12), which are "theoretically uninterested complexities that are unwanted" (Pickering and Garrod, 2004). Dialogue is also different from written text in high level discourse structure. The paragraphs in written text, which function as relatively standalone topic units, are constructed under the guidance of one consistent author. On the other hand, the constitution and transformation of topics in dialogue are more dynamic processes, which are the result of the joint activity from multiple speakers (Linell, 1998). In nature, written text is a monologue, while dialogue is a joint activity (Clark, 1996).

From the application perspective, investigating entropy in dialogue can help us better understand which speaker contributes the most information, and thus may potentially benefit tasks such as conversational roles identification (Traum, 2003) etc. From the theoretical perspective, we believe that such investigation will reveal some unique features of the formation of higher level discourse structure in dialogue that are different from written text, e.g., *topic episode* shifts, because previous studies have found the correlation between entropy decrease and potential topic shift in written text (Qian and Jaeger, 2011). Finally, entropy is closely related to predictability and processing demands, which has implications for cognitive aspects of communication.

The main purpose of this study is to characterize how lexical entropy changes in spoken language. We will focus on spontaneous dialogue of two speakers and carry out two steps of investigation. First, we examine the overall entropy patterns within dialogue as a whole context that does not differentiate speakers. Second, we zoom in to *topic episodes* within dialogue and explore how each of the two speakers' entropy develops. The goal of the second step is to account the complexity of topic shifts within spoken dialogues and to reach a more detailed understanding

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 537–546,
Berlin, Germany, August 7-12, 2016.

of human communication from an information-theoretic perspective. If topic shifts in dialogue do correlate with changes in entropy, how do they affect the two speakers, only one of whom typically initiates the topic shift, while another follows along? To answer this question, we use the transcribed text data from two well-developed corpora.

2 Related Work

2.1 The principle of entropy rate constancy

The *constancy rate principle* governing language generation in human communication was first proposed by Genzel and Charniak (2002). Inspired by ideas from Information Theory (Shannon, 1948), this principle asserts that people communicate (written or spoken) in a way that keeps the rate of information being transmitted approximately constant.

Genzel and Charniak (2002) provide evidence to support this principle by formulating the problem into Equation 1. They treat text as a sequence of random variables X_i, and X_i corresponds to the i th word in the corpus. They focus on the entropy of a word conditioned on its context, i.e., $X_i|X_1 = w_1, \ldots, X_{i-1} = w_{i-1}$, and decompose the context into two parts: the global context C_i that refers to all the words from preceding sentences, and the local context L_i that refers to all the preceding words within the same sentence as X_i. Thus, the conditioned entropy of X_i is also decomposed into two terms (see the right side of Equation 1): the local measure of entropy (first term), and the mutual information between the word and global context (second term).

$$H(X_i|C_i, L_i) = H(X_i|L_i) - I(X_i, C_i|L_i) \quad (1)$$

The constancy rate principle predicts that the left side of Equation 1 should be constant as i increases. Because $H(X_i|C_i, L_i)$ itself is difficult to estimate (because it is hard to define C_i mathematically), and that the mutual information turn $I(X_i, C_i|L_i)$ is known to increase with i, the whole problem becomes examining whether the local measure of entropy $H(X_i|L_i)$ also increases with i. Genzel and Charniak (2002) have confirmed this prediction by showing that $H(X_i|L_i)$ does increase with i within multiple genres of written text of different languages.

The constancy rate principle also leads to an interesting prediction about the relationship between entropy change and topic shift in text. Generally, a sentence that initiate a shift in topic will have lower mutual information between its context, because the previous context provides little information to the new topic. Thus, a topic shift corresponds to the drop of the mutual information term $I(X_i, C_i|L_i)$. Then in order to keep constancy of the left term as predicted by the principle, the entropy term needs to decrease when a topic shift happens. Genzel and Charniak (2003) verified this prediction by showing that paragraph-starting sentences have lower entropy than non-paragraph-starting ones, with the assumption that a new paragraph often indicates a topic shift in written text. More recently, latent topic modeling (Qian and Jaeger, 2011) showed that lower sentence entropy was associated with topic shifts.

Genzel and Charniak's work has been extended to integrate non-linguistic information into the principle. Doyle and Frank (2015) leveraged Twitter data to find further support to the constancy rate principle: the entropy of message gradually increases as the context builds up, and it sharply goes down when there is a sudden change in the non-linguistic context (Baseball world series news reports, Doyle and Frank, 2015). *Uniform Information Density* (UID) (Jaeger and Levy, 2006) extends the principle in a framework that governs how people manage the amount of information in language production, from lexical levels to all levels of linguistic representations, e.g., syntactic levels. Its core idea is that people avoid salient changes in the density of information (i.e., amount of information per amount of linguistic signal) by making specific linguistic choices under certain contexts (Jaeger, 2010).

2.2 Topic shift in dialogues

As a conversation unfolds, topic changes naturally happen when a current topic is exhausted or a new one occurs, which is referred to as *topic shift* in the field of *Conversation Analysis* (CA) (Ng and Bradac, 1993; Linell, 1998). In CA, the basic unit of topical structure analysis in dialogue is *episode*, which refers to a sequence of speech events that are "about" something specific in the world (Linell, 1998, ch 10, p 187). Here, to be precise, we use the term *topic episode*.

According to related theories in CA, the for-

Table 1: Basic statistics of corpora

Statistics	Switchboard	BNC
# of dialogues.	1126	1346
Avg # of turns in dialogue	109	52
Avg # of sentences in dialogue	141	70

mation of topic episode is a joint accomplishment from two speakers and a product of initiatives and responses (Linell, 1990). When establishing a new topic jointly, one speaker first produces an initiatory contribution that introduce a "candidate" topic, and the other speaker makes a response that shares his perspective on that (Linell, 1998). From the information theoretic point of view, the *initiator* of a new topic plays a role of introducing *novelty* or *surprisal* into the context, while the other speaker, the *responder*, is more of a *commenter* or *evaluator* of information, who does not contribute as much in terms of novelty.

Since previous studies have shown that the decrease of sentence entropy is correlated with topic shifts in written text (Genzel and Charniak, 2003; Qian and Jaeger, 2011), it is reasonable to expect the same effect to be present at the boundaries of topic episodes in dialogue. Furthermore, considering the *initiator* vs. *responder* discrepancy in speaker roles, we expect their entropy change patterns also to be different.

3 Overall Trend of Entropy in Dialogue

In this section we examine whether the overall entropy increase trend is present in dialogue text.

3.1 Corpus data

The Switchboard corpus (Godfrey et al., 1992) and the British National Corpus (BNC) (BNC, 2007) are used in this study. Switchboard contains 1126 dialogues by telephone between two native North-American English speakers in each dialogue. We use only a subset of BNC (spoken part) that contain spoken conversations with exactly two participants, so that the dialogue structures are consistent with Switchboard.

3.2 Computing Entropy of One Sentence

We use language model to estimate the sentence entropy, which is similar to Genzel and Charniak (2003)'s method. A sentence is considered as a sequence of words, $W = \{w_1, w_2, \ldots, w_n\}$, and its per-word entropy is estimated by:

$$H(w_1 \ldots w_n) = -\frac{1}{n} \sum_{w_i \in W} \log P(w_i | w_1 \ldots w_{i-1})$$

where $P(w_i|w_1 \ldots w_{i-1})$ is estimated using a trigram language model. The model is trained using Katz backoff (Katz, 1987) and Lidstone smoothing (Chen and Goodman, 1996).

For the two corpora respectively, we extract the first 100 sentences from each conversation, and apply a 10-fold cross-validation, i.e., dividing all the data into 10 folds. Then we choose each fold as the testing set, and compute the entropy of each sentence in it, using the language model trained against the rest of the folds.

3.3 Eliminating sentence length effects

Intuitively, longer sentences tend to convey more information than short ones. Thus, the per-word entropy of a sentence should be correlated with the sentence length, i.e., the number of words. This correlation is confirmed in our data by calculating the Pearson correlation between the per-word entropy and sentence length: For Switchboard, $r = 0.258, p < 0.001$; for BNC, $r = 0.088, p < 0.001$.

Sentence length is found to vary with its relative position in text (Keller, 2004). Thus, in order to truly examine the variation pattern of sentence entropy within dialogue, we need to eliminate the effect of sentence length from it. We calculate a *normalized entropy* that is independent of sentence length in the following way. (This method is used by Genzel and Charniak (2003) to get the length-independent *tree depth* and *branching factor* of sentence.) First, we compute $\bar{e}(n)$, the average per-word entropy of sentences of the same length n, for all lengths ($n = 1, 2, \ldots$) that have occurred

$$\bar{e}(n) = {}^{1}\!/_{|L(n)|} \sum\nolimits_{s \in L(n)} e(s)$$

where $e : S \rightarrow \mathbb{R}$ is the original per-word entropy of a sentence s, and $L(n) = s|l(s) = n$ is the set of sentences of length n. Then we compute the sentence-length adjusted entropy measure that we want by

$$e'(s) = \frac{e(s)}{\bar{e}(n)}$$

This normalized entropy measure sums up to 1, and is not sensitive to sentence length. In later part

of this paper, we demonstrate our results in both entropy and normalized entropy because the former is the direct measure of information content.

3.4 Results

We plot the per-word entropy and normalized entropy of sentence against its *global position*, which is the sentence position from the beginning of the dialogue (Figure 1). It can be seen that both measures increase with global position. BNC shows larger slope than Switchboard, and the latter has a flatter curve but sharper increase at the early stage of conversations.

To test the reliability of the observed increasing trend, we fit linear mixed-effect models using entropy and normalized entropy as response variables, and the global position of sentence as predictor (fixed effect), with a random intercept grouped by distinct dialogues. The `lme4` package in `R` is used (Bates et al., 2014). The results show that the fixed effects of global position are significant for both measures in both corpora: Entropy in Switchboard, $\beta = 4.2 \times 10^{-3},\ p < 0.001$; normalized entropy in Switchboard, $\beta = 5.9 \times 10^{-4},\ p < 0.001$; entropy in BNC, $\beta = 1.5 \times 10^{-2},\ p < 0.001$; normalized entropy in BNC, $\beta = 1.4 \times 10^{-3},\ p < 0.001$).

In particular, since the curves of Switchboard seem flat after a boost in the early phase (between 0 to 5 in global position), we fit extra models to examine whether the entropy increase for global positions larger than 10 is significant. The long-term changes are reliable, too: Entropy, $\beta = 3.4 \times 10^{-3},\ p < 0.001$; normalized entropy, $\beta = 5.1 \times 10^{-4},\ p < 0.001$.

In sum, we find increasing entropy over the course of the whole dialogue. These findings are consistent with previous findings on written text.

4 Topic Shift and Speaker Roles

Since the topic structure of dialogue differs from written text, it is our interest to investigate how this difference affects the sentence entropy patterns. First, we identify the boundaries of topic episodes, and examine the presence of entropy drop effect at the boundaries. Second, we differentiate the speakers' roles in initiating the topic episode, i.e., *initiator* vs. *responder*, and compare their entropy change patterns within the episode.

4.1 Topic segmentation

There are multiple computational frameworks for topic segmentation, such as the Bayesian model (Eisenstein and Barzilay, 2008), Hidden Markov model (Blei and Moreno, 2001), latent topic model (Blei et al., 2003) etc. Considering that performance is not the prior requirement in our task, and also to avoid being confounded by segmentation method that utilize entropy measure per se, we use a less sophisticated cohesion-based TextTiling algorithm (Hearst, 1997) to carry out topic segmentation.

TextTiling algorithm inserts boundaries into dialogue as a sequence of sentences. We treat the segments between those boundaries as topic episodes. For each episode within a dialogue, we assign it a unique *episode index*, indicating its relative position in the dialogue (e.g., from 1 to N for a dialogue that contains N episodes). For each sentence, we assign it a *within-episode position*, indicating its relative position within the topic episode.

In Figure 2 we plot the entropy (and normalized) of sentence against the within-episode positions, grouped by episode index. Due to the space limit, we only present the first 6 topic episodes and the first 10 sentences in each episode. It can be seen that entropy drops at the beginning of topic episode, and then increases within the episode.

To examine the reliability of the entropy increase within topic episodes, we fit linear mixed effect models using entropy (and normalized) as response variables, and the within-episode position of sentence as predictor (fixed effect), with a random intercept grouped by the unique episode index of each topic episode. We find a significant fixed effect of within-episode position on both measures for both corpora: Entropy in Switchboard, $\beta = 5.9 \times 10^{-4},\ p < 0.001$; normalized entropy in Switchboard, $\beta = 4.5 \times 10^{-3},\ p < 0.001$; entropy in BNC, $\beta = 2.5 \times 10^{-2},\ p < 0.001$; normalized entropy in BNC, $\beta = 3.0 \times 10^{-3},\ p < 0.001$.

Our results show that when we treat the sentences in dialogue indiscriminately, their entropy change patterns at topic boundaries are consistent with previous findings on written text.

4.2 Identifying topic initiating utterances

Having dialogue segmented into topic episodes, our next step is to identify each speaker's role in initiating the topic. According to the theories

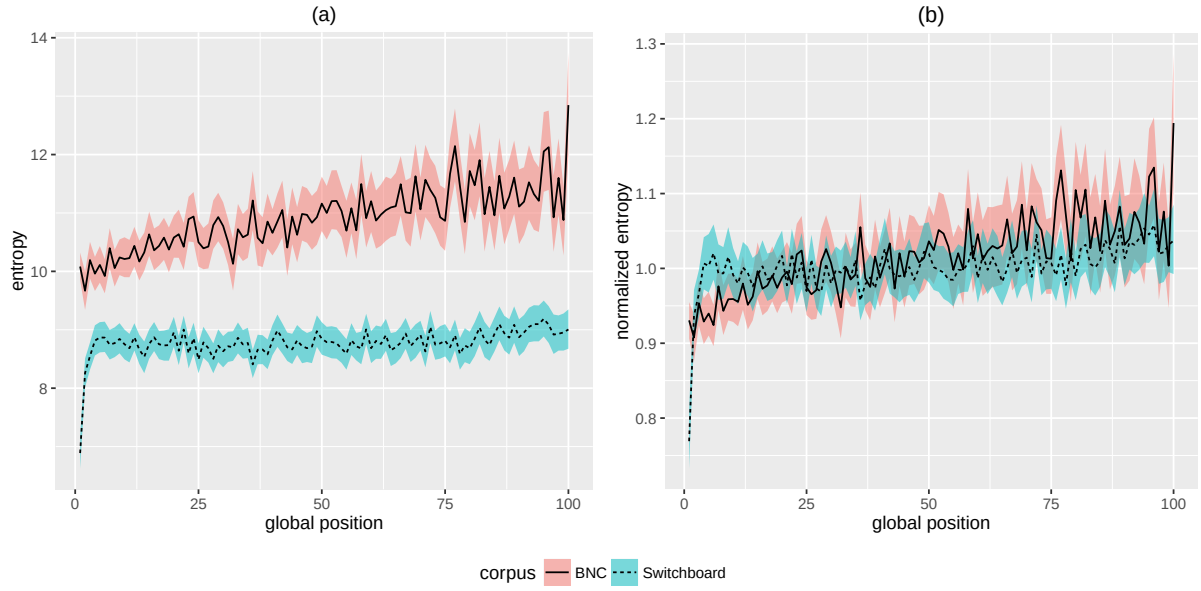


Figure 1: Entropy (a) and normalized entropy (b) against global position of sentences (from 1 to 100). Shadow area indicates 95% bootstrapped Confidence Interval.

reviewed in Section 2.2, the key to identify the speaker roles is to identify who produces the initiatory "candidate" topic. To be convenient, we use the term *topic initiating utterance* (TIU) to refer to the very first utterance produced by the *initiator* to bring up the new topic. Here, we give an empirical operational definition of TIU.

Since we treat dialogue as a series of sentences, and apply the TextTiling algorithm to insert topic boundaries indiscriminately (without differentiating whether adjacent sentences are from the same speaker or not), it results in two types of topic boundaries: *Within-turn* boundaries, the ones located in the middle of a turn (i.e., from one speaker). *Between-turn* boundaries, the ones located at the gap between two different turns (i.e., from two speakers). Our survey shows that in Switchboard 27.2% of the topic boundaries are within turns, and 72.8% are between turns. For BNC the two proportions are 41.2% and 58.8% respectively.

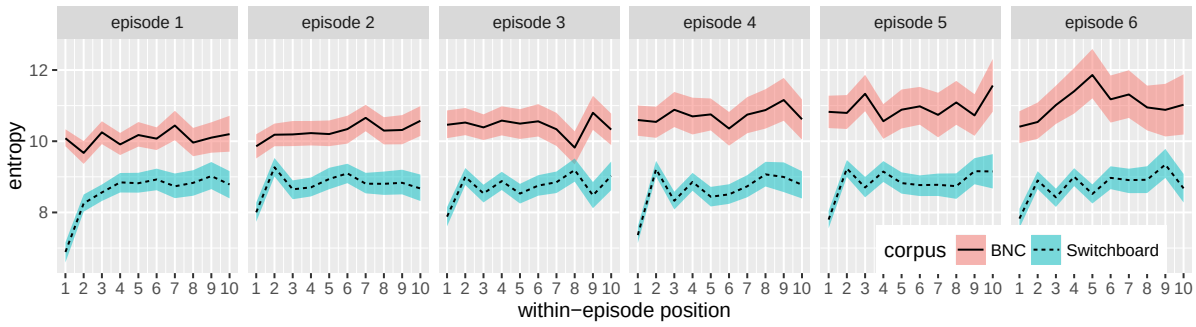
Intuitively, a within-turn topic boundary suggests that the speaker of the current turn is initiating the topic shift. On the other hand, a between-turn boundary suggests that the following speaker who first gives *substantial* contribution is more likely to be the initiator of the next topic. Following this intuition, for within-turn boundaries, we define TIU as the rest part of current turn after the boundary. For between-turn boundaries, we define TIU as the whole body of the next *relatively long* turn after the boundary, whose length is larger than N words. Note that the determination of threshold N is totally empirical, because our goal is to identify the most *probable* TIU, based on the intuition that longer sentences *tend* to contain more information, and thus are more likely to initiate a new topic. For the results shown later in this paper, we use $N = 5$, and our experiments draw similar results for $N \geq 5$. The operational definition of TIU is demonstrated in Figure 3.

4.3 The effect of topic initiator vs. responder

Based on the operational definition of topic initiating utterance (TIU), we distinguish the two speakers' roles in each topic segment: the author of TIU is the *initiator* of the current topic, while the other speaker is the *responder*.

Again, we plot the sentence entropy (and normalized) against the within-episode position respectively, this time, grouped by speaker roles (*initiator* vs. *responder*) in Figure 4. It can be seen that at the beginning of a topic, initiators have significantly higher entropy than responders. As the topic develops, the initiators' entropy decreases (Figure 4a) or stays relatively steady (Figure 4b), and the responder's entropy increases. Together they form a convergence trend within topic episode.

We use standard linear mixed models to examine the convergence trend observed, i.e., to test



(a) Entropy vs. within-episode position

(b) Normalized entropy vs. within-episode position

Figure 2: Entropy (a) and normalized entropy (b) against within-episode position grouped by episode index. The x-axis in each block indicates the within-episode position of sentence. The number 1 to 6 on top of the blocks are episode indexes. Shadow area indicates 95% bootstrapped Confidence Interval

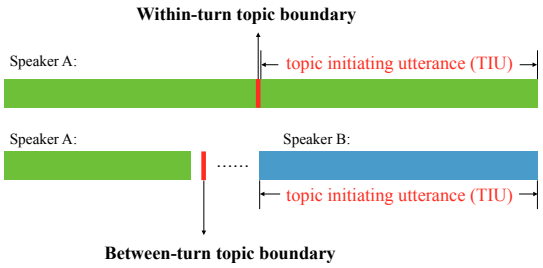
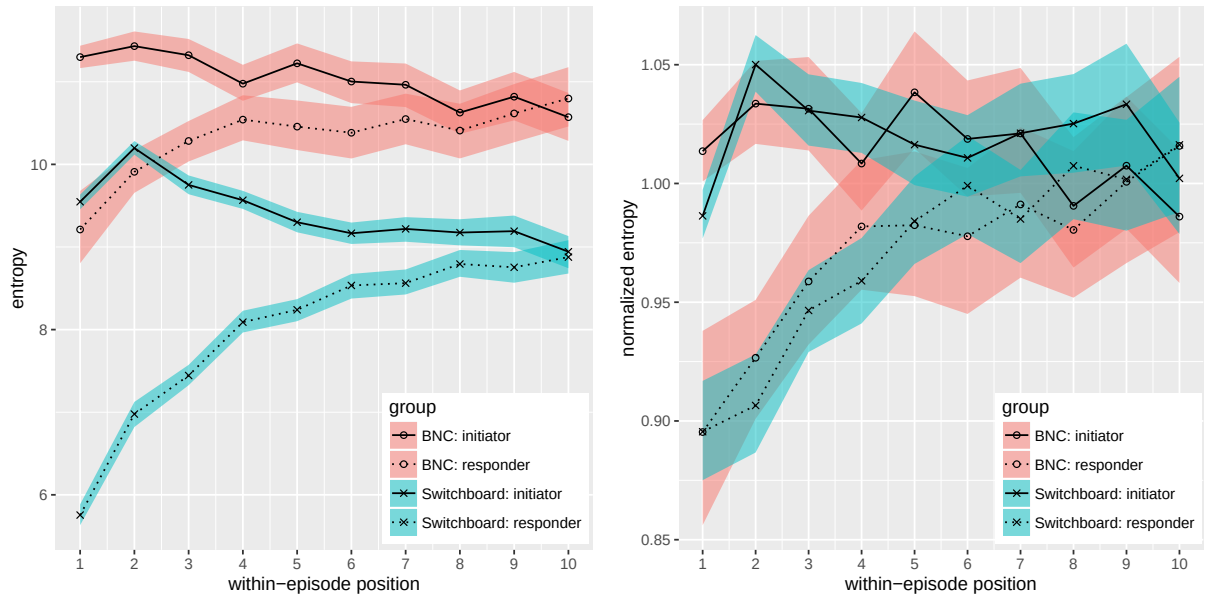


Figure 3: Operational definition of topic initiating utterances (TIUs). The red vertical bars indicate the topic boundaries placed using TextTiling. A complete horizontal bar of one color represents a turn from one speaker (green for speaker A and blue for speaker B). The upper line shows the case of within-turn topic boundary, and the lower line shows the case of between-turn topic boundary.

whether the initiators' entropy reliably decreases and whether the responders' entropy reliably increases. Models are fitted for initiators and responders respectively, using the entropy (and normalized) as response variables, and the within-episode position as predictor (fixed effect), with a random intercept grouped by the unique episode index. Our models show that for the entropy measure, the fixed effect of within-episode position is reliably negative for initiators (Switchboard, $\beta = -3.6 \times 10^{-2}$, $p < 0.001$; BNC, $\beta = -2.9 \times 10^{-2}$, $p < 0.05$) and reliably positive for responders (Switchboard, $\beta = 3.3 \times 10^{-1}$, $p < 0.001$; BNC, $\beta = 1.4 \times 10^{-1}$, $p < 0.001$). For the normalized entropy measure, the fixed effect of within-episode position is insignificant for initiators, which means there is neither increase nor decrease, and is reliably positive for responders (Switchboard, $\beta = 1.4 \times 10^{-2}$, $p < 0.001$; BNC, $\beta = 1.2 \times 10^{-2}$, $p < 0.001$). Thus, the convergence trend is confirmed.

The entropy change patterns of topic *initiators* (decrease or remain constant within topic episode) are inconsistent with previous findings that assert an entropy increase in written text (Genzel and



(a) (b)

Figure 4: Entropy (a) and normalized entropy (b) against within-episode position grouped by speaker roles (*topic initiator* vs. *topic responder*)

Charniak, 2002, 2003), which will be discussed in the next section.

5 Discussion

5.1 Summary

Our main contribution is that we find new entropy change patterns in dialogues that are different from those in written text. Specifically, when distinguishing the speakers' roles by topic *initiator* vs. *responder*, we see that the initiator's entropy decreases (or remain steady) whilst the responder's increases within a topic episode, and together they form a convergence pattern. The partial trend of entropy decrease in topic initiators seems to be contrary to the principle of entropy rate constancy, but as we will discuss next, it is actually an effect of the unique topic shift mechanism of dialogues that is different from written text, which does not violate the principle.

From an information theoretic perspective, we view dialogue as a process of information exchange, in which the interlocutors play the roles of information provider and receiver, interactively within each topic episode.

Beyond differences in speaker roles, we do observe that sentence entropy increases with its global position in the dialogue, which is consistent with written text data (Genzel and Charniak, 2002, 2003; Qian and Jaeger, 2011; Keller, 2004). Thus, overall speaking, spoken dialogue do follow the general principle of entropy rate constancy.

5.2 Dialogue as a process of information exchange

By combining topic segmentation techniques and fine-grained discourse analysis, we provide a new angle to view the big picture of human communication: the perspective of how information is distributed between different speakers.

One critical difference between written text and spoken text in conversation is that there is only one direct input source of information in the former, i.e., the author of the text, but for the latter, there are multiple direct input sources, i.e., the multiple speakers. That means, when language production is treated as a process of choosing proper words (or other representations) within a context, the definition of "context" is different between the two categories of text. In written language (see Equation 1 in Section 2), C_i, the global context of a word X_i, is assumed to be all the words in preceding sentences. This is a reasonable assumption, because when one author is writing a complete piece of text, he may organize information smoothly to keep the entropy rate constant. Within a dialogue, for any upcoming utterance, all preceding utterances together can be viewed as the *shared* context for the two speakers. To help us un-

derstand the nature of this shared context, we propose the following mental experiment. Suppose we, as researchers and "super-readers", observe the transcript of a dialogue between interlocutors A and B. To us, all utterances are based upon the context of previous ones, which is why we can observe consistent entropy increase within the whole dialogue (Figure 1 in Section 3). Also, to us, a new topic episode in dialogue is just like a new paragraph in written text, within which we can observe steady entropy increase without differentiating the utterances from the two speakers. By contrast, let's look at the context used by the two speakers. They will not necessarily leverage the preceding utterances as a coherent context. A *topic initiator* introduces new information from a context outside of the dialogue. Therefore the mutual information between the initiator's current sentence and the previous context is reduced, which causes the sentence entropy to start high before decreasing. On the other side, a *topic responder* relies much on the previous shared context (because he is not an active topic influencer). The responder is dynamically updating the context as the initiator pours new information into the mix. This causes the mutual information with the previous context to be high, and thus the sentence entropy start low before increasing again.

We think that the respective cognitive load in the topic responder imposed by following the other speaker in a new topic direction may be complemented by reduced information at the language level. This is, again, compatible with a cognitive communication framework that imposes a tendency to limit or keep constant overall information levels. It is also an example of extralinguistic information that causes complementary entropy changes in a speaker's language (cf., Doyle and Frank, 2015).

5.3 Dialogue as a process of building up common ground

Our findings can also be explained by a theory of *grounding* (Clark and Brennan, 1991; Clark, 1996) of communication. Dialogue can be seen as a joint activity during which multiple speakers contribute alternatively to build *common ground* (Clark and Brennan, 1991). Common ground can be understood as the mutual knowledge shared between interlocutors.

Clark (1996) proposes that joint activities have a number of characteristics: First, participants play different roles in the activity. Second, a major activity is usually comprised of sequences of subactivities, and the participants' role may differ from sub-activity to next. Third, to achieve the goal of the activity, it requires coordination between participants of different roles.

In our design, the local roles of *topic initiator* vs. *topic responder* correspond to roles suggested by the joint-activity theory. The initiator sets up the dominant goal of the sub-activity, i.e., developing a new *topic episode*, and the responder joins him or her in order to achieve the goal. The converging sentence entropy indicates that the mutual knowledge between them is accumulating, i.e., the common ground is being gradually built up. Once the goal is achieved, i.e., the current topic is fully developed, a new goal will emerge, and a new common ground needs to be built again, which is sometimes accompanied by a change in participants roles.

5.4 Convergence of linguistic behaviors

One mechanism that may lead to the convergence of sentence entropy may be the interactive alignment of linguistic features between speakers (Pickering and Garrod, 2004); repeating words and syntactic structure leads to increased similarity. The entropy-converging pattern also reflects the convergence of higher-level dialogical behavior, say, speakership occupancy; the discrepancy between the two speakers' roles gradually becomes smaller, i.e., the "speaker" becomes more of a "listener", and vice versa. A psychologist might treat the fragmented topic episodes in dialogues as the locus where interlocutors build temporarily shared understanding (Linell, 1998), through the process of "synchronization of two streams of consciousness" (Schutz, 1967).

6 Conclusion

In this study, we validate the principle of entropy rate constancy in spoken dialogue, using two common corpora. Besides the results that are consistent with previous findings on written text, we find new entropy change patterns unique to dialogue. Speakers that actively initiate a new topic tend to use language with higher entropy compared to the language of those who passively respond to the topic shift. These two speaker's respective entropy

levels converge as the topic develops. A model of this phenomenon may provide explanations from the perspectives of information exchange, common ground building, and the convergence of linguistic behaviors in general.

With this, we put forward what we think is a new perspective to analyzing dialogue. As much dialogue happens for the purpose of information exchange, loosely defined, it makes sense to apply information-theoretic models to the semantics as well as the form of speaker's messages. The quantitative approach taken here augments rather than supplants speech acts (Searle, 1976), identifying who leads the dialogic process by introducing topics and shifting them.

Furthermore, our approach actually provides a unified perspective of dialogue that combines Grounding theory (Clark and Brennan, 1991) and Interactive Alignment (Pickering and Garrod, 2004). These two models are often described as opposite; by applying each theory to the dialogic structure between and within topic episodes, we find both of them can explain our findings. The entropy measure of information content quantifies interlocutors' contributions to common ground and also allows us to show convergence patterns.

This unified information-theoretic perspective may eventually allow us to identify further systematic patterns of information exchange between dialogue participants. There is, of course, no reason to think that multi-party dialogue should work differently; we leave the empirical examination as an open task.

Acknowledgments

This work has been funded by the National Science Foundation under CRII IIS grant 1459300.

References

Douglas Bates, Martin Mächler, Ben Bolker, and Steve Walker. 2014. Fitting linear mixed-effects models using lme4. *arXiv preprint arXiv:1406.5823* .

David M Blei and Pedro J Moreno. 2001. Topic segmentation with an aspect hidden markov model. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New Orleans, LA, USA, pages 343–348.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *The Journal of Machine Learning Research* 3:993–1022.

BNC. 2007. The British National Corpus, version 3 (BNC XML Edition).

Stanley F Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, pages 310–318.

Herbert H Clark. 1996. *Using language*. Cambridge University Press.

Herbert H Clark and Susan E Brennan. 1991. Grounding in communication. *Perspectives on socially shared cognition* 13(1991):127–149.

Gabriel Doyle and Michael C Frank. 2015. Shared common ground influences information density in microblog texts. In *Proceedings of NAACL-HLT*. Denver, CO, USA.

Jacob Eisenstein and Regina Barzilay. 2008. Bayesian unsupervised topic segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Honolulu, HI, USA, pages 334–343.

Dmitriy Genzel and Eugene Charniak. 2002. Entropy rate constancy in text. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Philadelphia, PA, USA, pages 199–206.

Dmitriy Genzel and Eugene Charniak. 2003. Variation of entropy and parse trees of sentences as a function of the sentence number. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sapporo, Japan, pages 65–72.

John J Godfrey, Edward C Holliman, and Jane McDaniel. 1992. Switchboard: Telephone speech corpus for research and development. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. IEEE, volume 1, pages 517–520.

Marti A Hearst. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational linguistics* 23(1):33–64.

T Florian Jaeger. 2010. Redundancy and reduction: Speakers manage syntactic information density. *Cognitive Psychology* 61(1):23–62.

T Florian Jaeger and Roger P Levy. 2006. Speakers optimize information density through syntactic reduction. In *Advances in Neural Information Processing Systems*. pages 849–856.

Dan Jurafsky and James H Martin. 2014. *Speech and language processing*. Pearson.

Slava M Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *Acoustics, Speech and Signal Processing, IEEE Transactions on* 35(3):400–401.

Frank Keller. 2004. The entropy rate principle as a predictor of processing effort: An evaluation against eye-tracking data. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Barcelona, Spain, volume 317, page 324.

Per Linell. 1990. *The power of dialogue dynamics*. Harvester Wheatsheaf.

Per Linell. 1998. *Approaching dialogue: Talk, interaction and contexts in dialogical perspectives*, volume 3. John Benjamins Publishing.

Sik Hung Ng and James J Bradac. 1993. *Power in language: Verbal communication and social influence*. Sage.

Martin J Pickering and Simon Garrod. 2004. Toward a mechanistic psychology of dialogue. *Behavioral and Brain Sciences* 27(02):169–190.

Ting Qian and T Florian Jaeger. 2011. Topic shift in efficient discourse production. In *Proceedings of the 33rd Annual Conference of the Cognitive Science Society*. Boston, MA, USA, pages 3313–3318.

Alfred Schutz. 1967. *The phenomenology of the social world*. Northwestern University Press.

John R Searle. 1976. A classification of illocutionary acts. *Language in Society* 5(01):1–23.

Claude Elwood Shannon. 1948. A mathematical theory of communication. *The Bell System Technical Journal* 27:379–423.

David Traum. 2003. Issues in multiparty dialogues. In *Advances in Agent Communication*, Springer, pages 201–211.

Finding the Middle Ground - A Model for Planning Satisficing Answers

Sabine Janzen
Saarland University
Saarbrücken, Germany
janzen@iss.uni-saarland.de

Wolfgang Maaß
Saarland University
Saarbrücken, Germany
maass@iss.uni-saarland.de

Tobias Kowatsch
University of St. Gallen
St. Gallen, Switzerland
tobias.kowatsch@unisg.ch

Abstract

To establish sophisticated dialogue systems, text planning needs to cope with congruent as well as incongruent interlocutor interests as given in everyday dialogues. Little attention has been given to this topic in text planning in contrast to dialogues that are fully aligned with anticipated user interests. When considering dialogues with congruent and incongruent interlocutor interests, dialogue partners are facing the constant challenge of finding a balance between cooperation and competition. We introduce the concept of fairness that operationalize an equal and adequate, i.e. equitable satisfaction of all interlocutors' interests. Focusing on Question-Answering (QA) settings, we describe an answer planning approach that support fair dialogues under congruent and incongruent interlocutor interests. Due to the fact that fairness is subjective per se, we present promising results from an empirical study (N=107) in which human subjects interacted with a QA system implementing the proposed approach.

1 Introduction

For building dialogue systems that cope with contradictions and individual interests of dialog partners, text planning is required to process incongruent and congruent interests of interlocutors. So far, research on dialogue systems focusses on supporting dialogues that are fully aligned with anticipated user interests, e.g., (Hovy, 1991; Grosz and Kraus, 1996; Moore and Paris, 1993; Lochbaum, 1998; Rich and Sidner, 1997), and, thus, maximizing cooperativeness (Bunt and Black, 2000, 191 p. 5). Few approaches exist that investigate text planning with pure conflict, e.g., (Jameson et al., 1994; Hadjinikolis et al., 2013; Black and Atkinson, 2011; Prakken, 2006). When considering dialogues with congruent as well as incongruent interlocutors interests, dialogue partners are facing the constant challenge of finding a balance between cooperation and competition (Parikh, 2010). We introduce the concept of *fairness* that operationalize an equal and adequate, i.e. equitable satisfaction of all interlocutors' interests (Oxford Dictionaries, 2016). Focusing on Question-Answering (QA) settings, we describe an answer planning approach that support fair dialogues under congruent and incongruent interests of interlocutors. Due to the fact that fairness is subjective per se, we present results from an empirical study in which human subjects interacted with a QA system in various dialogue settings.

When determining appropriate answers in text planning, approaches range from (1) wrong answer avoidance concepts technically checking the correctness of answers, e.g., Dong et al. (2011), and (2) opponent models in persuasion dialogues for choosing most suitable arguments, e.g., Hadjinikolis et al. (2013), to (3) the prediction of emotions of interlocutors to generate answers, e.g., Hasegawa et al. (2013). Here, related work is relevant that focuses on the determination of appropriate answers by processing concepts like users' intentions (e.g., Levelt (1993)), desires (e.g., Rao & Georgeff (1995)), preferences (e.g., Li et al. (2013)), objectives (e.g., Schelling (1960)) and goals (e.g., Traum et al. (2008)) which we will hereafter subsume under the term motives. *Motives* refer to objectives or situations that interlocutors would like to accomplish, e.g., to find the best price when shopping. According to the belief-desire-intention model, motives can be described as desires in the sense of a motivational state (Georgeff et al., 1998). Motives do not in-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 547–557,
Berlin, Germany, August 7-12, 2016.

volve the mandatory purpose of being recognizable by other participants; so they are equivalent with the concept of intentions in (Levelt, 1993). Regarding the processing of congruent and incongruent motives, existing approaches rather focus on motives of single interlocutors or on joint motives, e.g., Paquette (2012), Li et al. (2013). In the following, the aggregation of congruent and incongruent interlocutor motives in dialogues will be described as *mixed motives*.

In this work, we propose a model that formalizes answer planning as psychological game (Bjorndahl et al., 2013) embedded in text planning approaches (Mann and Thompson, 1986; Moore and Paris, 1993) for creating dialogues perceived as fair by all interlocutors. Since traditional formalization of motives by means of utility functions is not sufficient to handle complex interactions as given in the considered dialogue setting (Bjorndahl et al., 2013), psychological games enrich classical game settings with user models, i.e. in our case explicit representations of mixed motives. One appeal of the model is the consideration of answer planning as psychological game that lifts the process of finding appropriate answers from the short-term linguistic level to the long-term motive level in contrast to other approaches (van Deemter, 2009; Stevens et al., 2015). Interlocutors do not have preferences for answers, but try to satisfy motives. So, we assume that this approach enables a more sophisticated simulation of human behavior in mixed motive interactions as well as the establishment of "cooperativeness in response formulation" (Bunt and Black, 2000, p. 5) for creating dialogues perceived as fair. By exemplifying the model within a QA system as natural language sales assistant for conducting sales dialogues, we were able to evaluate the proposed approach in an empirical user study (N=107) in terms of perceived fairness of created dialogues with promising results.

2 Planning Answers given Mixed Motives

Adopting a computational pragmatics perspective, we intend to compute relevant linguistic aspects of answers based on contextual aspects given by mixed motives (Bunt and Black, 2000, p. 3). When searching for answers that support an equitable satisfaction of mixed motives during dialogue, Ω represents the solution space with potential answers. An objective function $f : \Omega \rightarrow \mathbb{R}$ assigns values to all answers $x \in \Omega$ for representing their potential in satisfying motives of interlocutor $i \in I$. Of course, all interlocutors I prefer answers x that satisfy best their motives; $x_{\mathrm{a}} \succ x_{\mathrm{b}} \Leftrightarrow f(x_{\mathrm{a}}) > f(x_{\mathrm{b}})$. So, the goal would be to find an answer $x \in \Omega$ with highest satisfaction of motives $f(x)$ of interlocutor $i \in I$ in the sense of an optimal solution x^*; i.e. $f(x^*) = max\{f(x)|x \in \Omega\}$. But, in order to achieve fair outcomes regarding an equitable, i.e. equal and adequate satisfaction of mixed motives, this definition is not sufficient. First, decision making takes places in the context of social dialogue interaction, i.e. answers have to be selected based on multiple objective functions since motives of all interlocutors $i \in I$ shall be satisfied; $f_{\mathrm{i}} : \Omega \rightarrow \mathbb{R}$. For capturing the aspect of equal motive satisfaction, the potential of answers has to be represented absolutely and relatively. In other words, the performance of an answer in satisfying motives of an interlocutor $i \in I$ is combined with its performance in satisfying motives of counterparts $-i \in I$; $max\{f_{\mathrm{i,\text{-}i}}(x)|x \in \Omega\}$. Second, the aforementioned conflict between cooperation and competition needs to be solved adequately. Since it is impossible to find an answer satisfying all motives of all interlocutors at any time in the dialogue, we search for a compromise in form of a solution i.e. an answer x^+ with a minimum quality s so that $f(x^+) \geq s$. Adopting the concept of satisficing by Simon (1956), an approach that attempts to find the best alternative available in contrast to optimal decision making, the goal is to find an answer $x \in \Omega$ with highest sufficient satisfaction of motives $f(x) \geq s$ of all interlocutors I in the sense of a satisficing solution x^+; i.e. $f(x^+) = max\{f_{\mathrm{i,\text{-}i}}(x) \geq s|x \in \Omega\}$.

3 Model for Planning Satisficing Answers

To capture these issues, we defined a model for planning satisficing answers in dialogues with mixed motives. In the considered setting, a user with motives poses questions to a QA system that takes the role of a proxy for indirect interlocutors, e.g., retailers in online shopping scenarios. The QA system adopts their motives and develops strategies to satisfy them. Adopted motives as well as user motives that are anticipated by the system represent mixed motives in the dialogue. Task of the system is to process these mixed mo-

tives with the objective to create a dialogue that is perceived as fair by all interlocutors after a finite number of question answer pairs. As full satisfaction of all motives of all interlocutors at any time in a mixed motive dialogue is not possible, the QA system has to find a compromise, i.e. it has to plan answers that satisfice mixed motives during dialogue. Let us start by describing an example dialogue between a customer and a retailer in a shopping scenario:

Q: *Is the range of this wifi router appropriate for a house with 3 floors?*

A: *In case of 3 floors, I would recommend an additional wifi repeater that got very good feedback by other customers. You can buy both router and repeater as a bundle with 15% discount.*

In this dialogue snippet, the customer intends to get comprehensive product information regarding the wifi router; the retailer also wants to satisfy informational needs of the customer to establish excellent services. Beyond these congruent motives, the retailer wants to increase revenue and to raise sales figures. A balance between mixed motives is found by giving information regarding the wifi router as well as preferences of other customers followed by a discounted bundle offer.

In order to implement this kind of behavior into dialogue systems, the model for planning satisficing answers separates linguistics from conceptual non-linguistic aspects (Traum and Larsson, 2003; Allen et al., 2001) and consists of three main modules: *linguistic module, mapper* and *mixed motive module* (cf. Fig. 1). The linguistic module takes care for handling user questions as input as well for generating answers as output. Essential components of the linguistic module are the linguistic intention model and flexible text planning technologies. For the latter, we apply text plans according to the Rhetorical Structure Theory (Mann and Thompson, 1986) in form of plan operators (Moore and Paris, 1993) for generating answers. Each plan operator consists of a single compulsive part, called nucleus, that is related with diverse optional text segments, mentioned as satellites. We assume that beside supporting the effect of the nucleus, satellites represent an opportunity to satisfice mixed motives during dialogue. Satellites are linked with entities of the linguistic intention model, means linguistic intentions that capture the intended effects, i.e. functions of satellites within answers (Grosz and Sidner, 1986). By means of second module - the mapper - linguistic intentions are mapped onto motives and vice versa (cf. Fig. 1). Therefore domain-specific knowledge about correlations between linguistic intentions and mixed motives is required that is induced by a domain configurator and has to be derived empirically. Last, the mixed motive module combines an explicit representation and situated processing of mixed motives (Cohen and Levesque, 1990) with a game-theoretical equilibrium approach (Nash, 1951) to establish a psychological game setting (Bjorndahl et al., 2013) (cf. Fig. 1). Our approach operates by assuming that interlocutors are rational. That means they act strategically and purposively in pursuit of their own motives that they try to maximally satisfy. Therefore, we assume that game theory is an adequate prospect to deliver the analytical tools for planning answers in the context of mixed motives. In game theory literature, equilibrium concepts are widely applied, e.g., Nash equilibrium (Nash, 1951). A Nash equilibrium is an outcome that holds because no involved actor has a rational incentive to deviate from it, i.e., the final result is "good enough" for all actors in the sense of a happy medium. Adapted to this work, this refers to a satisficing combination of motives at a particular time in the dialogue, that is good enough for planning an answer that supports equitable satisfaction of mixed motives.

3.1 Concepts

From a conceptual perspective, the model uses several core entities. First, we have players $p \in P$ that represent interlocutors I. Players have domain-specific motives m for participating in the dialogue. For each player $p \in P$, we assume a $MotiveSet$ that consists of individual motives, $IndM$, as well as of motives the player, i.e. the interlocutor anticipates from counterparts, $AntM$.

$$MotiveSet_{\mathrm{p}} = IndM_{\mathrm{p}} + AntM_{\text{-p}} \quad (1)$$

Mixed motives MM are represented by the non-redundant aggregation of $1 \ldots \mathrm{n}$ $MotiveSet$ of players $p \in P$ in the dialogue.

$$MM = \{MotiveSet_{\mathrm{p_1}} \ldots MotiveSet_{\mathrm{p_n}}\} \quad (2)$$

All motives $m \in MM$ are operationalized by means of real-valued weights for each player covered by a weight vector $\overrightarrow{Weight}_{\mathrm{m}}$. Motives are formed earlier and persist during dialogue, but

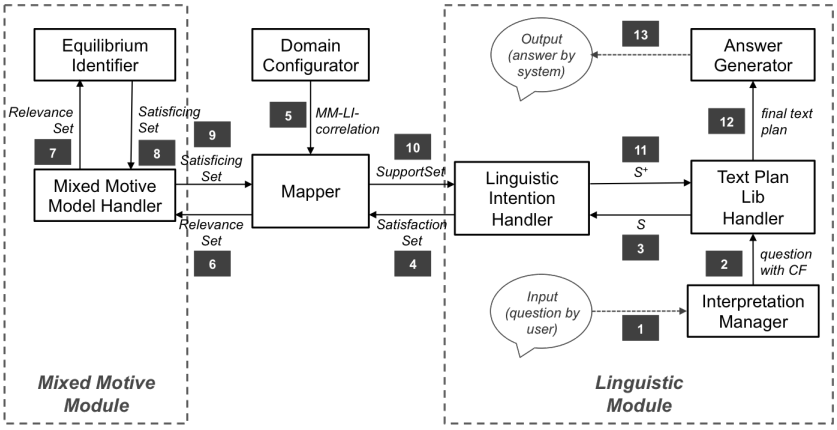


Figure 1: Model for planning satisficing answers in dialogues with mixed motives

players deliberate about weights of motives continuously (Bratman, 1987). The achievement and thereby satisfaction of motives is supported by linguistic intentions $li \in LI$ that are satisfied by satellites sat that are offered by plan operators and integrated into an answer. That means motives are achieved, if answers were given, that contributed to satisfaction of these motives.

3.2 Algorithm and example

For introducing the proposed approach, we will give an example course of satisficing answer planning starting with user question and ending with system answer. The description of the process will be supported by a model view marked with step numbers in Fig. 1 as well as by an algorithmic view in Alg. 1. In the example, we apply domain-specific knowledge that was derived empirically in the retailing domain. Although, in literature review, customer and retailer motives in sales dialogues were specified. Combinations of these motives were analyzed in simulated sales conversations between real retailers (N=3) and subjects acting as customers (N=12). Recorded as video files, conversations and identified motives were validated in a web-based user study (N=120) regarding their naturalness and relevance. Sales conversations were transcribed, aggregated to a text corpus and analyzed regarding question and answer structures. So, the domain-specific knowledge representation used in the example bases on results of this empirical analysis and covers all core model concepts introduced before: a mixed motive model with empirically derived default weights consisting of 19 customer and 4 retailer motives (cf. Tab. 1); 39 question and 33 answer schemata (McKeown, 1985), 31 plan operators (Moore and Paris, 1993), 21 satellites with 18 linguistic intentions (cf. Tab. 2) and 14 rhetorical relations (Hobbs, 1978; Hovy, 1993; Mann and Thompson, 1986), and exemplary product information.

Motive $m \in MM$	**Weight** $p_{\mathbf{a}}$	**Weight** $p_{\mathbf{b}}$
High level of reliability of product (m_{R})	1.90	1.00
Fair price of product (m_{FP})	0.70	0.00
Exclusive design of product (m_{ED})	0.53	1.00
Comprehensive product information (m_{CPI})	1.67	1.00
Improving customer relationship (m_{ICR})	0.00	4.00
Increase revenue (m_{IR})	4.00	4.00

Table 1: Extract of domain-specific mixed motive model with default weights for player (p_{a}) and player (p_{b}) representing customer and retailer

Imagine a sales conversation regarding consumer electronics between customer and retailer represented by player (p_{a}) and player (p_{b}). Sets of motives by players are equal regarding the motives included but differ in weights of individual and anticipated motives by players (cf. Tab. 1).

$$MM = MotiveSet_{\mathrm{p_a}} + MotiveSet_{\mathrm{p_b}} \quad (3)$$

$$MotiveSet_{\mathrm{p_a}} = IndM_{\mathrm{p_a}} + AntM_{\mathrm{p_b}}$$

$$MotiveSet_{\mathrm{p_b}} = IndM_{\mathrm{p_b}} + AntM_{\mathrm{p_a}}$$

The customer poses a question concerning products with a specific feature: *"How many tablets offer the wifi features 802.11A, 802.11B, 802.11G, 802.11n?"* Based on the identified question schema as well as the determined communicative function of the question, a dialogue system that instantiates the proposed model selects an appropriate plan operator (cf. Fig. 1, step 1 & 2).

Ling. Intention $li \in LI$	**supports** $m \in MM$	**Description**
Advantages (li_{A})	$\{m_{\text{ICR}}, m_{\text{FP}}, m_{\text{ED}}, m_{\text{IPD}}, m_{\text{R}}, m_{\text{HLS}}, m_{\text{ACB}}, m_{\text{SCD}}, m_{\text{I}}, m_{\text{HLP}}, m_{\text{PB}}, m_{\text{Q}}\}$	Integration of information about advantages of product(s) into answer
External_ Review (li_{ER})	$\{m_{\text{SI}}\}$	Presentation of customer reviews
My_Product (li_{MP})	$\{m_{\text{ICR}}, m_{\text{SP}}, m_{\text{R}}, m_{\text{HLC}}\}$	Mentioning products that could be interesting for customer
Functionality (li_{F})	$\{m_{\text{EU}}\}$	Extension of answer regarding product functions
Opinion (li_{O})	$\{m_{\text{HEM}}, m_{\text{SP}}, m_{\text{R}}, m_{\text{SI}}\}$	Integration of subjective (retailer) opinion into answer

Table 2: Extract of domain-specific linguistic intentions $li \in LI$ with supported motives and description

3.2.1 Definition of set S and determination of $SatisfactionSet$

In our case, a plan operator named NUMBER OF PRODUCTS is selected that offers an obligatory nucleus and a set S of four optional satellites (cf. Fig. 2):

$$S = \{sat_{\text{AAS}}, sat_{\text{VER}}, sat_{\text{DF}}, sat_{\text{EUP}}\} \quad (4)$$

Overall objective is to determine set S^{+} out of set S, that consists of satellites that - besides supporting the effect of the nucleus - contribute to satisficing mixes motives of customer and retailer during dialogue (cf. Alg. 1). According to (Grosz and Sidner, 1986; Moore and Paris, 1993), satellites are linked with linguistic intentions; i.e. they fulfill certain functions regarding the overall dialogue. Set S is sent to the linguistic intention handler that specifies the $SatisfactionSet$ (cf. Fig. 1, step 3 and Alg. 1, line 1-4). This set covers linguistic intentions that can be satisfied by satellites of set S (cf. Tab. 2):

$$SatisfactionSet = \{li_{\text{A}}, li_{\text{ER}}, li_{\text{F}}, li_{\text{MP}}\} \quad (5)$$

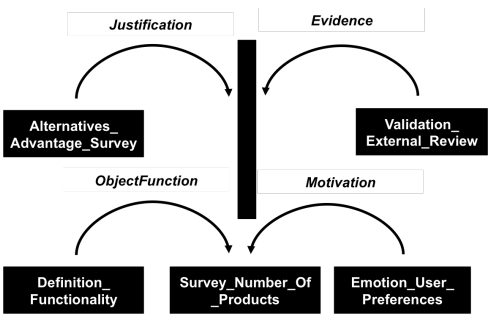


Figure 2: Plan operator NUMBER OF PRODUCTS

3.2.2 Mapping linguistic intentions onto mixed motives

Next, linguistic intentions have to be mapped onto motives. The m:n correlation between linguistic intentions and motives (Moore and Paris, 1993) is domain-specific, has to be specified empirically and is induced by the domain configurator (cf. Fig. 1, step 5). Each motive is supported by a set of linguistic intentions that contribute to the achievement of this motive (cf. Fig. 3). On the other hand, each linguistic intention can support the achievement of several motives. By processing

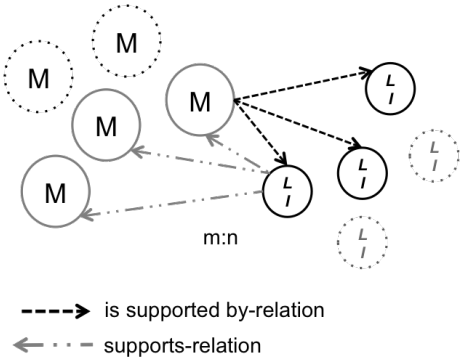


Figure 3: Correlations between motives (M) and linguistic intentions (LI)

the supports-relation between both concepts, the mapper specifies the $RelevanceSet$ based on the $SatisfactionSet$. The resulting $RelevanceSet$ represents all mixed motives relevant for planning the actual answer (cf. Alg. 1, line 5-8):

$$\begin{aligned} RelevanceSet = \{&m_{\text{Q}}, m_{\text{R}}, m_{\text{IPD}}, m_{\text{HCS}}, m_{\text{ACB}}, m_{\text{ICR}}, \\ &m_{\text{I}}, m_{\text{SCD}}, m_{\text{PB}}, m_{\text{EU}}, m_{\text{FP}}, m_{\text{ED}}, \\ &m_{\text{HLP}}, m_{\text{SI}}, m_{\text{SP}}, m_{\text{HLC}}\} \end{aligned} \quad (6)$$

3.2.3 Satisficing mixed motives

Having identified the $RelevanceSet$, we now intend to identify a satisficing combination of the involved motives. Therefore, the mapper sends the $RelevanceSet$ to the mixed motive model handler for specifying the $SatisficingSet$ that consists of motives that (1) are sufficiently interesting for all interlocutors (i.e. weighted positively), and (2) have preferably low conflict potential (i.e. small differences in player weights) (cf. Fig. 1, step 6). Satisficing mixed motives is considered as multi-player non-zero-sum game that is played for infinitely many rounds, more precisely pairs of user questions and system answers. In each round of the game, it has to be decided which motives

Algorithm 1 Determining set S^+ of satisficing satellites

Require: set of default satellites $S = \{sat_1 \dots sat_n\}$; set of players $P = \{p_1 \dots p_n\}$; set of mixed motives $MM = \{m_1 \dots m_n\}$; set of linguistic intentions $LI = \{li_1 \dots li_n\}$
Ensure: set of satisficing satellites $S^+ = \{sat_1 \dots sat_n\}$
1: Initialize $SatisfactionSet = \{li_1 \dots li_n \in LI | li.isSatisfiedBy(sat \in S)\}$
2: **for** $\forall sat \in S$ **do**
3: $SatisfactionSet \Leftarrow SatisfactionSet_{sat} \cup SatisfactionSet$
4: **end for**
5: Initialize $RelevanceSet = \{m_1 \dots m_n \in MM | m.isSupportedBy(li \in SatisfactionSet)\}$
6: **for** $\forall li \in SatisfactionSet$ **do**
7: $RelevanceSet \Leftarrow RelevanceSet_{li} \cup RelevanceSet$
8: **end for**
9: Determine $StrategySet \Leftarrow \mathcal{P}(RelevanceSet)$
10: Initialize $StrategyProfiles = \{\overrightarrow{s}_1 \dots \overrightarrow{s}_n\}$
11: **for** $\forall s \in StrategySet; \forall p \in P$ **do**
12: Calculate $LocalPayout(s)$
13: Define $\overrightarrow{s} = \{s_1 \dots s_n \in StrategySet | LocalPayout(s_p{}^* | s_{\text{-}p}) \geq LocalPayout(s_p | s_{\text{-}p})\}$
14: $StrategyProfiles.add(\overrightarrow{s})$
15: **end for**
16: **for** $\forall \overrightarrow{s} \in StrategyProfiles$ **do**
17: **if** $LocalPayout(s_p{}^* | s^*{}_{\text{-}p}) \geq LocalPayout(s_p | s^*{}_{\text{-}p})$ **then**
18: $\overrightarrow{s}^* \Leftarrow \overrightarrow{s}$
19: **end if**
20: **end for**
21: Determine $SatisficingSet = \{m_1 \dots m_n \in s \in \overrightarrow{s}^*\}$
22: **if** $SatisficingSet \neq \emptyset$ **then**
23: Initialize $SupportSet = \{li_1 \dots li_n \in LI | li.supports(m \in SatisficingSet)\}$
24: **for** $\forall m \in SatisficingSet$ **do**
25: $SupportSet \Leftarrow SupportSet_m \cup SupportSet$
26: **end for**
27: Return $S^+ = \{sat_1 \dots sat_n \in S | sat.satisfies(li \in SupportSet \cap SatisfactionSet)\}$
28: **else**
29: Return $S^+ = \{\emptyset\}$
30: **end if**

of the $RelevanceSet$ are selected as trigger for planning an answer that supports the creation of dialogues perceived as fair by all interlocutors. The equilibrium identifier specifies strategy sets $S_p = \{s_1 \dots s_n\}$ for all players P by generating the power set of the $RelevanceSet$ (cf. Fig. 1, step 7 and Alg. 1, line 9). Each of the 137 resulting strategies $s = \{m_1 \dots m_n\}$ represents a possible combination of motives or the empty set and is measured by a normalized local payout for each player based on weights of involved motives.

$$S_{p_a} = S_{p_b} = \{s_1 \dots s_{137}\}; s_{18} = \{m_Q, m_R\} \quad (7)$$

$$LocalPayout_{p_a,s_{18}} = 0.1280; LocalPayout_{p_b,s_{18}} = 0.0090$$

Strategy sets of players are identical regarding types of covered strategies, but differ in local payouts that can be expected by players when playing this strategy as shown in eq. (7). As players prefer those strategies that provide high local payouts, the equilibrium identifier identifies strategies $s^* \in S_p$ for each player that represent best answers regarding the behavior of counterparts $\overrightarrow{s_{\text{-}p}}$:

$$LocalPayout(s^*, \overrightarrow{s_{\text{-}p}}) \geq LocalPayout(s, \overrightarrow{s_{\text{-}p}}), \forall s \in S_p \quad (8)$$

Best answers of players in the sense of highest local payouts are aggregated to 17 strategy profiles, each a vector consisting of two strategies one for each player (cf. Alg. 1, line 10-15): $\overrightarrow{s} = \{s_x, s_y\}; s_x \in S_{p_a}, s_y \in S_{p_b}$.
Next, strategy profiles are selected that meet the Nash equilibrium condition, i.e. those strategy profiles exclusively cover strategies that represent mutual best answers of players (cf. Alg. 1, line 16-20):

$$LocalPayout(s^*, \overrightarrow{s_{\text{-}p}}^*) \geq LocalPayout(s, \overrightarrow{s_{\text{-}p}}^*) \quad (9)$$

$$\forall s \in \overrightarrow{s_1} \dots \overrightarrow{s_n}$$

In our example, we find two Nash equilibria. Those two strategy profiles represent best answers for the player p as well as the whole group of players P in the sense of a solution with minimum quality. No player has an incentive to deviate from those strategy profile because then its local payout would decrease. With $\overrightarrow{s} = \{s_{36}, s_{36}\}$, we select the non-pareto-dominant option for finding the strategy profile with the lowest difference in local payouts following the idea of the model to

create a balance between mixed motives. With each answer planning, players generate local payouts that are added during the course of dialogue to global payouts. Instead of gaining high global payouts, the objective of the model is to balance payouts of players during dialogue or to approximate them in case of drifting apart. We assume that similar global payouts of players can be regarded as evidence for satisficed mixed motives.
Based on the selected strategy profile, involved motives are aggregated to the $SatisficingSet = \{m_{\mathrm{R}}, m_{\mathrm{ICR}}\}$ that represents a combination of mixed motives that is satisficing for all players in this time in the dialogue (cf. Alg. 1, line 21).

3.2.4 Mapping mixed motives onto linguistic intentions

The resulting $SatisficingSet$ is forwarded to the mapper for mapping back motives onto linguistic intentions (cf. Fig. 1, step 8 & 9). In case, the $SatisficingSet$ covers zero motives, no mapping takes place, the process ends and none of the satellites in set S, cf. eq (4), will be considered in the actual answer planning. Otherwise, the mapper determines the set of supporting linguistic intentions by processing the inverse is-supported-by-relation between motives and linguistic intentions (cf. Alg. 1, line 22-30) (cf. Fig. 3). Comparing this set with the $SatisfactionSet$ (cf. eq. (5)), an intersection called $SupportSet$ is created that represents the set of linguistic intentions that will be satisfied in current answer planning:

$$SupportSet = \{li_{\mathrm{A}}, li_{\mathrm{MP}}\} \tag{10}$$

3.2.5 Determination of set S^+ and generation of answer

The linguistic intention handler determines the final set of satellites S^+ by analyzing 1:1 relations between linguistic intentions of the $SupportSet$ and satellites of the set S (cf. Fig. 1, step 10 and Alg. 1, line 27). The resulting set $S^+ = \{sat_{\mathrm{AAS}}, sat_{\mathrm{EUP}}\}$ consists of two satellites: *Alternatives_Advantages_Survey* (sat_{AAS}) and *Emotion_User_Preferences* (sat_{EUP}). The text plan lib handler adjusts the final text plan regarding the selected satellites before sending it to the answer generator (cf. Fig. 1, step 11 & 12). Last, the text plan provided by the plan operator NUMBER OF PRODUCTS is transformed into an answer. Thereby, answer schemata referenced by nucleus as well as satellites of set S^+ are instantiated (cf. Fig. 1, step 13):

Q: *"How many tablets offer the wifi features 802.11A, 802.11B, 802.11G, 802.11n?"*
A: *"*[$_{nuc}$ *The following tablets offer this feature: Sony SGPT122 Xperia.*] [$_{sat_{AAS}}$ *Due to its features, e.g., storage capacity: 32GB, Sony SGPT122 Xperia has some advantages compared to other products in this category.*] [$_{sat_{EUP}}$ *How about having a look at Sony SGPT122 Xperia by Sony?*]

3.2.6 Summary

In summary, satisficing answer planning is considered as a game consisting of four components $\langle P, S, F, A\rangle$: the set of players $P = \{p_{\mathrm{a}}, p_{\mathrm{b}}\}$, strategies of players $S = \{S_{\mathrm{p_a}}, S_{\mathrm{p_b}}\}$, objective functions of players $F = \{f_{\mathrm{p_a}}, f_{\mathrm{p_b}}\}$, and a state space $A = \{a^1 \dots a^{\mathrm{t}}\}$ that represents the rounds of the game, i.e. answers planned in the dialogue. The game starts in an initial state a^1. At a particular time t in the dialogue, the equilibrium identifier observes the state a^{t} characterized by P, S, and F and identifies best answers for all players; $s^{\mathrm{t}} \in S_{\mathrm{p}}; \forall p \in P$. Consequential, a strategy profile meeting the Nash equilibrium condition, $\overrightarrow{s}^{\mathrm{t}} = \{s^{\mathrm{t}}{}_{\mathrm{p_a}}, s^{\mathrm{t}}{}_{\mathrm{p_b}}\}$, is specified and resulting payouts are observed: $f(a^{\mathrm{t}}, \overrightarrow{s}^{\mathrm{t}}) \rightarrow LocalPayout \rightarrow \mathbb{R}$. The calculation of local payouts by means of objective functions $f \in F$ in state a^{t} does not depend solely on the selected strategy profile, but on results of former states in A, i.e. all answers planned in the dialogue until a^{t}. That means, infinite playing of the described non-zero-sum game $a^1, \overrightarrow{s}^1, \dots, a^{\mathrm{t}}, \overrightarrow{s}^{\mathrm{t}}, \dots$ generates a stream of payouts $f^1, f^2, \dots, f^{\mathrm{t}} = f(a^{\mathrm{t}}, \overrightarrow{s}^{\mathrm{t}})$. Besides relevant motives of the $RelevanceSet$, answer planning in state $a^{\mathrm{t+1}}$ is directly influenced by local payouts $f^{\mathrm{t}}(a^{\mathrm{t}}, \overrightarrow{s}^{\mathrm{t}})$ in a^{t} leading to a continuous deliberation of the mixed motive model during dialogue.

4 Implementation and Evaluation

Based on the proposed model (cf. Fig. 1), we implemented a German text-based QA system in form of a online shopping assistant (cf. Fig. 4)[1]. Users are able to construct questions term-by-term. Having tapped the last term of a question, the answer is given. The QA system uses the domain-specific knowledge representation mentioned in section 3.2 formalized in RDF[2].

[1] QA system was implemented as web application: http://redqueen.iss.uni-saarland.de/satin
[2] Resource Description Framework

Figure 4: Web-based QA system with posed question and given answer in German

4.1 Setting

To evaluate our approach, we conducted a user study with the implemented prototype in German that was set up as lab experiment. Goal of this study was to assess the perceived fairness and naturalness of the dialogue with the QA system as well as the extent of motive satisfaction of participants. For that purpose, four randomized groups were formed. Each group was characterized by a combination of motives by users (fair price of product (m_{FP}) or exclusive design of product (m_{ED})) and the QA system representing the retailer (increasing revenue (m_{IR}) or improving customer relationship (m_{ICR})) (cf. Tab. 3). These

		QA system	
	Motives	Increasing revenue (m_{IR})	Improving customer relationship (m_{ICR})
Users (N=107)	Fair price of product (m_{FP})	Mixed Motives Group #4	Mixed Motives Group #3
	Exclusive design of product (m_{ED})	Mixed Motives Group #2	Mixed Motives Group #1

Table 3: Groups and mixed motive combinations of user study

mixed motives were combined systematically by means of scenarios given to users and a manipulated mixed motive model of the QA system. Before interacting with the QA system that was embedded into a web-based questionnaire, participants had to opportunity to get to know the QA system and interacting with it for the first time (cf. Fig. 8). Participants were then asked to pose questions to the QA system and to evaluate generated answers against the background of their motive (e.g., m_{FP}) and the related scenario, e.g.:
"You are searching for a new tablet that shall be functional regarding standby and storage capacity. A fair price is important; no need for the latest innovation. You do not want to spend a lot of money for the new tablet. You are price conscious."
Participants were told to interact with the QA system as long as it needed to gain the information that was required by the scenario. Finally, seven-point Likert scales ranging from strongly disagree (1), neither (4) to strongly agree (7) were used to assess the perceived fairness of the dialogue, the naturalness of the dialogue and the motive satisfaction. Tab. 4 lists the questionnaire items for each of these constructs.

4.2 Results

In summary, 120 subjects participated in the experiment. A complete dataset from 107 participants (58,3% female) with an average age of 24.3 (SD=6.9) was considered for analysis. On average, interactions between participants (N=107) and the QA system covered 5.19 question answer pairs (cf. example dialogue in appendix A). 556 questions were posed by subjects; 35.07% of them were propositional questions (e.g., "Is product A up-to-date?"), 62.41% set questions (e.g., "Where is the difference between product A and product B?") and 2.52% choice questions (e.g., "Which product is better than product A?"), cf. Bunt et al. (2010).
Due to the fact that Cronbachs alpha values for all three multi-item constructs lie clearly above the recommended threshold of .70 (Nunnally, 1967),

Figure 5: Subject during interaction with QA system in user study

which indicates a good to excellent reliability of the scales, we calculated aggregated mean scores for each construct. The descriptive statistics of the three core constructs are presented in Tab. 4.

#	Construct	Item	Alpha	Aggregated scores (mean values of items)					Interpretation
				Mean	SD	one-sample t-test with test value 4			
						T	p	95% CI	
1	Perceived Naturalness of Dialogue	1. I perceived the sales dialogue with the shopping assistant as natural.	.847	4,05	1,19	0,41	0,686	-.18 - .27	Participants were undecided with respect to the naturalness of the dialogue with the QA system.
		2. I can imagine that this sales dialogue could have happened also in daily life.							
		3. I can remember a similar sales dialogue in my everyday life.							
		4. I can imagine to conduct a similar sales dialogue in my daily life.							
		5. I found that the sales dialogue with the shopping assistant was natural.							
		6. I could observe a similar sales dialogue in the past.							
2	Perceived Fairness of Dialogue	1. I found that the sales dialogue with the shopping assistant was pretty fair.	.947	5,17	1,20	10,10	0,000	.94 - 1.40	The dialogue was perceived as fair.
		2. The sales dialogue with the shopping assistant was particular equitably.							
		3.* I did not perceive the sales dialogue with the shopping assistant as being fair.							
		4.* To my understanding, the sales dialogue was not pretty equitably.							
3	Motive Satisfaction	1. I have achieved my motive in this sales dialogue.	.915	5,16	1,25	9,67	0,000	.93 - 1.40	Participants indicated that they were able to sufficiently satisfy their motives.
		2. I believe that I achieved my motive in this sales dialogue.							
		3. I was able to attain certain steps towards my motive in this sales dialogue.							
		4. * I was not able to attain my motive within the sales dialogue.							
* Reverse coded item									

Table 4: Descriptive statistics and results of one-sample t-tests for the empirical core constructs (N=107)

Additionally, results of one-sample t-tests are provided to evaluate whether the aggregated scores lie significantly above or below the neutral scale value of 4. Results indicate that the participants were undecided with respect to the "Perceived Naturalness of Dialogue" with the QA system. We assume that this is owed to the restricted QA setting since there were no significant differences among the four groups ($F(3,104) = 2.06$, $p = .11$) (cf. Tab. 3). However, the data support the conclusion that participants perceived the dialogue as fair and that they were able to sufficiently satisfy their motives. Assuming rather conflicting motives of subject and QA system as given for instance in group #4 in Tab. 3, it could be assumed that perceived fairness and motive satisfaction should be smaller than in rather congruent motive combinations as shown in group #1. Nonetheless, the mean value of the construct "Perceived Fairness of Dialogue" was 5.17 across all groups (significant above mean value 4) and there were no significant differences between the randomized groups ($F(3,104) = 1.59$, $p = .20$). Furthermore, "Motive Satisfaction" was rated with a mean value of 5.16 across all groups (significant above mean value 4) and again, there was not a significant effect of the group on motive satisfaction at the .05 level of significance ($F(3,104) = 2.33$, $p = .08$).
Overall, this indicates a positive evaluation of the QA system regarding its ability to generate satisficing answers despite of mixed motives of interlocutors.

5 Conclusion

We considered dialogues with congruent as well as incongruent interlocutor motives, where dialogue partners are facing the constant challenge of finding a balance between cooperation and competition. Despite of the overall presence of dialogues with such mixed motives in everyday life, little attention has been given to this topic in text planning in contrast to scrutinized dialogue systems that support dialogues fully aligned with anticipated user interests. Focusing on Question-Answering (QA) settings, we introduced a model that formalizes answer planning as psychological game embedded in text planning approaches for supporting fair dialogues under mixed motives. The model was exemplified within a QA sales assistant with domain-specific world knowledge for conducting sales dialogues. Due to the fact that fairness is subjective per se, we presented results from an empirical study (N=107) in which human subjects interacted with the QA system in various mixed motive settings. Results indicate a positive evaluation of the systems performance in planning answers that support fair dialogues despite of mixed motives of interlocutors.

Acknowledgments

This work was partially funded by the German Federal Ministry for Education and Research (BMBF) under the contract 01IS12030.

References

James F. Allen, Donna K Byron, Myroslava Dzikovska, George Ferguson, Lucian Galescu, and Amanda Stent. 2001. Toward conversational human-computer interaction. *AI magazine*, 22(4):27.

Adam Bjorndahl, Joseph Y Halpern, and Rafael Pass. 2013. Language-based games. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence*, pages 2967–2971.

Elizabeth Black and Katie Atkinson. 2011. Choosing persuasive arguments for action. In *10th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 905–912.

Michael Bratman. 1987. *Intention, Plans, and Practical Reason*. Center for the Study of Language and Information.

Harry Bunt and William Black. 2000. The abc of computational pragmatics. *Abduction, Belief and Context in Dialogue: Studies in Computational Pragmatics*, pages 1–46.

Harry Bunt, Jan Alexandersson, Jean Carletta, Jae-Woong Choe, Alex Chengyu Fang, Koiti Hasida, Kiyong Lee, Volha Petukhova, Andrei Popescu-Belis, Laurent Romary, et al. 2010. Towards an iso standard for dialogue act annotation. In *Seventh conference on International Language Resources and Evaluation (LREC'10)*.

Philip R. Cohen and Hector J. Levesque. 1990. Intention is choice with commitment. *Artif. Intell.*, 42(2-3):213–261.

Tiansi Dong, Ulrich Furbach, Ingo Glöckner, and Björn Pelzer. 2011. A natural language question answering system as a participant in human q&a portals. In *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence*, pages 2430–2435.

Michael P. Georgeff, Barney Pell, Martha E. Pollack, Milind Tambe, and Michael Wooldridge. 1998. The belief-desire-intention model of agency. In *Proc. of the 5th Int. Workshop on Intelligent Agents V, Agent Theories, Architectures, and Languages*, pages 1–10.

Barbara J. Grosz and Sarit Kraus. 1996. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269 – 357.

Barbara J. Grosz and Candace L. Sidner. 1986. Attention, intentions, and the structure of discourse. *Comput. Linguist.*, 12(3):175–204.

Christos Hadjinikolis, Yiannis Siantos, Sanjay Modgil, Elizabeth Black, and Peter McBurney. 2013. Opponent modelling in persuasion dialogues. In *Proc. of the 23rd Int. Joint Conf. on Artificial Intelligence*, pages 164–170.

Takayuki Hasegawa, Nobuhiro Kaji, Naoki Yoshinaga, and Masashi Toyoda. 2013. Predicting and eliciting addressee's emotion in online dialogue. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 964–972.

Jerry R. Hobbs. 1978. *Why is Discourse Coherent?: Technical Note 176*. Stanford Research Inst., Menlo Park.

Eduard H. Hovy, 1991. *Approaches to the planning of coherent text*, volume 119 of *The Kluwer International Series in Engineering and Computer Science*, pages 83–102. Springer.

Eduard H. Hovy, 1993. *Automated discourse generation using discourse structure relations*, pages 341–385. MIT Press.

Anthony Jameson, Bernhard Kipper, Alassane Ndiaye, Ralph Schäfer, Joep Simons, Thomas Weis, and Detlev Zimmermann. 1994. Cooperating to be non-cooperative: The dialog system pracma. In *Proc. of KI 1994*. Springer.

Willem JM Levelt. 1993. *Speaking: From intention to articulation*, volume 1. MIT press.

Fangtao Li, Yang Gao, Shuchang Zhou, Xiance Si, and Decheng Dai. 2013. Deceptive answer prediction with user preference graph. In *ACL (1)*, pages 1723–1732. Citeseer.

Karen E Lochbaum. 1998. A collaborative planning model of intentional structure. *Comput. Linguist.*, 24(4):525–572.

William C. Mann and Sandra A. Thompson. 1986. Assertions from discourse structure. In *Proc. of Workshop on Strategic comp. natural language*, pages 257–270.

Kathleen R. McKeown. 1985. Discourse strategies for generating natural-language text. *Artificial Intelligence*, 27(1):1–41.

Johanna D. Moore and Cécile L. Paris. 1993. Planning text for advisory dialogues: capturing intentional and rhetorical information. *Comput. Linguist.*, 19(4):651–694.

John Nash. 1951. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295.

Jum C. Nunnally. 1967. *Psychometric Theory*. McGraw-Hill, New York.

Oxford Dictionaries. 2016. "fair". Oxford University Press (http://oxforddictionaries.com).

Michel A. Paquette. 2012. Speech acts, dialogues and the common ground. In *FLAIRS Conference*.

Prashant Parikh. 2010. *Language and Equilibrium*. MIT Press.

Henry Prakken. 2006. Formal systems for persuasion dialogue. *The Knowledge Engineering Review*, 21(02):163–188.

Anand S. Rao and Michael P. Georgeff. 1995. Bdi agents: From theory to practice. In *Proc. of the 1st Intl. Conf. on Multiagent Systems (ICMAS)*, pages 312–319. The MIT Press.

Charles Rich and Candace L. Sidner. 1997. Collagen: when agents collaborate with people. In *Proceedings of the first international conference on Autonomous agents*, AGENTS '97, pages 284–291, New York, NY, USA. ACM.

Thomas C Schelling. 1960. *The strategy of conflict*. Harvard university press.

Herbert A. Simon. 1956. Rational choice and the structure of the environment. *Psychological review*, 63(2):129.

Jon Scott Stevens, Anton Benz, Sebastian Reue, and Ralf Klabunde. 2015. A strategic reasoning model for generating alternative answers. In *ACL (1)*, pages 534–542. The Association for Computer Linguistics.

David Traum and Staffan Larsson, 2003. *The information state approach to dialogue management*, pages 325–353. Springer.

David Traum, Stacy C Marsella, Jonathan Gratch, Jina Lee, and Arno Hartholt. 2008. Multi-party, multi-issue, multi-strategy negotiation for multi-modal virtual agents. In *Intelligent Virtual Agents*, pages 117–130. Springer.

Kees van Deemter. 2009. What game theory can do for nlg: the case of vague language. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 154–161.

A Example dialogue between user and QA system

In the following, an example dialogue between a user and the QA system consisting of four question answer pairs is given. Questions (Q) are segmentally constructed by the user. Answers (A) consisting of text, images, links to webpages and/or external product offers are generated by the QA system.

(1) Q: Which tablets are available in lower price range?

(2) A: In this price range, the following products are available: Lenovo Yoga. Compared to other products in this category, Lenovo Yoga offers advantages regarding its price of 198.99€. Currently, a lot of customers ask for Lenovo Yoga. Regarding the price-performance ratio, the product is absolutely recommended.

(3) Q: Is there a discount for Lenovo Yoga?

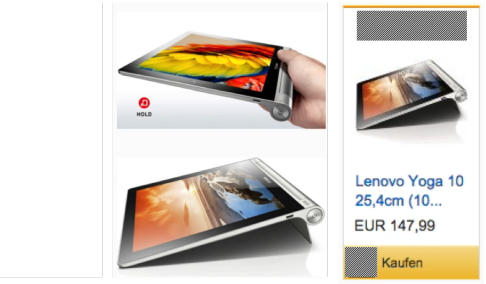
(4) A: For Lenovo Yoga, there is a discount of 1%. The total comes to 197.00€. Lenovo Yoga goes with Apple iPod Nano, Sony NWZ-E585B Walkman, Apple iPhone 5S and Apple iPhone 4S.

(5) Q: How many tablets offer a standby time of at least 18h?

(6) A: The following tablets offer this feature: Lenovo Yoga. How about having a look at Lenovo Yoga by Lenovo?

(7) Q: Are there any offers for Lenovo Yoga?

(8) A: You can order this product online, as you can see here:



A Sentence Interaction Network for Modeling Dependence between Sentences

Biao Liu[1], **Minlie Huang**[1]*, **Song Liu**[2], **Xuan Zhu**[2], **Xiaoyan Zhu**[1]
[1]State Key Lab. of Intelligent Technology and Systems
[1]National Lab. for Information Science and Technology
[1]Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
[2]Samsung R&D Institute, Beijing, China
[1]liubiao2638@gmail.com, {aihuang, zxy-dcs}@tsinghua.edu.cn

Abstract

Modeling interactions between two sentences is crucial for a number of natural language processing tasks including Answer Selection, Dialogue Act Analysis, etc. While deep learning methods like Recurrent Neural Network or Convolutional Neural Network have been proved to be powerful for sentence modeling, prior studies paid less attention on interactions between sentences. In this work, we propose a Sentence Interaction Network (SIN) for modeling the complex interactions between two sentences. By introducing "interaction states" for word and phrase pairs, SIN is powerful and flexible in capturing sentence interactions for different tasks. We obtain significant improvements on Answer Selection and Dialogue Act Analysis without any feature engineering.

1 Introduction

There exist complex interactions between sentences in many natural language processing (NLP) tasks such as Answer Selection (Yu et al., 2014; Yin et al., 2015), Dialogue Act Analysis (Kalchbrenner and Blunsom, 2013), etc. For instance, given a question and two candidate answers below, though they are all talking about cats, only the first answer correctly answers the question about cats' appearance. It is important to appropriately model the relation between two sentences in such cases.

Q	*What do cats look like?*
A1	*Cats have large eyes and furry bodies.*
A2	*Cats like to play with boxes and bags.*

* Correspondence author

For sentence pair modeling, some methods first project the two sentences to fix-sized vectors separately without considering the interactions between them, and then fed the sentence vectors to other classifiers as features for a specific task (Kalchbrenner and Blunsom, 2013; Tai et al., 2015). Such methods suffer from being unable to encode context information during sentence embedding.

A more reasonable way to capture sentence interactions is to introduce some mechanisms to utilize information from both sentences at the same time. Some methods attempt to introduce an attention matrix which contains similarity scores between words and phrases to approach sentence interactions (Socher et al., 2011; Yin et al., 2015). While the meaning of words and phrases may drift from contexts to contexts, simple similarity scores may be too weak to capture the complex interactions, and a more powerful interaction mechanism is needed.

In this work, we propose a Sentence Interaction Network (SIN) focusing on modeling sentence interactions. The main idea behind this model is that each word in one sentence may potentially influence every word in another sentence in some degree (the word "influence" here may refer to "answer" or "match" in different tasks). So, we introduce a mechanism that allows information to flow from every word (or phrase) in one sentence to every word (or phrase) in another sentence. These "information flows" are real-valued vectors describing how words and phrases interact with each other, for example, a word (or phrase) in one sentence can modify the meaning of a word (or phrase) in another sentence through such "information flows".

Specifically, given two sentences s_1 and s_2, for every word x_t in s_1, we introduce a "candidate interaction state" for every word x_τ in s_2. This

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 558–567,
Berlin, Germany, August 7-12, 2016.

state is regarded as the "influence" of x_τ to x_t, and is actually the "information flow" from x_τ to x_t mentioned above. By summing over all the "candidate interaction states", we generate an "interaction state" for x_t, which represents the influence of the whole sentence s_2 to word x_t . When feeding the "interaction state" and the word embedding together into Recurrent Neural Network (with Long Short-Time Memory unit in our model), we obtain a sentence vector with context information encoded. We also add a convolution layer on the word embeddings so that interactions between phrases can also be modeled.

SIN is powerful and flexible for modeling sentence interactions in different tasks. First, the "interaction state" is a vector, compared with a single similarity score, it is able to encode more information for word or phrase interactions. Second, the interaction mechanism in SIN can be adapted to different functions for different tasks during training, such as "word meaning adjustment" for Dialogue Act Analysis or "Answering" for Answer Selection.

Our main contributions are as follows:

- We propose a Sentence Interaction Network (SIN) which utilizes a new mechanism to model sentence interactions.
- We add convolution layers to SIN, which improves the ability to model interactions between phrases.
- We obtain significant improvements on Answer Selection and Dialogue Act Analysis without any handcrafted features.

The rest of the paper is structured as follows: We survey related work in Section 2, introduce our method in Section 3, present the experiments in Section 4, and summarize our work in Section 5.

2 Related Work

Our work is mainly related to deep learning for sentence modeling and sentence pair modeling.

For sentence modeling, we have to first represent each word as a real-valued vector (Mikolov et al., 2010; Pennington et al., 2014) , and then compose word vectors into a sentence vector. Several methods have been proposed for sentence modeling. Recurrent Neural Network (RNN) (Elman, 1990; Mikolov et al., 2010) introduces a hidden state to represent contexts, and repeatedly feed the hidden state and word embeddings to the network to update the context representation. RNN suffers from gradient vanishing and exploding problems which limit the length of reachable context. RNN with Long Short-Time Memory Network unit (LSTM) (Hochreiter and Schmidhuber, 1997; Gers, 2001) solves such problems by introducing a "memory cell" and "gates" into the network. Recursive Neural Network (Socher et al., 2013; Qian et al., 2015) and LSTM over tree structures (Zhu et al., 2015; Tai et al., 2015) are able to utilize some syntactic information for sentence modeling. Kim (2014) proposed a Convolutional Neural Network (CNN) for sentence classification which models a sentence in multiple granularities.

For sentence pair modeling, a simple idea is to first project the sentences to two sentence vectors separately with sentence modeling methods, and then feed these two vectors into other classifiers for classification (Tai et al., 2015; Yu et al., 2014; Yang et al., 2015). The drawback of such methods is that separately modeling the two sentences is unable to capture the complex sentence interactions. Socher et al. (2011) model the two sentences with Recursive Neural Networks (Unfolding Recursive Autoencoders), and then feed similarity scores between words and phrases (syntax tree nodes) to a CNN with dynamic pooling to capture sentence interactions. Hu et al. (2014) first create an "interaction space" (matching score matrix) by feeding word and phrase pairs into a multilayer perceptron (MLP), and then apply CNN to such a space for interaction modeling. Yin et al. (2015) proposed an Attention based Convolutional Neural Network (ABCNN) for sentence pair modeling. ABCNN introduces an attention matrix between the convolution layers of the two sentences, and feed the matrix back to CNN to model sentence interactions. There are also some methods that make use of rich lexical semantic features for sentence pair modeling (Yih et al., 2013; Yang et al., 2015), but these methods can not be easily adapted to different tasks.

Our work is also related to context modeling. Hermann et al. (2015) proposed a LSTM-based method for reading comprehension. Their model is able to effectively utilize the context (given by a document) to answer questions. Ghosh et al. (2016) proposed a Contextual LSTM (CLSTM) which introduces a topic vector into LSTM for context modeling. The topic vector in CLSTM is

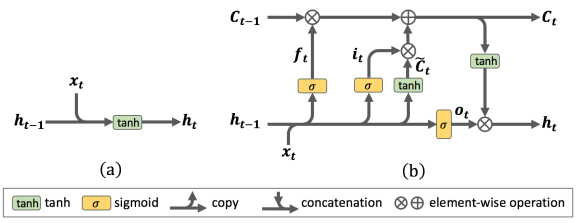


Figure 1: RNN (a) and LSTM (b) [1]

computed according to those already seen words, and therefore reflects the underlying topic of the current word.

3 Method

3.1 Background: RNN and LSTM

Recurrent Neural Network (RNN) (Elman, 1990; Mikolov et al., 2010), as depicted in Figure 1(a), is proposed for modeling long-distance dependence in a sequence. Its hidden layer is connected to itself so that previous information is considered in later times. RNN can be formalized as

$$h_t = f(W_x x_t + W_h h_{t-1} + b_h)$$

where x_t is the input at time step t and h_t is the hidden state. Though theoretically, RNN is able to capture dependence of arbitrary length, it tends to suffer from the gradient vanishing and exploding problems which limit the length of reachable context. In addition, an additive function of the previous hidden layer and the current input is too simple to describe the complex interactions within a sequence.

RNN with Long Short-Time Memory Network unit (LSTM, Figure 1(b)) (Hochreiter and Schmidhuber, 1997; Gers, 2001) solves such problems by introducing a "memory cell" and "gates" into the network. Each time step is associated with a subnet known as a memory block in which a "memory cell" stores the context information and "gates" control which information should be added or discarded or reserved. LSTM can be formalized as

$$\begin{aligned} f_t &= \sigma(W_f \cdot [x_t, h_{t-1}] + b_f) \\ i_t &= \sigma(W_i \cdot [x_t, h_{t-1}] + b_i) \\ \tilde{C}_t &= \tanh(W_C \cdot [x_t, h_{t-1}] + b_C) \\ C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\ o_t &= \sigma(W_o \cdot [x_t, h_{t-1}] + b_o) \\ h_t &= o_t * \tanh(C_t) \end{aligned}$$

where $*$ means element-wise multiplication, f_t, i_t, o_t is the forget, input and output gate that control which information should be forgot, input and output, respectively. $\tilde{C}_t$ is the candidate information to be added to the memory cell state C_t. h_t is the hidden state which is regarded as a representation of the current time step with contexts.

In this work, we use LSTM with peephole connections, namely adding C_{t-1} to compute the forget gate f_t and the input gate i_t, and adding C_t to compute the output gate o_t.

3.2 Sentence Interaction Network (SIN)

Sentence Interaction Network (SIN, Figure 2) models the interactions between two sentences in two steps.

First, we use a LSTM (referred to as LSTM_1) to model the two sentences s_1 and s_2 separately, and the hidden states related to the t-th word in s_1 and the τ-th word in s_2 are denoted as $z_t^{(1)}$ and $z_\tau^{(2)}$ respectively. For simplicity, we will use the position (t, τ) to denote the corresponding words hereafter.

Second, we propose a new mechanism to model the interactions between s_1 and s_2 by allowing information to flow between them. Specifically, word t in s_1 may be potentially influenced by all words in s_2 in some degree. Thus, for word t in s_1, a candidate interaction state $\tilde{c}_{t\tau}^{(i)}$ and an input gate $i_{t\tau}^{(i)}$ are introduced for each word τ in s_2 as follows:

$$\begin{aligned} \tilde{c}_{t\tau}^{(i)} &= \tanh(W_c^{(i)} \cdot [z_t^{(1)}, z_\tau^{(2)}] + b_c^{(i)}) \\ i_{t\tau}^{(i)} &= \sigma(W_i^{(i)} \cdot [z_t^{(1)}, z_\tau^{(2)}] + b_i^{(i)}) \end{aligned}$$

here, the superscript "i" indicates "interaction". $W_c^{(i)}, W_i^{(i)}, b_c^{(i)}, b_i^{(i)}$ are model parameters. The interaction state $c_t^{(i)}$ for word t in s_1 can then be formalized as

$$c_t^{(i)} = \sum_{\tau=1}^{|s_2|} \tilde{c}_{t\tau}^{(i)} * i_{t\tau}^{(i)}$$

where $|s_2|$ is the length of sentence s_2, and $c_t^{(i)}$ can be viewed as the total interaction information received by word t in s_1 from sentence s_2. The interaction states of words in s_2 can be similarly

[1] This figure referred to http://colah.github.io/posts/2015-08-Understanding-LSTMs/

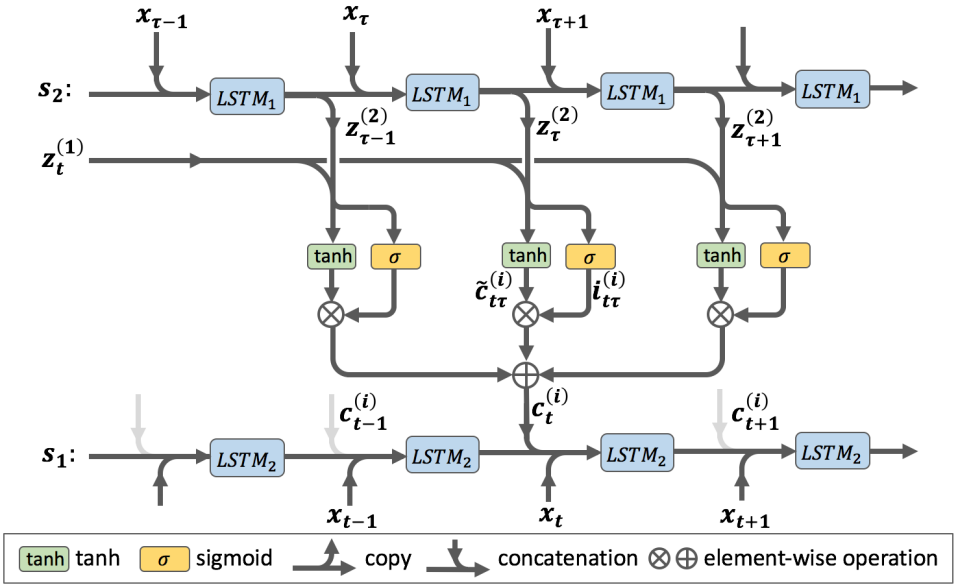


Figure 2: SIN for modeling sentence s_1 at timestep t. First, we model s_1 and s_2 separately with LSTM$_1$ and obtain the hidden states $z_t^{(1)}$ for s_1 and $z_\tau^{(2)}$ for s_2. Second, we compute interaction states based on these hidden states, and incorporate $c_t^{(i)}$ into LSTM$_2$. Information flows (interaction states) from s_1 to s_2 are not depicted here for simplicity.

computed by exchanging the position of $z_t^{(1)}$ and $z_\tau^{(2)}$ in $\tilde{c}_{t\tau}^{(i)}$ and $i_{t\tau}^{(i)}$ while sharing the model parameters.

We now introduce the interaction states into another LSTM (referred to as LSTM$_2$) to compute the sentence vectors. Therefore, information can flow between the two sentences through these states. For sentence s_1, at timestep t, we have

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [x_t, h_{t-1}, c_t^{(i)}, C_{t-1}] + b_f) \\
i_t &= \sigma(W_i \cdot [x_t, h_{t-1}, c_t^{(i)}, C_{t-1}] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [x_t, h_{t-1}, c_t^{(i)}] + b_C) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [x_t, h_{t-1}, c_t^{(i)}, C_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned}
$$

By averaging all hidden states of LSTM$_2$, we obtain the sentence vector v_{s_1} of s_1, and the sentence vector v_{s_2} of s_2 can be computed similarly. v_{s_1} and v_{s_2} can then be used as features for different tasks.

In SIN, the candidate interaction state $\tilde{c}_{t\tau}^{(i)}$ represents the potential influence of word τ in s_2 to word t in s_1, and the related input gate $i_{t\tau}^{(i)}$ controls the degree of the influence. The element-wise multiplication $\tilde{c}_{t\tau}^{(i)} * i_{t\tau}^{(i)}$ is then the actual influence. By summing over all words in s_2, the interaction state $c_t^{(i)}$ gives the influence of the whole sentence s_2 to word t.

3.3 SIN with Convolution (SIN-CONV)

SIN is good at capturing the complex interactions of words in two sentences, but not strong enough for phrase interactions. Since convolutional neural network is widely and successfully used for modeling phrases, we add a convolution layer before SIN to model phrase interactions between two sentences.

Let $v_1, v_2, ..., v_{|s|}$ be the word embeddings of a sentence s, and let $c_i \in \mathbb{R}^{wd}, 1 \leq i \leq |s| - w + 1$, be the concatenation of $v_{i:i+w-1}$, where w is the window size. The representation p_i for phrase $v_{i:i+w-1}$ is computed as:

$$p_i = \tanh(F \cdot c_i + b)$$

where $F \in \mathbb{R}^{d \times wd}$ is the convolution filter, and d is the dimension of the word embeddings.

In SIN-CONV, we first use a convolution layer to obtain phrase representations for the two sentences s_1 and s_2, and the SIN interaction procedure is then applied to these phrase representations as before to model phrase interactions. The average of all hidden states are treated as sentence vectors $v_{s_1}^{cnn}$ and $v_{s_2}^{cnn}$. Thus, SIN-CONV is SIN with word vectors substituted by phrase vectors. The

two phrase-based sentence vectors are then fed to a classifier along with the two word-based sentence vectors together for classification.

The LSTM and interaction parameters are not shared between SIN and SIN-CONV.

4 Experiments

In this section, we test our model on two tasks: Answer Selection and Dialogue Act Analysis. Both tasks require to model interactions between sentences. We also conduct auxiliary experiments for analyzing the interaction mechanism in our SIN model.

4.1 Answer Selection

Selecting correct answers from a set of candidates for a given question is quite crucial for a number of NLP tasks including question-answering, natural language generation, information retrieval, etc. The key challenge for answer selection is to appropriately model the complex interactions between the question and the answer, and hence our SIN model is suitable for this task.

We treat Answer Selection as a classification task, namely to classify each question-answer pair as "correct" or "incorrect". Given a question-answer pair (q, a), after generating the question and answer vectors v_q and v_a using SIN, we feed them to a logistic regression layer to output a probability. And we maximize the following objective function:

$$p_\theta(q, a) = \sigma(W \cdot [v_q, v_a]) + b)$$
$$\mathcal{L} = \sum_{(q,a)} \hat{y}_{q,a} \log p_\theta(q, a) + (1 - \hat{y}_{q,a}) \log(1 - p_\theta(q, a))$$

where $\hat{y}_{q,a}$ is the true label for the question-answer pair (q, a) (1 for correct, 0 for incorrect). For SIN-CONV, the sentence vector v_q^{cnn} and v_a^{cnn} are also fed to the logistic regression layer.

During evaluation, we rank the answers of a question q according to the probability $p_\theta(q, a)$. The evaluation metrics are mean average precision (MAP) and mean reciprocal rank (MRR).

4.1.1 Dataset

The WikiQA[2](Yang et al., 2015) dataset is used for this task. Following Yin et al. (2015), we filtered out those questions that do not have any correct answers from the development and test set. Some statistics are shown in Table 1.

	Q	QA pair	A/Q	correct A/Q
Train	2,118	20,360	9.61	0.49
Dev	126	1,130	8.97	1.11
Test	243	2,351	9.67	1.21

Table 1: Statistics of WikiQA (Q=Question, A=Answer)

4.1.2 Setup

We use the 100-dimensional GloVe vectors[3] (Pennington et al., 2014) to initialize our word embeddings, and those words that do not appear in Glove vectors are treated as unknown. The dimension of all hidden states is set to 100 as well. The window size of the convolution layer is 2. To avoid overfitting, dropout is introduced to the sentence vectors, namely setting some dimensions of the sentence vectors to 0 with a probability p (0.5 in our experiment) randomly. No handcrafted features are used in our methods and the baselines.

Mini-batch Gradient Descent (30 question-answer pairs for each mini batch), with AdaDelta tuning learning rate, is used for model training. We update model parameters after every mini batch, check validation MAP and save model after every 10 batches. We run 10 epochs in total, and the model with highest validation MAP is treated as the optimal model, and we report the corresponding test MAP and MRR metrics.

4.1.3 Baselines

We compare our SIN and SIN-CONV model with 5 baselines listed below:

- LCLR: The model utilizes rich semantic and lexical features (Yih et al., 2013).
- PV: The cosine similarity score of paragraph vectors of the two sentences is used to rank answers (Le and Mikolov, 2014).
- CNN: Bigram CNN (Yu et al., 2014).
- ABCNN: Attention based CNN, no handcrafted features are used here (Yin et al., 2015).
- LSTM: The question and answer are modeled by a simple LSTM. Different from SIN, there is no interaction between sentences.

[2]http://aka.ms/WikiQA

[3]http://nlp.stanford.edu/projects/glove/

4.1.4 Results

Results are shown in Table 2. SIN performs much better than LSTM, PV and CNN, this justifies that the proposed interaction mechanism well captures the complex interactions between the question and the answer. But SIN performs slightly worse than ABCNN because it is not strong enough at modeling phrases. By introducing a simple convolution layer to improve its phrase-modeling ability, SIN-CONV outperforms all the other models.

For SIN-CONV, we do not observe much improvements by using larger convolution filters (window size ≥ 3) or stacking more convolution layers. The reason may be the fact that interactions between long phrases is relatively rare, and in addition, the QA pairs in the WikiQA dataset may be insufficient for training such a complex model with long convolution windows.

4.2 Dialogue Act Analysis

Dialogue acts (DA), such as *Statement, Yes-No-Question, Agreement*, indicate the sentence pragmatic role as well as the intention of the speakers (Williams, 2012). They are widely used in natural language generation (Wen et al., 2015), speech and meeting summarization (Murray et al., 2006; Murray et al., 2010), etc. In a dialogue, the DA of a sentence is highly relevant to the content of itself and the previous sentences. As a result, to model the interactions and long-range dependence between sentences in a dialogue is crucial for dialogue act analysis.

Given a dialogue (n sentences) $d = [s_1, s_2, ..., s_n]$, we first use a LSTM (LSTM_1) to model all the sentences independently. The hidden states of sentence s_i obtained at this step are used to compute the interaction states of sentence s_{i+1}, and SIN will generate a sentence vector v_{s_i} using another LSTM (LSTM_2) for each sentence s_i in the dialogue (see Section 3.2) . These sentence vectors can be used as features for dialogue act analysis. We refer to this method as SIN (or SIN-CONV for adding a convolution layer).

For dialogue act analysis, we add a softmax layer on the sentence vector v_{s_i} to predict the probability distribution:

$$p_\theta(y_j|v_{s_i}) = \frac{\exp(v_{s_i}^T \cdot w_j + b_j)}{\sum_k \exp(v_{s_i}^T \cdot w_k + b_k)}$$

[4]With extra handcrafted features, ABCNN's performance is: MAP(0.692), MRR(0.711).

Model	MAP	MRR
LCLR	0.599	0.609
PV	0.511	0.516
CNN	0.619	0.628
ABCNN	0.660	0.677
LSTM	0.634	0.648
SIN	0.657	0.672
SIN-CONV	**0.674**	**0.693**

Table 2: Results on answer selection[4].

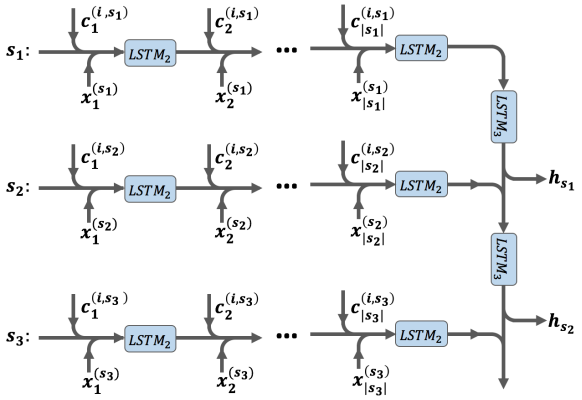


Figure 3: SIN-LD for dialogue act analysis. LSTM_1 is not shown here for simplicity. $x_t^{(s_j)}$ means word t in s_j, $c_t^{(i,s_j)}$ means the interaction state for word t in s_j.

where y_j is the j-th DA tag, w_j and b_j is the weight vector and bias corresponding to y_j. We maximize the following objective function:

$$\mathcal{L} = \sum_{d \in \mathcal{D}} \sum_{i=1}^{|d|} \log p_\theta(\hat{y}_{s_i}|v_{s_i})$$

where $\mathcal{D}$ is the training set, namely a set of dialogues, $|d|$ is the length of the dialogue, s_i is the i-th sentence in d, $\hat{y}_{s_i}$ is the true dialogue act label of s_i.

In order to capture long-range dependence in the dialogue, we can further join up the sentence vector v_{s_i} with another LSTM (LSTM_3). The hidden state h_{s_i} of LSTM_3 are treated as the final sentence vector, and the probability distribution is given by substituting v_{s_i} with h_{s_i} in $p_\theta(y_j|v_{s_i})$. We refer to this method as SIN-LD (or SIN-CONV-LD for adding a convolution layer), where LD means long-range dependence. Figure 3 shows the whole structure (LSTM_1 is not shown here for simplicity).

Dialogue Act	Example	Train(%)	Test(%)
Statement-non-Opinion	*Me, I'm in the legal department.*	37.0	31.5
Backchannel/Acknowledge	*Uh-huh.*	18.8	18.3
Statement-Opinion	*I think it's great*	12.8	17.2
Abandoned/Uninterpretable	*So,-*	7.6	8.6
Agreement/Accept	*That's exactly it.*	5.5	5.0
Appreciation	*I can imagine.*	2.4	1.8
Yes-No-Question	*Do you have to have any special training?*	2.3	2.0
Non-Verbal	*[Laughter], [Throat-clearing]*	1.8	2.3
Yes-Answers	*Yes.*	1.5	1.7
Conventional-closing	*Well, it's been nice talking to you.*	1.3	1.9
Other Labels(32)		9.1	9.8
Total number of sentences		196258	4186
Total number of dialogues		1115	19

Table 3: Dialogue act labels

4.2.1 Dataset

We use the Switch-board Dialogue Act (SwDA) corpus (Calhoun et al., 2010) in our experiments[5]. SwDA contains the transcripts of several people discussing a given topic on the telephone. There are 42 dialogue act tags in SwDA,[6] and we list the 10 most frequent tags in Table 3.

The same data split as in Stolcke et al. (2000) is used in our experiments. There are 1,115 dialogues in the training set and 19 dialogues in the test set[7]. We also randomly split the original training set as a new training set (1,085 dialogues) and a validation set (30 dialogues).

4.2.2 Setup

The setup is the same as that in Answer Selection except: (1) Only the most common 10,000 words are used, other words are all treated as unknown. (2) Each mini batch contains all sentences from 3 dialogues for Mini-batch Gradient Descent. (3) The evaluation metric is accuracy. (4) We run 30 epochs in total. (5) We use the last hidden state of LSTM_2 as sentence representation since the sentences here are much shorter compared with those in Answer Selection.

4.2.3 Baselines

We compare with the following baselines:

- unigram, bigram, trigram LM-HMM: HMM variants (Stolcke et al., 2000).
- RCNN: Recurrent Convolutional Neural Networks (Kalchbrenner and Blunsom, 2013). Sentences are first separately embedded with CNN, and then joined up with RNN.
- LSTM: All sentences are modeled separately by one LSTM. Different from SIN, there is no sentence interactions in this method.

Model	Accuracy(%)
unigram LM-HMM	68.2
bigram LM-HMM	70.6
trigram LM-HMM	71.0
RCNN	73.9
LSTM	72.8
SIN	74.8
SIN-CONV	75.1
SIN-LD	76.0
SIN-CONV-LD	**76.5**

Table 4: Accuracy on dialogue act analysis. Inter-annotator agreement is 84%.

4.2.4 Results

Results are shown in Table 4. HMM variants, RCNN and LSTM model the sentences separately during sentence embedding, and are unable to capture the sentence interactions. With our interaction mechanism, SIN outperforms LSTM, and proves that well modeling the interactions between sentences in a dialogue is important for dialogue act analysis. After introducing a convolution layer, SIN-CONV performs slightly better than SIN. SIN-LD and SIN-CONV-LD model the

[5] http://compprag.christopherpotts.net /swda.html.

[6] SwDA actually contains 43 tags in which "+" should not be treated as a valid tag since it means continuation of the previous sentence.

[7] http://web.stanford.edu/%7ejurafsky/ws97/

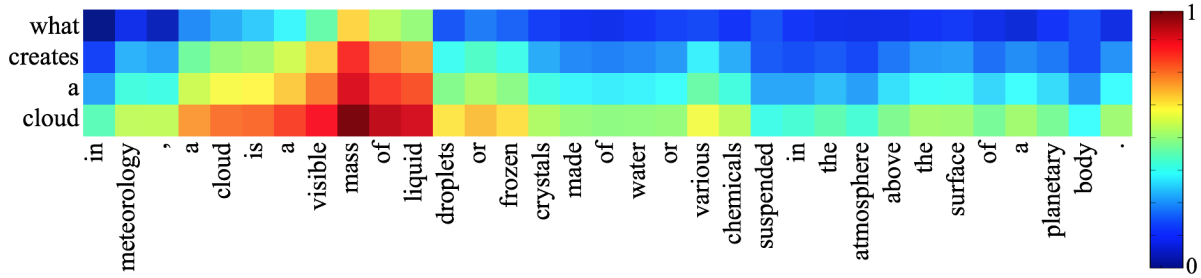


Figure 4: L_2-norm of the interaction states from question to answer (linearly mapped to $[0, 1]$).

Q:	*what creates a cloud*
A:	*in meteorology , a cloud is a visible mass of liquid droplets or frozen crystals made of water or various chemicals suspended in the atmosphere above the surface of a planetary body.*

Table 5: A question-answer pair example.

long-range dependence in the dialogue with another LSTM, and obtain further improvements.

4.3 Interaction Mechanism Analysis

We investigate into the interaction states of SIN for Answer Selection to see how our proposed interaction mechanism works.

Given a question-answer pair in Table 5, for SIN, there is a candidate interaction state $\tilde{c}_{\tau t}^{(i)}$ and an input gate $i_{\tau t}^{(i)}$ from each word t in the question to each word τ in the answer. We investigate into the L_2-norm $||\tilde{c}_{\tau t}^{(i)} * i_{\tau t}^{(i)}||_2$ to see how words in the two sentences interact with each other. Note that we have linearly mapped the original L_2-norm value to $[0, 1]$ as follows:

$$f(x) = \frac{x - x_{min}}{x_{max} - x_{min}}$$

As depicted in Figure 4, we can see that the word *"what"* in the question has little impact to the answer through interactions. This is reasonable since *"what"* appears frequently in questions, and does not carry much information for answer selection[8]. On the contrary, the phrase *"creates a cloud"*, especially the word *"cloud"*, transmits much information through interactions to the answer, this conforms with human knowledge since we rely on these words to answer the question as well.

In the answer, interactions concentrate on the phrase *"a cloud is a visible mass of liquid droplets"* which seems to be a good and complete answer to the question. Although there are also other highly related words in the answer, they are almost ignored. The reason may be failing to model such a complex phrase (three relatively simple sentences joined by "or") or the existence of the previous phrase which is already a good answer.

This experiment clearly shows how the interaction mechanism works in SIN. Through interaction states, SIN is able to figure out what the question is asking about, namely to detect those highly informative words in the question, and which part in the answer can answer the question.

5 Conclusion and Future Work

In this work, we propose Sentence Interaction Network (SIN) which utilizes a new mechanism for modeling interactions between two sentences. We also introduce a convolution layer into SIN (SIN-CONV) to improve its phrase modeling ability so that phrase interactions can be handled. SIN is powerful and flexible to model sentence interactions for different tasks. Experiments show that the proposed interaction mechanism is effective, and we obtain significant improvements on Answer Selection and Dialogue Act Analysis without any handcrafted features.

Previous works have showed that it is important to utilize the syntactic structures for modeling sentences. We also find out that LSTM is sometimes unable to model complex phrases. So, we are going to extend SIN to tree-based SIN for sentence modeling as future work. Moreover, applying the models to other tasks, such as semantic relatedness measurement and paraphrase identification, would

[8]Our statements focus on the interaction, in a sense of *"answering" or "matching"*. Definitely, such words like "what" and "why" are very important for answering questions from the general QA perspective since they determine the type of answers.

also be interesting attempts.

6 Acknowledgments

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2012CB316301/2013CB329403, the National Science Foundation of China under grant No. 61272227/61332007, and the Beijing Higher Education Young Elite Teacher Project. The work was also supported by Tsinghua University – Beijing Samsung Telecom R&D Center Joint Laboratory for Intelligent Media Computing.

References

Sasha Calhoun, Jean Carletta, Jason M Brenier, Neil Mayo, Dan Jurafsky, Mark Steedman, and David Beaver. 2010. The nxt-format switchboard corpus: a rich resource for investigating the syntax, semantics, pragmatics and prosody of dialogue. *Language resources and evaluation*, 44(4):387–419.

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

Felix Gers. 2001. Long short-term memory in recurrent neural networks. *Unpublished PhD dissertation, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland*.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *arXiv preprint arXiv:1306.3584*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, pages 1045–1048.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 367–374. Association for Computational Linguistics.

Gabriel Murray, Giuseppe Carenini, and Raymond Ng. 2010. Generating and validating abstracts of meeting conversations: a user study. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 105–113. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Qiao Qian, Bo Tian, Minlie Huang, Yang Liu, Xuan Zhu, and Xiaoyan Zhu. 2015. Learning tag embeddings and tag-specific composition functions in recursive neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 1365–1374.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. 2000. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373.

Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Jason D Williams. 2012. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on Future Directions and Needs in the Spoken Dialog Community: Tools and Data*, pages 23–24. Association for Computational Linguistics.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Citeseer.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Xiaodan Zhu, Parinaz Sobhani, and Hongyu Guo. 2015. Long short-term memory over tree structures. *arXiv preprint arXiv:1503.04881*.

Towards more variation in text generation: Developing and evaluating variation models for choice of referential form

Thiago Castro Ferreira and **Emiel Krahmer** and **Sander Wubben**
Tilburg center for Cognition and Communication (TiCC)
Tilburg University
The Netherlands
{tcastrof,e.j.krahmer,s.wubben}@tilburguniversity.edu

Abstract

In this study, we introduce a non-deterministic method for referring expression generation. We describe two models that account for individual variation in the choice of referential form in automatically generated text: a Naive Bayes model and a Recurrent Neural Network. Both are evaluated using the VaREG corpus. Then we select the best performing model to generate referential forms in texts from the GREC-2.0 corpus and conduct an evaluation experiment in which humans judge the coherence and comprehensibility of the generated texts, comparing them both with the original references and those produced by a random baseline model.

1 Introduction

Automatic text generation is the process of converting non-linguistic data into coherent and comprehensible text (Reiter and Dale, 2000). In recent years, interest in text generation has substantially increased, due to the emergence of new applications such as "robot-journalism" (Clerwall, 2014). Even though computers these days are perfectly capable of automatically producing text, the results are arguably often rather rigid, always producing the same kind and style of text, which makes them somewhat "boring" to read, especially when reading multiple texts in succession.

Human-written texts, by contrast, do not suffer from this problem, presumably because human authors have an innate tendency to produce variation in their use of words and constructions. Indeed, psycholinguistic research has shown that when speakers produce referring expressions in comparable contexts, they non-deterministically vary both the form and the contents of their references (Dale and Viethen, 2010; Van Deemter et al., 2012). In this paper, we present and evaluate models of referring expression generation that mimic this human non-determinacy and show that this enables us to generate varied references in texts, which, in terms of coherence and comprehensibility, did not yield significant differences from human-produced references according to human judges.

In particular, in this study we focus on the choice of referential *form*, which is the first decision to be made by referring expression generation models (Reiter and Dale, 2000) and which determines whether a reference takes the form of a proper name, a pronoun, a definite description, etc. Several such models have been proposed (Reiter and Dale, 2000; Henschel et al., 2000; Callaway and Lester, 2002; Krahmer and Theune, 2002; Gupta and Bandopadhyay, 2009; Greenbacker and McCoy, 2009). However, all of these are fully deterministic, always choosing the same referential form in the same context.

The fact that these models are generally based on text corpora which have only one gold standard form per reference (the one produced by the original author) does not help either. When the corpus contains, say, a description at some point in the text, this does not mean that, for example, a proper name could not occur in that position as well (Yeh and Mellish, 1997; Ferreira et al., 2016). Generally, we just don't know. To counter this problem, a recent corpus, called VaREG, was developed in which 20 different writers were asked to produce references for a particular topic in a variety of texts, giving rise to a distribution over forms per reference (Ferreira et al., 2016). This gives us the possibility to distinguish situations where there is more or less agreement between writers in their choices of referential form. But it also enables a new paradigm for choosing referential forms,

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 568–577,
Berlin, Germany, August 7-12, 2016.

where instead of predicting the most likely referential form, we can in fact predict the frequency in which a reference assumes a specific form, allowing us to turn the choice of referential form into a non-deterministic probabilistic model.

In this study, we introduce two different models that take the individual variation into account for the choice of referential form, one based on Naive Bayes and one on Recurrent Neural Networks. Both are evaluated using the VaREG corpus. Furthermore, we use the best performing model to generate referential forms in texts from the GREC-2.0 corpus, based on the roulette-wheel generation process (Belz, 2008), and conduct an evaluation experiment in which humans judge the coherence and comprehensibility of the generated texts, comparing them both with the original references and those produced by a random baseline model.

2 Related Studies

Several models for the choice of referential form have been proposed in the literature. They can roughly be distinguished in two groups: rule-based and data-driven models.

Many rule-based models were created for pronominalization, i.e, to choose whether an object or person should be referred to using a pronoun or not. Reiter and Dale (2000) proposed one of the first rule-based models, which opts for a pronominal reference only if the referent was previously mentioned in the discourse and no mention to an entity of same gender can be found between the reference and its antecedent. Henschel et al. (2000) presented a pronominalization model based on recency, discourse status, syntactic position, parallelism and ambiguity. To decide among a pronoun or a definite description, Callaway and Lester (2002) also proposed a rule-based model which makes the choices based on information about the discourse, rhetorical structure, recency and distance. Krahmer and Theune (2002) extended the Incremental algorithm so that if a referent achieves a level of salience in the discourse (measured by a salience weight), a pronoun is used. Otherwise, a definite description is produced to distinguish the referent from the distractors.

Aiming to make choices similar to humans, some studies proposed machine learning models trained on human choices of referential form. The GREC project (Belz et al., 2010) motivated the development of many of those data-driven models. One of the project's shared tasks aimed to predict the form of the references to the main topics of texts taken from Wikipedia. Among the participants of the task, Gupta and Bandopadhyay (2009) presented a model that combined rules and a machine learning technique based on semantic and syntactic category, paragraph and sentence positions, and reference number. Similarly, Greenbacker and McCoy (2009) proposed a decision tree that, besides the features used in Gupta and Bandopadhyay (2009), was also based on recency and part-of-speech features. For more information on the GREC shared task, see Belz et al. (2010).

One limitation that these models all have in common is that they fail to model individual variation. According to their predictions, a reference will always assume the most likely referential form. For example, a model that takes into account syntactic position will always choose the same referential form for the subject of a sentence, while humans tend to vary in their choices of referential form. One of the reasons for this problem arises from the data these models are trained on. Most corpora only contain one referring expression per reference. Only the newly introduced VaREG corpus takes variation into account, containing 20 different expressions for each reference, allowing us to model distributions over referential slots.

3 The VaREG corpus

The VaREG corpus was collected for the study of individual variation in the choice of referential form (Ferreira et al., 2016). The corpus is based on a number of texts, which were presented to participants in such a way that all references to the main topic of the text had been replaced with gaps. Each participant was asked to fill each of those gaps with a referring expression for the topic.

The resulting corpus consists of 9,588 referring expressions, produced by 78 participants for 563 referential gaps - around 20 referring expressions per reference - in 36 English texts. The texts were equally distributed over 3 genres: news texts, reviews of commercial products and encyclopedic texts. The references were annotated according to their syntactic position (subject, object, etc.), referential status (new or old, in text, paragraph and sentence) and recency (number of words between previous reference to the same object or entity),

and the referring expressions of the participants were classified into 5 referential forms: proper names, pronouns, definite descriptions, demonstratives and empty references.

The analysis of the corpus revealed considerable variation among participants in their choices of referential forms. Various factors influenced the amount of variation that occurred. High amounts of variation, for example, were found in product reviews and also in the object position of sentences. Besides allowing us to distinguish between situations with relatively high and relatively low individual variation in choices of referential form, this corpus introduces a new paradigm for the development and evaluation of models for referential choice. Rather than predicting the most likely form of a reference, as is usually done, the new corpus allows us to develop a model that can predict the frequency with which a particular reference can assume different referential forms. In this study, we explore this possibility.

4 Models

We model the individual variation in the choice of referential form in the following way: each reference consists of a tuple (X, y), where X is the set of feature values that describes the reference and y is a distribution of referential forms that indicates the frequency (in proportion) in which X assumes each form. So given X, we expect to find a distribution $\hat{y}$ similar to y.

Table 1 depicts the features used to describe X. The influence of those discourse factors in the choice of referential form has been often studied in the literature. Concerning syntactic position, Brennan (1995) argued that references in the subject position of a sentence are more likely to be shorter than references in the the object position. In favor of status and recency, Chafe (1994) showed that references to previously mentioned referents in the discourse and ones that are close to their antecedents are more likely to be shorter than references to new referents or ones that are distant from their antecedents.

All features were defined categorically, including the recency. This latter is treated by describing if a reference's antecedent is 10 or less words away, between 11 and 20 words, between 21 and 30 words, between 31 and 40 words and more than 40 words away.

To predict a distribution $\hat{y}$ based on X, we propose two models: a Naive Bayes and a Recurrent Neural Network.

4.1 Naive Bayes

Given a set of referential forms F, the probability that a reference assumes a particular form $f \in F$ according to this model is given by:

$$P(f \mid X) \propto \frac{P(f) \prod_{x \in X} P(x \mid f)}{\sum_{f' \in F} P(f') \prod_{x \in X} P(x \mid f')} \quad (1)$$

To avoid zero probabilities, we used additive smoothing with $\alpha = 2e^{-308}$. So given a reference described by X, $\hat{y}$ is the distribution over F:

$$\hat{y} = \begin{bmatrix} P(f_1 \mid X) \\ ... \\ P(f_{|F|} \mid X) \end{bmatrix} \quad (2)$$

4.2 Recurrent Neural Network

Some referential theories support the idea that a referential form is chosen based on previous choices to the same referent. Arnold (1998) argued that subjects of a sentence are more likely to be later pronominalized, as well as references in parallel syntactic position with their antecedents. Chafe (1994) sustained that referents mentioned in recent clauses also tend to be pronominalized. Since Naive Bayes does not take into account the sequential nature of text, we use a Recurrent Neural Network (RNN) to be able to take context into account.

RNN is a powerful structure to handle sequences of data. It can map a sequence of references $(X_1, ..., X_t)$ to their referential forms distributions $(y_1, ..., y_t)$ based on the previous steps.

Our approach here is similar to the one presented by Mesnil et al. (2013). But instead of word continuous representations, a referential embedding is created for each combination of feature values in X. So given a reference X_t and a context window size win, the embeddings of the references $X_{t-win/2}^{t-1}$, X_t and $X_{t+1}^{t+win/2}$ are merged to form a representation e_t. This representation is used in equations 3 and 4 to find a distribution over the referential forms that X_t could assume.

$$h_t = sigmoid(W^{hx} e_t + W^{hh} h_{t-1}) \quad (3)$$

$$\hat{y}_t = softmax(W^{yh} h_t) \quad (4)$$

Feature	Description
Syntactic position	Subject, object or a genitive noun phrase in the sentence.
Referential Status	First mention to the referent (new) or not (old) at the level of text, paragraph and sentence.
Recency	Distance between a given reference and the last, previous reference to the same referent.

Table 1: Features used to describe the references.

We assume a sequence of tuples $\{(X_1, y_1)..., (X_t, y_t)\}$ as all the references to a referent throughout a text.

We trained our RNN using Backpropagation Through Time. To measure the error among y and $\hat{y}$, we use cross entropy as a cost function. The values for the remaining parameters of the RNN are introduced in Table 2. We chose them based on an ad-hoc analysis, where we searched for an optimal combination to obtain the best predictions.

Batch Size	10
Context Window Size	3
Epochs	15
Embedding Dimension	50
Hidden Layer Size	50
Learning Rate	0.1

Table 2: RNN Settings

5 Individual Variation Experiments

For each reference slot encountered in the VaREG corpus, we evaluated how well a model takes the individual variation into account in the choice of referential form by comparing its predicted distribution of referential forms ($\hat{y}$) with the real distribution (y). We performed this comparison through two experiments.

In the first, the models were trained and tested with VaREG corpus. In the second, we aimed to check to what extent the referring expressions from the GREC-2.0 corpus are similar in form to the referring expressions from VaREG corpus by training the models with the first corpus and testing with the second.

5.1 Method

4-fold-cross-validation was used to train the models in the first experiment. The number of folds was chosen based on the set-up of the VaREG corpus, which consists of 4 groups of texts. Given the structure of the corpus, we decided that training our model with 3 groups of texts and testing it on the held-out group was the most natural solution to avoid overfitting. Each fold has the same amount of texts per genre.

Unlike VaREG, GREC-2.0 corpus does not have a set of referring expressions for the exact same reference. So, in the second experiment, the referential form distributions y were defined globally by grouping the references by X and computing the frequency of each referential form.

We also re-annoted the GREC-2.0 corpus to make it compatible with the VaREG corpus. In particular, we added features for status and recency to the GREC-2.0 corpus and made the terminology consistent beween the two corpora[1]. Both the VaREG corpus and the re-annotated GREC-2.0 corpus are publicly available[2].

5.2 Metrics

For each reference, Jensen-Shannon divergence (Lin, 1991) was used to measure the similarity between y and $\hat{y}$:

$$JSD(y||\hat{y}) = \frac{1}{2}D(y||m) + \frac{1}{2}D(\hat{y}||m) \quad (5)$$

$$\text{where } m = \tfrac{1}{2}(y + \hat{y})$$

In this measure, D is the Kullback-Leibler divergence (Kullback, 1968). The Jensen-Shannon divergence ranges from 0 to 1, in which 0 indicates full convergence of the two distributions and 1 full divergence. Therefore, a lower number indicates a better individual variation modeling.

To check the behaviour of $\hat{y}$ based on y in each reference, the referential forms of both distributions were ranked and their relation were analysed with the Spearman's rank correlation coefficient. This measure ranges between -1 and 1, where -1 indicates a fully opposed behaviour among the variables and 1 the exact same behaviour among them. 0 indicates a non-linear correlation among the involved variables.

[1] Texts also used in VaREG had their references removed from the GREC-2.0 version used in here.

[2] http://ilk.uvt.nl/~tcastrof/acl2016

5.3 Baselines

We considered two baseline models in the experiments. The first, called *Random*, assumes $\hat{y}$ as a random distribution of forms for each reference.

The second model, called *ParagraphStatus*, always chooses a proper name when the reference is to a new topic in the paragraph (the distribution will assume the value 1 to the proper name form and 0 to the others), and a pronoun otherwise (value 1 to the pronoun form and 0 to the others).

5.4 Results

5.4.1 Cross-validation on VaREG corpus

Models	JSD	$\rho_{y,\hat{y}}$
Random	0.63	-0.01
ParagraphStatus	0.43	0.66
NB+Syntax−Status−Recency	0.39	0.69
NB−Syntax+Status−Recency	0.32	**0.75**
NB−Syntax−Status+Recency	0.41	0.68
NB+Syntax+Status−Recency	**0.31**	**0.75**
NB+Syntax−Status+Recency	0.38	0.70
NB−Syntax+Status+Recency	0.33	0.73
NB+Syntax+Status+Recency	**0.31**	0.74
RNN+Syntax−Status−Recency	0.37	0.71
RNN−Syntax+Status−Recency	0.36	0.72
RNN−Syntax−Status+Recency	0.40	0.70
RNN+Syntax+Status−Recency	0.33	0.73
RNN+Syntax−Status+Recency	0.37	0.71
RNN−Syntax+Status+Recency	0.36	0.72
RNN+Syntax+Status+Recency	0.33	0.72

Table 3: Average Jensen-Shannon divergence and Spearman's correlation coefficient of the models in Experiment 1.

Table 3 depicts the Jensen-Shannon divergence and Spearman's correlation coefficient of the models cross-validated on VaREG corpus. All our models outperformed the baselines.

Considering the models in which the references are described by only one kind of feature, it seems that the status features (+Status) are the ones that best contributed to model the individual variation in the choice of referential form, whereas the recency (+Recency) is the worst. Syntactic position is sandwiched among the previous two.

In the comparison within Naive Bayes and RNN models, the ones in which the references are described by syntactic position and referential status (+Syntax+Status−Recency) obtained the best results for both measures. Figure 1 depicts the average Jensen-Shannon divergences by genre of Naive Bayes and RNN models in which the references are described by this combination of features. Both models presented the best results in

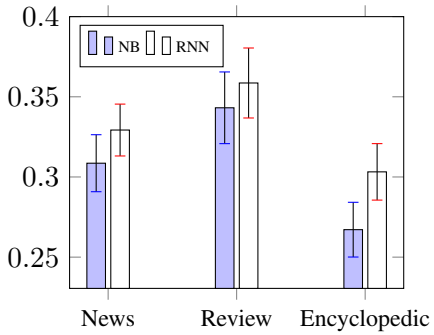


Figure 1: Jensen-Shannon divergence of NB+Syntax+Status−Recency (NB) and RNN+Syntax+Status−Recency (RNN) by genre in Experiment 1. Error bars represent 95% confidence intervals.

encyclopedic texts, and the worst in product reviews.

Although RNNs are able to model the individual variation in a reference based on its antecedents, they did not introduce significantly better results than Naive Bayes. In fact, NB+Syntax+Status−Recency is significantly better than RNN+Syntax+Status−Recency in modeling the individual variation in news (Wilcoxon $Z = 11574.5$, $p < 0.01$) and encyclopedic texts (Wilcoxon $Z = 4232.5$, $p < 0.001$).

5.4.2 Training on GREC-2.0 and evaluating on VaREG corpus

Models	JSD	$\rho_{y,\hat{y}}$
Random	0.63	-0.01
ParagraphStatus	0.43	0.66
NB+Syntax+Status−Recency	**0.36**	**0.67**
NB+Syntax+Status+Recency	0.37	0.64
RNN+Syntax+Status−Recency	0.37	0.62
RNN+Syntax+Status+Recency	0.37	0.64

Table 4: Average Jensen-Shannon divergence and Spearman's correlation coefficient of the models in Experiment 2.

Table 4 shows the results of models trained with GREC-2.0 and tested with VaREG corpus. These models are the two versions of Naive Bayes, and the two versions of RNN which were best evaluated in the previous experiment.

The results of this experiment follow the results of the previous one. Our models outperformed the baselines and NB+Syntax+Status−Recency was the model that obtained the best results for both measures.

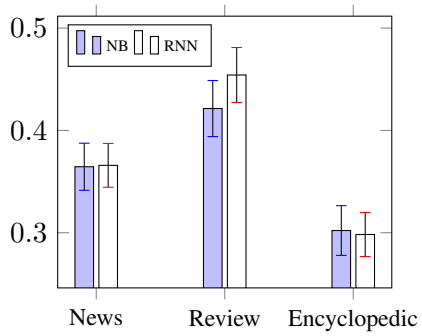


Figure 2: Average Jensen-Shannon divergence of NB+Syntax+Status−Recency (NB) and RNN+Syntax+Status-Recency (RNN) by genre in Experiment 2. Error bars represent 95% confidence intervals.

Figure 2 depicts the Jensen-Shannon divergence measures of models NB+Syntax+Status-Recency and RNN+Syntax+Status-Recency by text genre. As in the previous experiment, both Naive Bayes and RNN models best modeled the individual variation in encyclopedic texts. Moreover, there was not significant difference among NB+Syntax+Status-Recency and RNN+Syntax+Status-Recency in the three text genres.

In general, the models trained with VaREG corpus seemed to model the individual variation in the choice of referential form better than the models trained with GREC-2.0 corpus.

6 Coherence and comprehensibility of the texts

In this section, we investigate to what extent texts generated by our method, including variation of referential form, are judged coherent and comprehensible by readers. We do this by comparing texts from the GREC-2.0 corpus in which all references were (re)generated using our method, with the original text and with a variant that includes random variation of referential form.

6.1 Our model for choice of referential form

To generate the referring expressions for the topic of a given text of GREC-2.0, we first group all references by syntactic position and referential status values. Then for each group, we shuffle the references and choose their forms according to the distribution predicted by our best performing model (the NB+Syntax+Status−Recency trained on VaREG). The choice of referential forms follows the roulette-wheel generation process (Belz, 2008). This process entails that if a group has 5 references and our model predicts a distribution of 0.75 proper names and 0.25 pronouns, 4 references of the group will be proper names and 1 a pronoun.

This covers the selection of referential forms (deciding which form to use at which particular point in the text). To deal with their linguistic realisation, we implemented the following heuristics. For the cases in which a proper name reference is selected, we choose a realization depending on referential status. If the reference is the first mention to the topic in the text, the reference is realized with the topic's longest proper name. Otherwise, the reference is realized with its shortest proper name. For the cases in which a definite description is selected, but where the original GREC-2.0 corpus does not provide a description for the topic, we select the shortest predicate adjective of the first sentence of the text, immediately following the main verb. For instance, for the sentence *"Alan Mathison Turing was an English mathematician, logician, and cryptographer."*, the selected definite description would be "**The English mathematician**". In the cases where a reference should assume the form of a demonstrative, the definite article of the definite description is replaced by the demonstrative "this" (In the previous example, "**This English mathematician**").

6.2 Evaluation Method

We evaluated three versions of each text. The *Original* is the original text in the corpus, including the original referring expressions selected by the author. We compare this with a *Random* variant, which does include variation of referential forms, but selects them in a fully random way. Finally, in the third, *Generated* version, all references are generated according to the method outlined at Section 6.1. Table 5 depicts an example of text in the three versions.

In total, we make 3 versions of 9 pseudo-randomly selected texts (5 covering animate topics and 4 inanimate ones, varying in length) from the GREC-2.0 corpus, yielding 27 texts in total. These were distributed over 3 lists, such that each list contains one variant of each text, and there is an equal number of texts from the 3 conditions (*Original*, *Random*, *Generated*). In all texts, all

Version	Text
Original	**Spain**, officially the Kingdom of Spain, is a country located in Southern Europe, with two small exclaves in North Africa (both bordering Morocco). **Spain** is a democracy **which** is organized as a parliamentary monarchy. **It** is a developed country with the ninth-largest economy in the world. **It** is the largest of the three sovereign nations that make up the Iberian Peninsula–the others are Portugal and the microstate of Andorra.
Random	**It**, officially the Kingdom of Spain, is a country located in Southern Europe, with two small exclaves in North Africa (both bordering Morocco). **The country** is a democracy **that** is organized as a parliamentary monarchy. **It** is a developed country with the ninth-largest economy in the world. **This country** is the largest of the three sovereign nations that make up the Iberian Peninsula–the others are Portugal and the microstate of Andorra.
Generated	**Spain**, officially the Kingdom of Spain, is a country located in Southern Europe, with two small exclaves in North Africa (both bordering Morocco). **Spain** is a democracy **that** is organized as a parliamentary monarchy. **The country** is a developed country with the ninth-largest economy in the world. **It** is the largest of the three sovereign nations that make up the Iberian Peninsula–the others are Portugal and the microstate of Andorra.

Table 5: Example of text in the Original, Random and Generated version.

references to the topic were highlighted in yellow. The experiment was run on CrowdFlower and is publicly available[3].

The experiment was performed by 30 participants (10 per list). Their average age was 36 years, and 22 were female. All were proficient in English (the language of the experiment), 26 participants were native speakers. They were asked to rate each text in terms of how coherent and comprehensible they considered it, on a scale from 1 (Very Bad) to 5 (Very Good).

6.3 Results

Figure 3 depicts the average coherence and comprehensibility of the texts where their topics are described by the *Original*, *Random* and *Generated* approaches, respectively. Inspection of this Figure clearly shows that the *Random* texts are rated lower than both the *Original* and the *Generated* texts, and that the latter are rated very similarly on both dimensions.

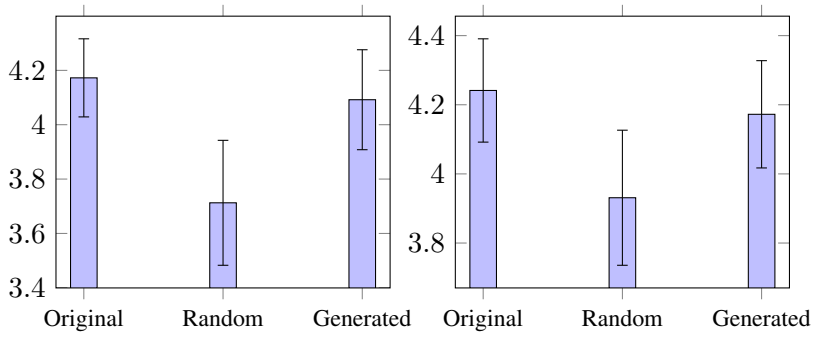
This is confirmed by the statistical analysis. According to a Friedman test, there is statistically significant difference in the coherence ($\chi^2 = 11.79$, $p < 0.005$) and comprehensibility ($\chi^2 = 8.98$, $p = 0.01$) for the three kinds of texts. We then conducted a post hoc analysis with Wilcoxon signed-rank test corrected for multiple comparisons using the Bonferroni method, resulting in a significance level set at $p < 0.017$. Texts of the *Original* approach are statistically more coherent ($Z = 322$, $p < 0.017$) and comprehensible ($Z = 407.5$, $p < 0.017$) than texts of the *Random* one. Texts of the *Generated* approach are also statistically more coherent ($Z = 275$, $p < 0.017$), but not more comprehensible ($Z = 378$, $p < 0.05$) than texts of the *Random* one. Finally, and crucially, comparing *Original* and *Generated* texts revealed no significant differences for coherence ($Z = 540$, $p < 0.5$) nor for comprehensibility ($Z = 391.5$, $p < 0.5$).

[3] http://ilk.uvt.nl/~tcastrof/acl2016

7 Discussion

In this paper we explored the possibilities of introducing more variation in automatically generated texts, by trying to model individual variation in the selection of referential form. We relied on a new corpus (VaREG (Ferreira et al., 2016)), which does not contain a single expression for each reference in a text, but rather a distribution of referential forms produced by 20 different people. In contrast to earlier models for referential choice which always deterministically choose the most likely form of a reference, we proposed a Naive Bayes and a Recurrent Neural Network model which aimed to predict the frequency distribution with which a reference can assume a specific referential form, based on discourse features including syntactic position, referential status and recency. Given a reference, we evaluated how well each different model could capture the individual variation found in the VaREG corpus by comparing its predicted distribution of referential forms with the real one in the corpus. We trained the models in two different ways: first using the VaREG, and second using the GREC-2.0 corpus. The Naive Bayes model, trained on VaREG corpus, in which the references were described by syntactic position and referential status features was the one that best modeled the individual variation in the choice of referential form.

Features Referential status features were the most helpful for modeling the individual variation in the choice of referential form. They were fol-



(a) Coherence (b) Comprehensibility

Figure 3: Average coherence (3a) and comprehensibility (3b) of the texts with the original, randomized and generated referring expressions. Error bars represent 95% confidence intervals.

lowed by the syntactic position feature. Both of these findings are consistent with the observations about human variation in the selection of referential forms, as discussed by Ferreira et al. (2016). This study argued that writers are more likely to vary in their choices when a reference is in the object position, and when it is an old mention in the text, but new in the sentence. Recency was not a helpful feature for our models, and this may be due to the way the feature was represented - i.e., as a categorical rather than a continuous feature. Moreover, the recency feature was measured in terms of words between the current reference and the most recent previous one to the same referent. Perhaps, it would be better to measure recency in terms of different discourse entities mentioned between two references to the same referent.

Genre In agreement with Ferreira et al. (2016), we also found that genre mattered. For modeling variation, our models performed best when applied to encyclopedic texts, and worst in product reviews, with news sandwiched in between.

Naive Bayes model vs. RNNs Although the RNNs were able to model individual variation in the choice of referential form to some extent, they did not perform significantly better than the Naive Bayes models, which might have to do with the relatively small dataset. However, we think the size of the corpus matches the relatively low complexity of the problem we address. In the most complex case (i.e., when a reference is described by its syntactic position, status and recency), an input can be represented in 120 different ways to predict a multinomial distribution of size 5 (number of referential forms). This complexity is much smaller than other problems typically modeled by RNNs. In text production, for instance, an input may be represented by thousands of words to predict a large multinomial distribution over a vocabulary (Sutskever et al., 2014). Additionally, it is important to stress that we actually have a real multinomial distribution to compare with the distribution predicted by the RNN in each situation. We observed that it is possible to compute more fine-grained error costs in our case, which makes the RNN converge faster when it is backpropagated. In sum, we believe that those two factors combined compensate for the size of the dataset. A possible explanation for the non-difference among the Naive Bayes model and RNNs is the use of the referential status features, which perhaps are already enough to model the relation among a reference and its antecedents.

VaREG corpus vs. GREC-2.0 corpus Interestingly, our proposed models yielded better performance when trained on the VaREG than on the GREC-2.0 corpus. This shows a difference among the referential choices of both corpora. We conjecture this difference is partly due to differences in text genres, since the VaREG corpus contains texts from three different genres, whereas the GREC-2.0 corpus only has encyclopedic texts. Earlier work has also highlighted the influence of text genre on the amount of individual variation in writers' choices for referential forms (Ferreira et al., 2016).

Coherence and comprehensibility In the second part of the study, we used the best performing model to generate referential forms in texts from the GREC-2.0 corpus, using a roulette-based model sampling from the predicted distributions over referential forms. We evaluated the texts gen-

erated in this way in an experiment in which humans were asked to judge the coherence and comprehensibility of the generated texts, comparing them both with the original references and those produced by a random baseline model. In terms of coherence and comprehensibility, we found that the texts in which the references were generated by our model were not significantly different than the human generated ones, and significantly better than the randomly generated ones. This shows that our solution does not only model the individual variation in the choice of referential form, but that this also does not negatively affect the quality of the texts. This is an important step towards developing new models for automatic text generation that are less predictable and more varied.

Acknowledgments

This work has been supported by the National Council of Scientific and Technological Development from Brazil (CNPq).

References

Jennifer E Arnold. 1998. *Reference form and discourse patterns*. Ph.D. thesis, Stanford University Stanford, CA.

Anja Belz. 2008. Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Nat. Lang. Eng.* 14(4):431–455.

Anja Belz, Eric Kow, Jette Viethen, and Albert Gatt. 2010. Empirical methods in natural language generation. Springer-Verlag, Berlin, Heidelberg, chapter Generating Referring Expressions in Context: The GREC Task Evaluation Challenges, pages 294–327.

Susan E. Brennan. 1995. Centering attention in discourse. *Language and Cognitive Processes* 10(2):137–167.

Charles B. Callaway and James C. Lester. 2002. Pronominalization in generated discourse and dialogue. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '02, pages 88–95.

Wallace L. Chafe. 1994. *Discourse, Consciousness, and Time: The Flow and Displacement of Conscious Experience in Speaking and Writing*. University of Chicago Press.

Christer Clerwall. 2014. Enter the robot journalist: Users' perceptions of automated content. *Journalism Practice* 8(5):519–531.

Robert Dale and Jette Viethen. 2010. Attribute-centric referring expression generation. In *Empirical methods in natural language generation*, Springer, pages 163–179.

Thiago Castro Ferreira, Emiel Krahmer, and Sander Wubben. 2016. Individual variation in the choice of referential form. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California.

Charles F Greenbacker and Kathleen F McCoy. 2009. Feature selection for reference generation as informed by psycholinguistic research. In *Proceedings of the CogSci 2009 Workshop on Production of Referring Expressions (PRE-Cogsci 2009)*.

Samir Gupta and Sivaji Bandopadhyay. 2009. Junlg-msr: A machine learning approach of main subject reference selection with rule based improvement. In *Proceedings of the 2009 Workshop on Language Generation and Summarisation*. Association for Computational Linguistics, Stroudsburg, PA, USA, UCNLG+Sum '09, pages 103–104.

Renate Henschel, Hua Cheng, and Massimo Poesio. 2000. Pronominalization revisited. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*. Association for Computational Linguistics, pages 306–312.

Emiel Krahmer and Mariët Theune. 2002. Efficient context-sensitive generation of referring expressions. In K. van Deemter and R. Kibble, editors, *Information sharing: Reference and presupposition in language generation and interpretation*, CSLI, Stanford, CA, pages 223–264.

Solomon Kullback. 1968. *Information theory and statistics*. Courier Corporation.

Jianhua Lin. 1991. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on* 37(1):145–151.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and

learning methods for spoken language understanding. In *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, Lyon, France, August 25-29, 2013*. pages 3771–3775.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY, USA.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*. pages 3104–3112.

Kees Van Deemter, Albert Gatt, Roger PG van Gompel, and Emiel Krahmer. 2012. Toward a computational psycholinguistics of reference production. *Topics in cognitive science* 4(2):166–183.

Ching-Long Yeh and Chris Mellish. 1997. An empirical study on the generation of anaphora in chinese. *Comput. Linguist.* 23(1):171–190.

How Much is 131 Million Dollars? Putting Numbers in Perspective with Compositional Descriptions

Arun Tejasvi Chaganty
Computer Science Department
Stanford University
chaganty@cs.stanford.edu

Percy Liang
Computer Science Department
Stanford University
pliang@cs.stanford.edu

Abstract

How much is 131 million US dollars? To help readers put such numbers in context, we propose a new task of automatically generating short descriptions known as perspectives, e.g. "$131 million is about the cost to employ everyone in Texas over a lunch period". First, we collect a dataset of numeric mentions in news articles, where each mention is labeled with a set of rated perspectives. We then propose a system to generate these descriptions consisting of two steps: formula construction and description generation. In construction, we compose formulae from numeric facts in a knowledge base and rank the resulting formulas based on familiarity, numeric proximity and semantic compatibility. In generation, we convert a formula into natural language using a sequence-to-sequence recurrent neural network. Our system obtains a 15.2% F_1 improvement over a non-compositional baseline at formula construction and a 12.5 BLEU point improvement over a baseline description generation.

1 Introduction

When posed with a mention of a number, such as "Cristiano Ronaldo, the player who Madrid acquired for [...] a *$131 million*" (Figure 1), it is often difficult to comprehend the scale of large (or small) absolute values like *$131 million* (Paulos, 1988; Seife, 2010). Studies have shown that providing relative comparisons, or *perspectives*, such as "about the cost to employ everyone in Texas over a lunch period" significantly improves comprehension when measured in terms of memory retention or outlier detection (Barrio et al., 2016).

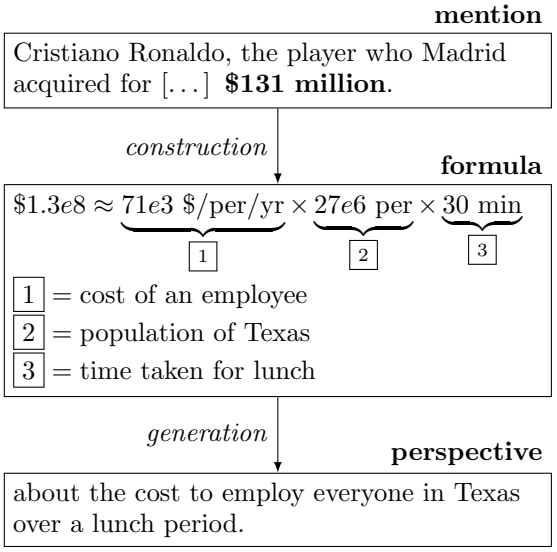


Figure 1: An overview of the perspective generation task: given a *numeric mention*, generate a short description (a *perspective*) that allows the reader to appreciate the scale of the mentioned number. In our system, we first construct a *formula* over facts in our knowledge base and then generate a description of that formula.

Previous work in the HCI community has relied on either manually generated perspectives (Barrio et al., 2016) or present a fact as is from a knowledge base (Chiacchieri, 2013). As a result, these approaches are limited to contexts in which a relevant perspective already exists.

In this paper, we generate perspectives by composing facts from a knowledge base. For example, we might describe *$100,000* to be "about twice the median income for a year", and describe *$5 million* to be the "about how much the average person makes over their lifetime". Leveraging compositionality allows us to achieve broad coverage of numbers from a relatively small collection of familiar facts, e.g. median income and a person's

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 578–587,
Berlin, Germany, August 7-12, 2016.

lifetime.

Using compositionality in perspectives is also concordant with our understanding of how people learn to appreciate scale. Jones and Taylor (2009) find that students learning to appreciate scale do so mainly by *anchoring* with familiar concepts, e.g. $50,000 is slightly less than the median income in the US, and by *unitization*, i.e. improvising a system of units that is more relatable, e.g. using the Earth as a measure of mass when describing the mass of Jupiter to be that of 97 Earths. Here, compositionality naturally unitizes the constituent facts: in the examples above, money was unitized in terms of median income, and time was unitized in a person's lifetime. Unitization and anchoring have also been proposed by Chevalier et al. (2013) as the basis of a design methodology for constructing visual perspectives called concrete scales.

When generating compositional perspectives, we must address two key challenges: constructing familiar, relevant and meaningful formulas and generating easy-to-understand descriptions or perspectives. We tackle the first challenge using an overgenerate-and-rank paradigm, selecting formulas using signals from familiarity, compositionality, numeric proximity and semantic similarity. We treat the second problem of generation as a translation problem and use a sequence-to-sequence recurrent neural network (RNN) to generate perspectives from a formula.

We evaluate individual components of our system quantitatively on a dataset collected using crowdsourcing. Our formula construction method improves on F_1 over a non-compositional baseline by about 17.8%. Our generation method improves over a simple baseline by 12.5 BLEU points.

2 Problem statement

The input to the *perspective generation* task is a sentence s containing a *numeric mention* x: a span of tokens within the sentence which describes a quantity with value $x.\text{value}$ and of unit $x.\text{unit}$. In Figure 1, the numeric mention x is "$131 million", $x.\text{value} = 1.31e8$ and $x.\text{unit} = \$$. The output is a description y that puts x in perspective.

We have access to a knowledge base $\mathcal{K}$ with numeric tuples $t = (t.\text{value}, t.\text{unit}, t.\text{description})$. Table 1 has a few examples of tuples in our knowledge base. Units (e.g. $/per/yr) are fractions composed either of fundamental units (length, area, volume, mass, time) or of ordinal units (e.g. cars, people, etc.).

Description	Value Unit
cost of an employee	71e3 $/year/person
population of Texas	27e3 person
number of employees at Google	57e3 person
average household size	2.54 person
time taken for a basketball game	60 minute
average lifetime for a person	79 year
a week	1 week
time taken for lunch	30 minute
cost of property in the Bay area	1e3 $/ft^2
area of a city block	10e3 m^2

Table 1: A subset of our knowledge base of numeric tuples. Tuples with fractional units (e.g. $/ft^2) can be combined with other tuples to create formulas.

The first step of our task, described in Section 4, is to construct a *formula* f over numeric tuples in $\mathcal{K}$ that has the same value and unit as the numeric mention x. A valid formula comprises of an arbitrary multiplier $f.m$ and a sequence of tuples $f.\text{tuples}$. The value of a formula, $f.\text{value}$, is simply the product of the multiplier and the values of the tuples, and the unit of the formula, $f.\text{unit}$, is the product of the units of the tuples. In Figure 1, the formula has a multiplier of 1 and is composed of tuples $\boxed{1}$, $\boxed{2}$ and $\boxed{3}$; it has a value of $1.3e8$ and a unit of $.

The second step of our task, described in Section 5, is to generate a *perspective* y, a short noun phrase that realizes f. Typically, the utterance will be formed using variations of the descriptions of the tuples in $f.\text{tuples}$.

3 Dataset construction

We break our data collection task into two steps, mirroring formula selection and description generation: first, we collect descriptions of formulas constructed exhaustively from our knowledge base (for generation), and then we use these descriptions to collect preferences for perspectives (for construction).

Collecting the knowledge base. We manually constructed a knowledge base with 142 tuples and 9 fundamental units[1] from the United States Bu-

[1]Namely, length, area, volume, time, weight, money, people, cars and guns. These units were chosen because they

reau of Statistics, the orders of magnitude topic on Wikipedia and other Wikipedia pages. The facts chosen are somewhat crude; for example, though "the cost of an employee" is a very context dependent quantity, we take its value to be the median cost for an employer in the United States, $71,000. Presenting facts at a coarse level of granularity makes them more familiar to the general reader while still being appropriate for perspective generation: the intention is to convey the right scale, not necessarily the precise quantity.

Collecting numeric mentions. We collected 53,946 sentences containing numeric mentions from the newswire section of LDC2011T07 using simple regular expression patterns like `$([0-9]+(,[0-9]+)*(.[0-9]+)? ((hundred)|(thousand)|(million)|(billion)|(trillion)))`. The values and units of the numeric mentions in each sentence were normalized and converted to fundamental units (e.g. from miles to length). We then randomly selected up to 200 mentions of each of the 9 types in bins with boundaries $10^{-3}, 1, 10^3, 10^6, 10^9, 10^{12}$ leading to 4,931 mentions that are stratified by unit and magnitude.[2] Finally, we chose mentions which could be described by at least one numeric expression, resulting in the 2,041 mentions that we use in our experiments (Figure 2). We note that there is a slight bias towards mentions of money and people because these are more common in the news corpus.

Generating formulas. Next, we exhaustively generate valid formulas from our knowledge base. We represent the knowledge base as a graph over units with vertices and edges annotated with tuples (Figure 3). Every vertex in this graph is labeled with a unit u and contains the set of tuples with this unit: $\{t \in \mathcal{K} : t.\text{unit} = u\}$. Additionally, for every vertex in the graph with a unit of the form u_1/u_2, where u_2 has no denominator, we add an edge from u_1/u_2 to u_1, annotated with all tuples of type u_2: in Figure 3 we add an edge from *money/person* to *money* annotated with the three person tuples in Table 1. The set of formulas with unit u is obtained by enumerating all paths in the graph which terminate at the vertex u. The multiplier of the formula is set so that the value of

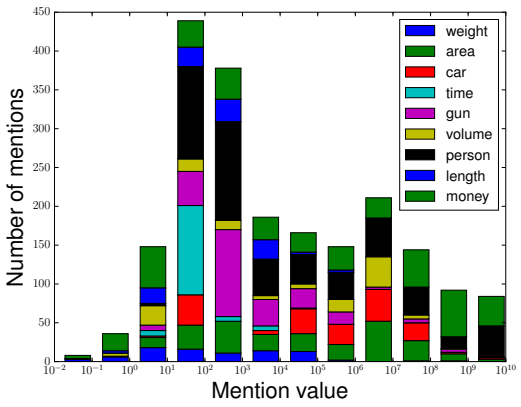


Figure 2: A histogram of the absolute values of numeric mentions by type. There are 100–300 mentions of each unit.

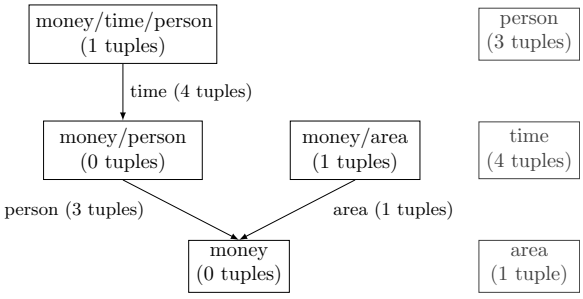


Figure 3: The graph over tuples generated from the knowledge base subset in Table 1.

the formula matches the value of the mention. For example, the formula in Figure 1 was constructed by traversing the graph from *money/time/person* to *money*: we start with a tuple in *money/time/person* (*cost of an employee*) and then multiply by a tuple with unit *time* (*time for lunch*) and then by unit *person* (*population of Texas*), thus traversing two edges to arrive at *money*.

Using the 142 tuples in our knowledge base, we generate a total of 1,124 formulas sans multiplier.

Collecting descriptions of formulas. The main goal of collecting descriptions of formulas is to train a language generation system, though these descriptions will also be useful while collecting training data for formula selection. For every unit in our knowledge base and every value in the set $\{10^{-7}, 10^{-6} \ldots, 10^{10}\}$, we generated all valid formulas. We further restricted this set to formulas with a multiplier between $1/100$ and 100, based on the rationale that human cognition of scale sharply drops beyond an order of magnitude (Tretter et al.,

were well represented in the corpus.

[2]Some types had fewer than 200 mentions for some bins.

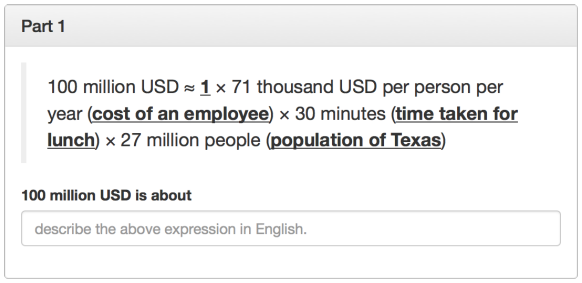


Figure 4: A screenshot of the crowdsourced task to generate natural language descriptions, or perspectives, from formulas.

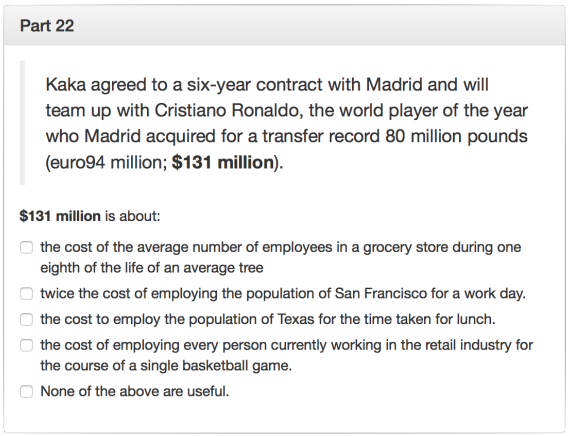


Figure 5: A screenshot of the crowdsourced task to identify which formulas are useful to crowdworkers in understanding the highlighted mentioned number.

2006). In total, 5000 formulas were presented to crowdworkers on Amazon Mechanical Turk, with a prompt asking them to rephrase the formula as an English expression (Figure 4).[3] We obtained 5–7 descriptions of each formula, leading to a total of 31,244 unique descriptions.

Collecting data on formula preference. Finally, given a numeric mention, we ask crowdworkers which perspectives from the description dataset they prefer. Note that formulas generated for a particular mention may differ in multiplier with a formula in the description dataset. We thus relax our constraints on factual accuracy while collecting this formula preference dataset: for each mention x, we choose a random perspective from the description dataset described above corresponding to a formula whose value is within a factor of 2 from the mention's value, $x.\text{value}$. A smaller factor led to too many mentions without a valid comparison, while a larger one led to blatant factual inaccuracies. The perspectives were partitioned into sets of four and displayed to crowdworkers along with a "None of the above" option with the following prompt: "We would like you to pick up to two of these descriptions that are useful in understanding the scale of the highlighted number" (Figure 5). A formula is rated to be useful by simple majority.[4]

Figure 6 provides a summary of the dataset collected, visualizing how many formulas are useful, controlling for the size of the formula. The exhaustive generation procedure produces a large number of spurious formulas like "20 × trash generated in the US × a minute × number of employees on Medicare". Nonetheless, compositional formulas are quite useful in the appropriate context; Table 2 presents some mentions with highly rated perspectives and formulas.

4 Formula selection

We now turn to the first half of our task: given a numeric mention x and a knowledge base $\mathcal{K}$, select a formula f over $\mathcal{K}$ with the same value and unit as the mention. It is easy to generate a very large number of formulas for any mention. For the example, "Cristiano Ronaldo, the player who Madrid acquired for [. . .] $131 million.", the small knowledge base in Table 1 can generate the 12 different formulas,[5] including the following:

1. 1 × the cost of an employee × the population of Texas × the time taken for lunch.
2. 400 × the cost of an employee × average household size × a week.
3. 1 × the cost of an employee × number of employees at Google × a week.
4. 1 × cost of property in the Bay Area × area of a city block.

Some of the formulas above are clearly worse than others: the key challenge is picking a formula that will lead to a meaningful and relevant perspective.

Criteria for ranking formulas. We posit the following principles to guide our choice in features (Table 3).

[3] Crowdworkers were paid $0.08 per description.

[4] Crowdworkers were paid $0.06 to vote on each set of perspectives.

[5] The full knowledge base described in Section 3 can generate 242 formulas with the unit money (sans multiplier).

Sentence	That's about ...	Formula
The Billings-based Stillwater Mining produced **601,000 ounces of platinum**.	4 times the weight of an elephant.	4 × weight of an elephant.
Authorities estimate there are about **60 million guns** in Yemen.	twice the gun ownership of the population of Texas	2 × gun ownership × population of Texas
Water is flowing into Taihu lake at a rate of **150 cubic meters** per second.	how much water would flow from a tap left on for a week.	rate of flow of water from tap × a week
The bank had held auctions, selling around **US$1 billion** worth of three-month bills.	half the cost of employing the population of Texas for a work day.	1/2 × cost of an employee × time taken for a work day × population of Texas
The government[s] have promised to rent about **1.2 million sq. feet**.	the area of forest logged in a single minute	90 × area of forest logged × a minute

Table 2: Examples of numeric mentions, perspectives and their corresponding formulas in the dataset. All the examples except the last one are rated to be useful by crowdworkers.

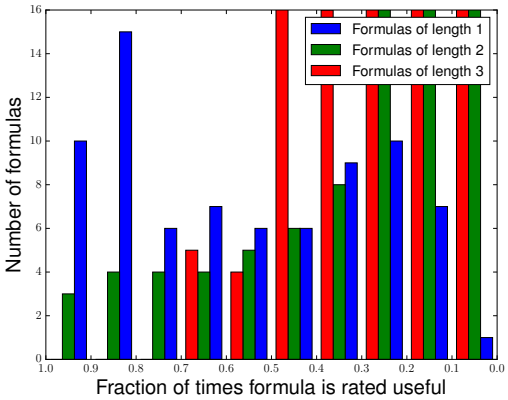


Figure 6: A histogram comparing formula length to ratings of usefulness (clipped for readability). Non-compositional perspectives with a single tuple are broadly useful. Useful compositional perspectives tend to be more context-specific than non-compositional ones, and many of the formulas that can be generated from the knowledge base are spurious.

Type	Features	#
Proximity	$\text{sign}(\log(f.m)), \lvert\log(f.m)\rvert$	1
Familiarity	$\mathbb{I}[t]$	142
Compatibility	$\mathbb{I}[t, t']$	20022
Similarity	$\text{wvec}(s)^\top$ $\text{wvec}(t.\text{description})$	1

Table 3: Feature templates used to score a formulas f and their counts (#), where $f.m$ is the formula's multiplier and $t, t' \in f.\text{tuples}$ are tuples in the formula.

Proximity: *A numeric perspective should be within an order of magnitude of the mentioned value*. Conception of scale quickly fails with quantities that exceed "human scales" (Tretter et al., 2006): numbers that are significantly away from $1/10$ and 10. We use this principle to prune formulas with multipliers not in the range $[1/100, 100]$ (e.g. example 2 above) and introduce features for numeric proximity.

Familiarity: *A numeric perspective should be composed of concepts familiar to the reader*. The most common technique cited by those who do well at scale cognition tests is reasoning in terms of familiar objects (Tretter et al., 2006; Jones and Taylor, 2009; Chevalier et al., 2013). Intuitively, the average American reader may not know exactly how many people are in Texas, but is familiar enough with the quantity to effectively reason using Texas' population as a unit. On the other hand, it is less likely that the same reader is familiar with even the concept of Angola's population.

Of course, because it is so personal, familiarity is difficult to capture. With additional information about the reader, e.g. their location, it is possible to personalize the chosen tuples (Kim et al., 2016). Without this information, we back off to a global preference on tuples by using indicator features for each tuple in the formula.

Formula	Score
Studies estimate **36,000 people** die on average each year from seasonal flu.	
1/4 × global death rate × a day	0.67
5 × death rate in the US × a day	0.64
1/3 × number of employees at Microsoft	0.60
Gazprom's exports to Europe [...] will total **60 billion cubic meters** ...	
oil produced by the US × average lifetime	0.78
average coffee consumption × population of the world × average lifetime	0.78
2 × average coffee consumption × population of Asia × average lifetime	0.73

Table 4: The top three examples outputted by the ranking system with the scores reported by the system.

Compatibility: Similarly, some tuple combinations are more natural ("median income × a month") while others are less so ("weight of a person × population of Texas"). We model compatibility between tuples in a formula using an indicator feature.

Similarity: *A numeric perspective should be relevant to the context*. Apart from helping with scale cognition, a perspective should also place the mentioned quantity in appropriate context: for example, NASA's budget of $17 billion could be described as 0.1% of the United States' budget or the amount of money it could cost to feed Los Angeles for a year. While both perspectives are appropriate, the former is more relevant than the latter.

We model context relevance using word vector similarity between the tuples of the formula and the sentence containing the mention as a proxy for semantic similarity. Word vectors for a sentence or tuple description are computed by taking the mean of the word vectors for every non-stop-word token. The word vectors at the token level are computed using word2vec (Mikolov et al., 2013).

Evaluation. We train a logistic regression classifier using the features described in Table 3 using the perspective ratings collected in Section 3. Recall that the formula for each perspective in the dataset is assigned a positive ("useful") label if it was labeled to be useful to the majority of the workers. Table 5a presents results on classifying formulas as useful with a feature ablation.[6]

Familiarity and compatibility are the most useful features when selecting formulas, each having a significant increase in F_1 over the proximity baseline. There are minor gains from combining these two features. On the other hand, semantic similarity does not affect performance relative to the baseline. We find that this is mainly due to the disproportionate number of unfamiliar formulas present in the dataset that drown out any signal. Table 4 presents two examples of the system's ranking of formulas.

5 Perspective generation

Our next goal is to generate natural language descriptions, also known as perspectives, given a formula. Our approach models the task as a sequence-to-sequence translation task from formulas to natural language. We first describe a rule-based baseline and then describe a recurrent neural network (RNN) with an attention-based copying mechanism (Jia and Liang, 2016).

Baseline. As a simple approach to generate perspectives, we just combine tuples in the formula with the neutral prepositions *of* and *for*, e.g. "1/5th *of* the cost of an employee *for* the population of Texas *for* the time taken for lunch."

Sequence-to-sequence RNN. We use formula-perspective pairs from the dataset to create a sequence-to-sequence task: the input is composed using the formula's multiplier and descriptions of its tuples connected with the symbol '*'; the output is the perspective (Figure 7).

Our system is based on the model described in Jia and Liang (2016). Given a sequence of input tokens ($\mathbf{x} = (x_i)$), the model computes a context-dependent vector ($\mathbf{b} = (b_i)$) for each token using a bidirectional RNN with LSTM units. We then generate the output sequence (y_j) left to right as follows. At each output position, we have a hidden state vector (s_j) which is used to produce an "attention" distribution ($\boldsymbol{\alpha}_j = (\alpha_{ji})$) over input tokens: $\alpha_{ji} = \mathrm{Attend}(s_j, b_i)$. This distribution is used to generate the output token and update the hidden state vector. To generate the token, we ei-

[6] Significance results are computed by the bootstrap test as described in Berg-Kirkpatrick et al. (2012) using the output of classifiers trained on the entire training set.

Feature set	Train			Dev		
	P	R	F_1	P	R	F_1
Proximity	56.4	48.7	52.2	56.3	48.8	52.3
Similarity	65.1	34.9	45.4	65.1	34.9	45.4
Familiarity*	70.5	63.5	66.8	69.6	62.9	66.1
Compatibility$^+$	66.9	**74.4**	70.4	65.4	**73.1**	69.0
F + C†	**73.8**	70.3	**72.1**	**71.5**	68.9	**70.1**
F + C + P†	**73.8**	70.3	**72.1**	**71.5**	68.9	**70.1**
F + C + P + S†	**73.8**	70.3	72.0	71.4	68.6	69.9

(a) the formula construction system. Precision, Recall and F_1 are cross-validated on 10-folds. *significant F_1 versus P and S with $p < 0.01$. $^+$significant F_1 versus P, S and F with $p < 0.01$. †significant F_1 versus P, S, F and C with $p < 0.05$.

System	Train BLEU	Test BLEU
Baseline	65.00	57.32
RNN*	**81.50**	**69.79**

(b) the description generation system. *significant BLEU score versus the baseline with $p < 0.01$.

Table 5: Evaluation of perspective generation subsystems.

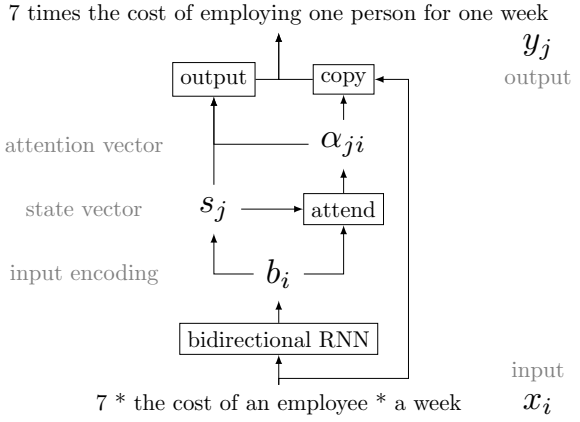


Figure 7: We model description generation as a sequence transduction task, with input as formulas (at bottom) and output as perspectives (at top). We use a RNN with an attention-based copying mechanism.

ther sample a word from the current state or copy a word from the input using attention. Allowing our model to copy from the input is helpful for our task, since many of the entities are repeated verbatim in both input and output. We refer the reader to Jia and Liang (2016) for more details.

Evaluation. We split the perspective description dataset into a training and test set such that no formula in the test set contains the same set of tuples as a formula in the training set.[7] Table 5b compares the performance of the baseline and sequence-to-sequence RNN using BLEU. The sequence-to-sequence RNN performs significantly better than the baseline, producing more natural rephrasings. Table 6 shows some output generated by the system (see Table 6).

6 Human evaluation

In addition to the automatic evaluations for each component of the system, we also ran an end-to-end human evaluation on an independent set of 211 mentions collected using the same methodology described in Section 3. Crowdworkers were asked to choose between perspectives generated by our full system (LR+RNN) and those generated by the baseline of picking the numerically closest tuple in the knowledge base (BASELINE). They could also indicate if either both or none of the shown perspectives appeared useful.[8]

Table 7 summarizes the results of the evaluation and an error analysis conducted by the authors. Errors were characterized as either being errors in generation (e.g. Table 6) or violations of the criteria in selecting good formulas described in Section 4 (Table 7c). The *other* category mostly contains cases where the output generated by LR+RNN appears reasonable by the above criteria but was not chosen by a majority of workers. A few of the mentions shown did not properly describe a numeric quantity, e.g. "... claimed responsibility for a *2009 gun* massacre ..." and were labeled *invalid mentions*. The most common error is the selection of a formula that is not contextually relevant to the mentioned text because no such

[7]Note that formulas with the same set of tuples can occur multiple times in the either the training or test set with different multipliers.

[8]Crowdworkers were paid $0.06 per to choose a perspective for each mention. Each mention and set of perspectives were presented to 5 crowdworkers.

Input formula	Generated perspective
7 × the cost of an employee × a week	7 times the cost of employing one person for one week
1/10 × the cost of an employee × the population of California × the time taken for a football game	one tenth the cost of an employee during a football game by the population of California
1 × coffee consumption × a minute × population of the world	the amount of coffee consumed in one minute on the world
6 × weight of a person × population of California	six times the weight of the people who is worth

Table 6: Examples of perspectives generated by the sequence-to-sequence RNN. The model is able to capture rephrasings of fact descriptions and reordering of the facts. However, it often confuses prepositions and, very rarely, can produce nonsensical utterances.

LR+RNN perspective rated useful?	BASELINE	#
Yes	Yes	31
Yes	No	63
No	Yes	61
No	No	56

(a) A summary of the number of times the perspective generated by LR+RNN or BASELINE was rated useful by a majority of crowdworkers.

Cause of error	#
Proximity	9
Familiarity	6
Compatibility	8
Similarity	49
Generation	24
Other	14
Invalid mention	7
Total	117

(b) An analysis of errors produced by LR+RNN when its perspectives were not rated useful. Errors caused by poor formula selection are further categorized by selection criteria violated.

Cat.	Mention
	LR+RNN perspective (*vs.* BASELINE)
Prox.	... ready to ship about **2,300 miles** across the Pacific to the mainland ...
	three times the distance from San Francisco to Los Angeles (*vs.* the distance from San Francisco to Dallas TX).
Sim.	China had disposed of about **100,000 tons** of CFCs" ...
	one fifth of the weight of garbage produced in the United States by the population of Texas in one week. (*vs.* the average food wasted every year).
Fam.	... the project could save New England ratepayers **$4.6 billion** in energy costs over 25 years.
	one eighth the cost of employing the population of Asia for one hour. (*vs.* the construction cost of The Cosmopolitan in Las Vegas.)
Comp.	Hominids started shaping stone tools about **2.6 million years** ago.
	5 times the total time taken to build the number of cars registered. (*vs.* 17000 times the average lifetime for a tree).

(c) Examples of errors categorized by the criteria defined in Section 4.

Table 7: Results of an end-to-end human evaluation of the output produced by our perspective generation system (LR+RNN) and a baseline (BASELINE) that picks the numerically closest tuple in the knowledge base for each mention.

	Mention	Perspective (that's about...)
+	In 2007, Turkmenistan exported **50 billion cubic meters** of gas to Russia.	the amount of oil produced by the US during a lifetime
+	It can carry up to 10 nuclear warheads and has a range of **8,000 km**.	the distance from San Francisco to Beijing
-	the **2.7 million square feet** that Mission Bay's largest developer is entitled to build	twice the area of forest logged in a minute
-	Las Vegas Sands claims the **10.5 million square feet** is the largest building in Asia.	one half of an area of an average farm

Table 8: Examples of perspectives generated by our system that frame the mentioned quantity to be larger or smaller (top to bottom) than initially the authors thought.

formula exists within the knowledge base (within an order of magnitude of the mentioned value): a larger knowledge base would significantly decrease these errors.

7 Related work and discussion

We have proposed a new task of perspective generation. Compositionality is the key ingredient of our approach, which allows us synthesize information across multiple sources of information. At the same time, compositionality also poses problems for both formula selection and description generation.

On the formula selection side, we must compose facts that make sense. For semantic compatibility between the mention and description, we have relied on simple word vectors (Mikolov et al., 2013), but more sophisticated forms of semantic relations on larger units of text might yield better results (Bowman et al., 2015).

On the description generation side, there is a long line of work in generating natural language descriptions of structured data or logical forms Wong and Mooney (2007); Chen and Mooney (2008); Lu and Ng (2012); Angeli et al. (2010). We lean on the recent developments of neural sequence-to-sequence models (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015). Our problem bears some similarity to the semantic parsing work of Wang et al. (2015), who connect generated canonical utterances (representing logical forms) to real utterances.

If we return to our initial goal of helping people understand numbers, there are two important directions to explore. First, we have used a small knowledge base, which limits the coverage of perspectives we can generate. Using Freebase (Bollacker et al., 2008) or even open information extraction (Fader et al., 2011) would dramatically increase the number of facts and therefore the scope of possible perspectives.

Second, while we have focused mostly on basic compatibility, it would be interesting to explore more deeply how the juxtaposition of facts affects framing. Table 8 presents several examples generated by our system that frame the mentioned quantities to be larger or smaller than the authors originally thought. We think perspective generation is an exciting setting to study aspects of numeric framing (Teigen, 2015).

Reproducibility All code, data, and experiments for this paper are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x243284b4d81d4590b46030cdd3b72633/`.

Acknowledgments

We would like to thank Glen Chiacchieri for providing us information about the Dictionary of Numbers, Maneesh Agarwala for useful discussions and references, Robin Jia for sharing code for the sequence-to-sequence RNN, and the anonymous reviewers for their constructive feedback. This work was partially supported by the Sloan Research fellowship to the second author.

References

G. Angeli, P. Liang, and D. Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learn-

ing to align and translate. *arXiv preprint arXiv:1409.0473* .

P. J. Barrio, D. G. Goldstein, and J. M. Hofman. 2016. Improving the comprehension of numbers in the news. In *Conference on Human Factors in Computing Systems (CHI)*.

T. Berg-Kirkpatrick, D. Burkett, and D. Klein. 2012. An empirical investigation of statistical significance in NLP. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 995–1005.

K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *International Conference on Management of Data (SIGMOD)*. pages 1247–1250.

S. Bowman, G. Angeli, C. Potts, and C. D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Empirical Methods in Natural Language Processing (EMNLP)*.

D. L. Chen and R. J. Mooney. 2008. Learning to sportscast: A test of grounded language acquisition. In *International Conference on Machine Learning (ICML)*. pages 128–135.

F. Chevalier, R. Vuillemot, and G. Gali. 2013. Using concrete scales: A practical framework for effective visual depiction of complex measures. *IEEE Transactions on Visualization and Computer Graphics* 19:2426–2435.

G. Chiacchieri. 2013. Dictionary of numbers. `http://www.dictionaryofnumbers.com/`.

A. Fader, S. Soderland, and O. Etzioni. 2011. Identifying relations for open information extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

R. Jia and P. Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*.

M. G. Jones and A. R. Taylor. 2009. Developing a sense of scale: Looking backward. *Journal of Research in Science Teaching* 46:460–475.

Y. Kim, J. Hullman, and M. Agarwala. 2016. Generating personalized spatial analogies for distances and areas. In *Conference on Human Factors in Computing Systems (CHI)*.

W. Lu and H. T. Ng. 2012. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1611–1622.

M. Luong, H. Pham, and C. D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1412–1421.

T. Mikolov, K. Chen, G. Corrado, and Jeffrey. 2013. Efficient estimation of word representations in vector space. *arXiv* .

J. A. Paulos. 1988. *Innumeracy: Mathematical illiteracy and its consequences*. Macmillan.

C. Seife. 2010. *Proofiness: How you're being fooled by the numbers*. Penguin.

I. Sutskever, O. Vinyals, and Q. V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. pages 3104–3112.

K. H. Teigen. 2015. Framing of numeric quantities. *The Wiley Blackwell Handbook of Judgment and Decision Making* pages 568–589.

T. R. Tretter, M. G. Jones, and J. Minogue. 2006. Accuracy of scale conceptions in science: Mental maneuverings across many orders of spatial magnitude. *Journal of Research in Science Teaching* 43:1061–1085.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

Y. W. Wong and R. J. Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*. pages 172–179.

Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus

Iulian Vlad Serban[*°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Alberto García-Durán[*◇]
Université de Technologie de Compiègne CNRS
Rue du Dr Schweitzer,
Compigne, France

Caglar Gulcehre[°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Sungjin Ahn[°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Sarath Chandar[°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Aaron Courville[°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Yoshua Bengio[†°]
University of Montreal
2920 chemin de la Tour,
Montréal, QC, Canada

Abstract

Over the past decade, large-scale supervised learning corpora have enabled machine learning researchers to make substantial advances. However, to this date, there are no large-scale question-answer corpora available. In this paper we present the 30M Factoid Question-Answer Corpus, an enormous question-answer pair corpus produced by applying a novel neural network architecture on the knowledge base Freebase to transduce facts into natural language questions. The produced question-answer pairs are evaluated both by human evaluators and using automatic evaluation metrics, including well-established machine translation and sentence similarity metrics. Across all evaluation criteria the question-generation model outperforms the competing template-based baseline. Furthermore, when presented to human evaluators, the generated questions appear to be comparable in quality to real human-generated questions.

* First authors.
° Email: {iulian.vlad.serban,caglar.gulcehre, sungjin.ahn,sarath.chandar.anbil.parthipan, aaron.courville,yoshua.bengio}@umontreal.ca
◇ Email: alberto.garcia-duran@utc.fr
† CIFAR Senior Fellow

1 Introduction

A major obstacle for training question-answering (QA) systems has been due to the lack of labeled data. The question answering field has focused on building QA systems based on traditional information retrieval procedures (Lopez et al., 2011; Dumais et al., 2002; Voorhees and Tice, 2000). More recently, researchers have started to utilize large-scale knowledge bases (KBs) (Lopez et al., 2011), such as Freebase (Bollacker et al., 2008), WikiData (Vrandečić and Krötzsch, 2014) and Cyc (Lenat and Guha, 1989).[1] Bootstrapping QA systems with such structured knowledge is clearly beneficial, but it is unlikely alone to overcome the lack of labeled data. To take into account the rich and complex nature of human language, such as paraphrases and ambiguity, it would appear that labeled question and answer pairs are necessary. The need for such labeled pairs is even more critical for training neural network-based QA systems, where researchers until now have relied mainly on hand-crafted rules and heuristics to synthesize artificial QA corpora (Bordes et al., 2014; Bordes et al., 2015).

Motivated by these recent developments, in this paper we focus on generating questions based on the Freebase KB. We frame question generation as a transduction problem starting from a Freebase fact, represented by a triple consisting of a subject, a relationship and an object, which is trans-

[1] Freebase is now a part of WikiData.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 588–598,
Berlin, Germany, August 7-12, 2016.

duced into a question about the subject, where the object is the correct answer (Bordes et al., 2015). We propose several models, largely inspired by recent neural machine translation models (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015), and we use an approach similar to Luong et al. (2015) for dealing with the problem of rare-words. We evaluate the produced questions in a human-based experiment as well as with respect to automatic evaluation metrics, including the well-established machine translation metrics BLEU and METEOR and a sentence similarity metric. We find that the question-generation model outperforms the competing template-based baseline, and, when presented to untrained human evaluators, the produced questions appear to be indistinguishable from real human-generated questions. This suggests that the produced question-answer pairs are of high quality and therefore that they will be useful for training QA systems. Finally, we use the best performing model to construct a new factoid question-answer corpus – The 30M Factoid Question-Answer Corpus – which is made freely available to the research community.[2]

2 Related Work

Question generation has attracted interest in recent years with notable work by Rus et al. (2010), followed by the increasing interest from the Natural Language Generation (NLG) community. A simple rule-based approach was proposed in different studies as *wh-fronting* or *wh-inversion* (Kalady et al., 2010; Ali et al., 2010). This comes at the disadvantage of not making use of the semantic content of words apart from their syntactic role. The problem of determining the *question type* (e.g. that a *Where-question* should be triggered for locations), which requires knowledge of the category type of the elements involved in the sentence, has been addressed in two different ways: by using named entity recognizers (Mannem et al., 2010; Yao and Zhang, 2010) or semantic role labelers (Chen et al., 2009). In Curto et al. (2012) questions are split into classes according to their syntactic structure, prefix of the question and the category of the answer, and then a pattern is learned to generate questions for that class of questions. After the identification of key points, Chen et al. (2009) apply handcrafted-templates to generate questions framed in the right target expression by following the analysis of Graesser et al. (1992), who classify questions according to a taxonomy consisting of 18 categories.

The works discussed so far propose ways to map unstructured text to questions. This implies a two-step process: first, transform a text into a symbolic representation (e.g. a syntactic representation of the sentence), and second, transform the symbolic representation of the text into the question (Yao et al., 2012). On the other hand, going from a symbolic representation (structured information) to a question, as we will describe in the next section, only involves the second step. Closer to our approach is the work by Olney et al. (2012). They take triples as input, where the edge relation defines the question template and the head of the triple replaces the placeholder token in the selected question template. In the same spirit, Duma et al. (2013) generate short descriptions from triples by using templates defined by the relationship and replacing accordingly the placeholder tokens for the subject and object.

Our baseline is similar to that of Olney et al. (2012), where a set of relationship-specific templates are defined. These templates include placeholders to replace the string of the subject. The main difference with respect to their work is that our baseline does not explicitly define these templates. Instead, each relationship has as many templates as there are different ways of framing a question with that relationship in the training set. This yields more diverse and semantically richer questions by effectively taking advantage of the fact-question pairs, which Olney et al. did not have access to in their experiments.

Unlike the work by Berant and Liang (2014), which addresses the problem of deterministically generating a set of candidate logical forms with a canonical realization in natural language for each, our work addresses the inverse problem: given a logical form (fact) it outputs the associated question.

It should also be noted that recent work in question answering have used simpler rule-based and template-based approaches to generate synthetic questions to address the lack of question-answer pairs to train their models (Bordes et al., 2014; Bordes et al., 2015).

[2] www.agarciaduran.org

3 Task Definition

3.1 Knowledge Bases

In general, a KB can be viewed as a multi-relational graph, which consists of a set of nodes (entities) and a set of edges (relationships) linking nodes together. In Freebase (Bollacker et al., 2008) these relationships are directed and always connect exactly two entities. For example, in Freebase the two entities *fires_creek* and *nantahala_national_forest* are linked together by the relationship *contained_by*. Since the triple {*fires_creek*, *contained_by*, *nantahala_national_forest*} represents a complete and self-contained piece of information, it is also called a *fact* where *fires_creek* is the subject (head of the edge), *contained_by* is the relationship and *nantahala_national_forest* is the object (tail of the edge).

3.2 Transducing Facts to Questions

We aim to transduce a fact into a question, such that:

1. The question is concerned with the subject and relationship of the fact, and
2. The object of the fact represents a valid answer to the generated question.

We model this in a probabilistic framework as a directed graphical model:

$$P(Q|F) = \prod_{n=1}^{N} P(w_n|w_{<n}, F), \qquad (1)$$

where $F = (subject, relationship, object)$ represents the fact, $Q = (w_1, \ldots, w_N)$ represents the question as a sequence of tokens $w_1, \ldots, w_N$, and $w_{<n}$ represents all the tokens generated before token w_n. In particular, w_N represents the question mark symbol '?'.

3.3 Dataset

We use the SimpleQuestions dataset (Bordes et al., 2015) in order to train our models. This is by far the largest dataset of question-answer pairs created by humans based on a KB. It contains over 100K question-answer pairs created by users on Amazon Mechanical Turk[3] in English based on the Freebase KB. In order to create the questions, human participants were shown one whole Freebase fact at a time and they were asked to phrase a question such that the object of the presented fact becomes the answer of the question.[4] Consequently, both the subject and the relationship are explicitly given in each question. But indirectly characteristics of the object may also be given since the humans have an access to it as well. Often when phrasing a question the annotators tend to be more informative about the target object by giving specific information about it in the question produced. For example, in the question *What city is the American actress X from?* the city name given in the object informs the human participant that it was in America - information, which was not provided by either the subject or relationship of the fact. We have also observed that the questions are often ambiguous: that is, one can easily come up with several possible answers that may fit the specifications of the question. Table 1 shows statistics of the dataset.

Questions	Entities	Relationships	Words
108,442	131,684	1,837	∼77k

Table 1: Statistics of SimpleQuestions

4 Model

We propose to attack the problem with the models inspired by the recent success of neural machine translation models (Sutskever et al., 2014; Bahdanau et al., 2015). Intuitively, one can think of the transduction task as a "lossy translation" from structured knowledge (facts) to human language (questions in natural language), where certain aspects of the structured knowledge is intentionally left out (e.g. the name of the object). These models typically consist of two components: an encoder, which encodes the source phrase into one or several fixed-size vectors, and a decoder, which decodes the target phrase based on the results of the encoder.

4.1 Encoder

In contrast to the neural machine translation framework, our source language is not a proper language but instead a sequence of three variables making up a fact. We propose an encoder sub-model, which encodes each atom of the fact into an embedding. Each atom $\{s, r, o\}$, may

[3] www.mturk.com

[4] It is not necessary for the object to be the only answer, but it is required to be one of the possible answers.

stand for subject, relationship and object, respectively, of a fact $F = (s, r, o)$ is represented as a 1-of-K vector x_{atom}, whose embedding is obtained as $e_{\text{atom}} = E_{\text{in}} x_{\text{atom}}$, where $E_{\text{in}} \in \mathbb{R}^{D_{\text{Enc}} \times K}$ is the embedding matrix of the input vocabulary and K is the size of that vocabulary. The encoder transforms this embedding into $\text{Enc}(F)_{\text{atom}} \in \mathbb{R}^{H_{\text{Dec}}}$ as $\text{Enc}(F)_{\text{atom}} = W_{\text{Enc}} e_{\text{atom}}$, where $W_{\text{Enc}} \in \mathbb{R}^{H_{\text{Dec}} \times D_{\text{Enc}}}$.

This embedding matrix, E_{in}, could be another parameter of the model to be learned, however, as discussed later (see Section 4.3), we have learned it separately and beforehand with *TransE* (Bordes et al., 2013), a model aimed at modeling this kind of multi-relational data. We fix it and do not allow the encoder to tune it during training.

We call *fact embedding* $\text{Enc}(F) \in \mathbb{R}^{3H_{\text{Dec}}}$ the concatenation $[\text{Enc}(F)_s, \text{Enc}(F)_r, \text{Enc}(F)_o]$ of the atom embeddings, which is the input for the next module.

4.2 Decoder

For the decoder, we use a GRU recurrent neural network (RNN) (Cho et al., 2014) with an attention-mechanism (Bahdanau et al., 2015) on the encoder representation to generate the associated question Q to that fact F. Recently, it has been shown that the GRU RNN performs equally well across a range of tasks compared to other RNN architectures, such as the LSTM RNN (Greff et al., 2015). The hidden state of the decoder RNN is computed at each time step n as:

$$g_n^r = \sigma(W_r E_{out} w_{n-1} + C_r c(F, h_{n-1}) + U_r h_{n-1}) \quad (2)$$

$$g_n^u = \sigma(W_u E_{out} w_{n-1} + C_u c(F, h_{n-1}) + U_u h_{n-1}) \quad (3)$$

$$\tilde{h} = \tanh(W E_{out} w_{n-1} + C c(F, h_{n-1}) + U(g_n^r \circ h_{n-1})) \quad (4)$$

$$h_n = g_n^u \circ h_{n-1} + (1 - g_n^u) \circ \tilde{h}, \quad (5)$$

where σ is the sigmoid function, s.t. $\sigma(x) \in [0, 1]$, and the circle, $\circ$, represents element-wise multiplication. The initial state h_0 of this RNN is given by the output of a feedforward neural network fed with the fact embedding. The product $E_{out} w_n \in \mathbb{R}^{D_{\text{Dec}}}$ is the decoder embedding vector corresponding to the word w_n (coded as a 1-of-V vector, with V being the size of the output vocabulary), the variables $U_r, U_u, U, C_r, C_u, C \in \mathbb{R}^{H_{\text{Dec}} \times H_{\text{Dec}}}$, $W_r, W_u, W \in \mathbb{R}^{H_{\text{Dec}} \times D_{\text{Dec}}}$ are the parameters of the GRU and $c(F, h_{n-1})$ is the context vector (defined below Eq. 6). The vector g^r is called the *reset gate*, g^u as the *update gate* and $\tilde{h}$ the *candidate activation*. By adjusting g^r and g^u appropriately, the model is able to create linear *skip-connections* between distant hidden states, which in turn makes the credit assignment problem easier and the gradient signal stronger to earlier hidden states. Then, at each time step n the set of probabilities over word tokens is given by applying a softmax layer over $V_o \tanh(V_h h_n + V_w E_{out} w_{n-1} + V_c c(F, h_{n-1}))$, where $V_o \in \mathbb{R}^{V \times H_{\text{Dec}}}$, $V_h, V_c \in \mathbb{R}^{H_{\text{Dec}} \times H_{\text{Dec}}}$ and $V_w \in \mathbb{R}^{H_{\text{Dec}} \times D_{\text{Dec}}}$. Lastly, the function $c(F, h_{n-1})$ is computed using an attention-mechanism:

$$c(F, h_{n-1}) = \alpha_{\text{s},n-1} \text{Enc}(F)_{\text{s}} + \alpha_{\text{r},n-1} \text{Enc}(F)_{\text{r}} + \alpha_{\text{o},n-1} \text{Enc}(F)_{\text{o}}, \quad (6)$$

where $\alpha_{\text{s},n-1}, \alpha_{\text{r},n-1}, \alpha_{\text{r},n-1}$ are real-valued scalars, which weigh the contribution of the subject, relationship and object representations.

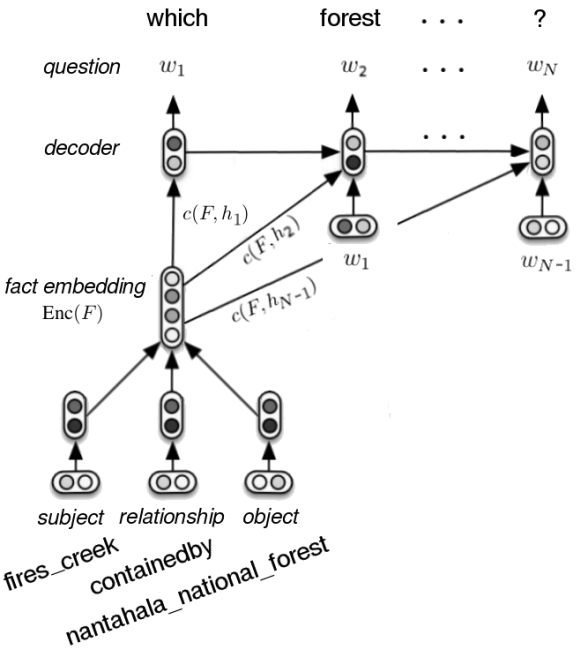


Figure 1: The computational graph of the question-generation model, where $\text{Enc}(F)$ is the fact embedding produced by the encoder model, and $c(F, h_{n-1})$ for $n = 1, \ldots, N$ is the fact representation weighed according to the attention-mechanism, which depends on both the fact F and the previous hidden state of the decoder RNN h_{n-1} . For the sake of simplicity, the attention-mechanism is not shown explicitly.

They correspond to the *attention* of the model, and are computed by applying a one-layer neural network with tanh-activation function on the encoder representations of the fact, Enc(F), and the previous hidden state of the RNN, h_{n-1}, followed by the sigmoid function to restrict the attention values to be between zero and one. The need for the attention-mechanism is motivated by the intuition that the model needs to attend to the subject only once during the generation process while attending to the relationship at all other times during the generation process. The model is illustrated in Figure 1.

4.3 Modeling the Source Language

A particular problem with the model presented above is related to the embeddings for the entities, relationships and tokens, which all have to be learned in one way or another. If we learn these naively on the SimpleQuestions training set, the model will perform poorly when it encounters previously unseen entities, relationships or tokens. Furthermore, the multi-relational graph defined by the facts in SimpleQuestions is extremely sparse, i.e. each node has very few edges to other nodes, as can be expected due to high ratio of unique entities over number of examples. Therefore, even for many of the entities in SimpleQuestions, the model may perform poorly if the embedding is learned solely based on the SimpleQuestions dataset alone.

On the source side, we can resolve this issue by initializing the subject, relationship and object embeddings to those learned by applying multi-relational embedding-based models to the knowledge base. Multi-relational embedding-based models (Bordes et al., 2011) have recently become popular to learn distributed vector embeddings for knowledge bases, and have shown to scale well and yield good performance. Due to its simplicity and good performance, we choose to use TransE (Bordes et al., 2013) to learn such embeddings. TransE is a translation-based model, whose energy function is trained to output low values when the fact expresses true information, i.e. a fact which exists in the knowledge base, and otherwise high values. Formally, the energy function is defined as $f(s, r, o) = ||e_s + e_r - e_o||_2$, where e_s, e_r and e_o are the real-valued embedding vectors for the subject, relationship and object of a fact. Further details are given by Bordes et al. (2013).

Embeddings for entities with few connections are easy to learn, yet the quality of these embeddings depends on how inter-connected they are. In the extreme case where the subject and object of a triple only appears once in the dataset, the learned embeddings of the subject and object will be semantically meaningless. This happens very often in SimpleQuestions, since only around 5% of the entities have more than 2 connections in the graph. Thus, by applying TransE directly over this set of triples, we would eventually end up with a layout of entities that does not contain clusters of semantically close concepts. In order to guarantee an effective semantic representation of the embeddings, we have to learn them together with additional triples extracted from the whole Freebase graph to complement the SimpleQuestions graph with relevant information for this task.

We need a coarse representation for the entities contained in SimpleQuestions, capturing the *basic* information, like the profession or nationality, the annotators tend to use when phrasing the questions, and accordingly we have ensured the embeddings contain this information by taking triples coming from the Freebase graph[5] regarding:

1. Category information: given by the *type/instance* relationship, this ensures that all the entities of the same semantic category are close to each other. Although one might think that the expected category of the subject/object could be inferred directly from the relationship, there are fine-grained differences in the expected types that be extracted only directly by observing this category information.

2. Geographical information: sometimes the annotators have included information about nationality (e.g. *Which French president...?*) or location (e.g. *Where in Germany...?*) of the subject and/or object. This information is given by the relationships *person/nationality* and *location/contained_by*. By including these facts in the learning, we ensure the existence of a fine-grained layout of the embeddings regarding this information within a same category.

[5]Extracted from one of the latest Freebase dumps (downloaded in mid-August 2015) `https://developers.google.com/freebase/data`

Closest neighbors to	Warner Bros. Entertainment	Manchester	hindi language
SQ	Billy Gibbons	Ricky Anane	nepali indian
	Jenny Lewis	Lee Dixon	Naseeb
	Lies of Love	Jerri Bryne	Ghar Ek Mandir
	Swordfish	Greg Wood	standard chinese
SQ + FB	Paramount Pictures	Oxford	dutch language
	Sony Pictures Entertainment	Sale	italian language
	Electronic Arts	Liverpool	danish language
	CBS	Guildford	bengali language

Table 2: Examples of differences in the local structure of the vector space embeddings when adding more FB facts

3. Gender: similarly, sometimes annotators have included information about gender (e.g. *Which male audio engineer...*?). This information is given by the relationship *person/gender*.

To this end, we have included more than $300,000$ facts from Freebase in addition to the facts in SimpleQuestions for training. Table 2 shows the differences in the embeddings before and after adding additional facts for training the TransE representations.

4.4 Generating Questions

To resolve the problem of data sparsity and previously unseen words on the target side, we draw inspiration from the placeholders proposed for handling rare words in neural machine translation by Luong et al. (2015). For every question and answer pair, we search for words in the question which overlap with words in the subject string of the fact.[6] We heuristically estimate the sequence of most likely words in the question, which correspond to the subject string. These words are then replaced by the placeholder token *<placeholder>*. For example, given the fact {fires_creek, contained_by, nantahala_national_forest} the original question *Which forest is Fires Creek in?* is transformed into the question *Which forest is <placeholder>in?*. The model is trained on these modified questions, which means that model only has to learn decoder embeddings for tokens which are not in the subject string. At test time, after outputting a question, all placeholder tokens are replaced by the subject string and then the outputs are evaluated. We call this the Single-Placeholder (SP) model. The main difference with respect to that of Luong et al. (2015) is that we do not use placeholder tokens in the input language, because then the entities and relationships in the input would not be able to transmit semantic (e.g. topical) information to the decoder. If we had included placeholder tokens in the input language, the model would not be able to generate informative words regarding the subject in the question (e.g. it would be impossible for the model to learn that the subject *Paris* may be accompanied by the words *French city* when generating a question, because it would not see *Paris* but only a placeholder token).

A single placeholder token for all question types could unnecessarily limit the model. We therefore also experiment with another model, called the Multi-Placeholder (MP) model, which uses 60 different placeholder tokens such that the placeholder for a given question is chosen based on the subject category extracted from the relationship (e.g. contained_by is classified in the category location, and so the transformed question would be *Which forest is <location placeholder> in?*). This could make it easier for the model to learn to phrase questions about a diverse set of entities, but it also introduces additional parameters, since there are now 60 placeholder embeddings to be learned, and therefore the model may suffer from overfitting. This way of addressing the sparsity in the output reduces the vocabulary size to less than 7000 words.

4.5 Template-based Baseline

To compare our neural network models, we propose a (non-parametric) template-based baseline model, which makes use of the entire training set when generating a question. The baseline operates on questions modified with the placeholder as in the preceding section. Given a fact F as input, the baseline picks a candidate fact F_c in the training set at uniformly random, where F_c has the same relationship as F. Then the baseline considers the questions corresponding to F_c and as in the

[6] We use the tool difflib: `https://docs.python.org/2/library/difflib.html`.

SP model, in the final step the placeholder token in the question is replaced by the subject string of the fact F.

5 Experiments

5.1 Training Procedure

All neural network models were implemented in Theano (Theano Development Team, 2016). To train the neural network models, we optimized the log-likelihood using the first-order gradient-based optimization algorithm Adam (Kingma and Ba, 2015). To decide when to stop training we used early stopping with patience (Bengio, 2012) on the METEOR score obtained for the validation set. In all experiments, we use the default split of the SimpleQuestions dataset into training, validation and test sets.

We trained TransE embeddings with embedding dimensionality 200 for each subject, relationship and object. Based on preliminary experiments, for all neural network models we fixed the learning rate to 0.00025 and clipped parameter gradients with norms larger than 0.1. We further fixed the embedding dimensionality of words to be 200, and the hidden state of the decoder RNN to have dimensionality 600.

5.2 Evaluation

To investigate the performance of our models, we make use of both automatic evaluation metrics and human evaluators.

5.2.1 Automatic Evaluation Metrics

BLEU (Papineni et al., 2002) and METEOR (Banerjee and Lavie, 2005) are two widely used evaluation metrics in statistical machine translation and automatic image-caption generation (Chen et al., 2015). Similar to statistical machine translation, where a phrase in the source language is mapped to a phrase in the target language, in this task a KB fact is mapped to a natural language question. Both tasks are highly constrained, e.g. the set of valid outputs is limited. This is true in particular for short phrases, such as one sentence questions. Furthermore, in both tasks, the majority of valid outputs are paraphrases of each other, which BLEU and METEOR have been designed to capture. We therefore believe that BLEU and METEOR constitute reasonable performance metrics for evaluating the generated questions.

Although we believe that METEOR and BLEU are reasonable evaluation metrics, they may have not recognize certain paraphrases, in particular paraphrases of entities. We therefore also make use of a sentence similarity metric, as proposed by Rus and Lintean (2012), which we will denote *Embedding Greedy* (Emb. Greedy). The metric makes use of a word similarity score, which in our experiments is the cosine similarity between two Word2Vec word embeddings (Mikolov et al., 2013).[7] The metric finds a (non-exclusive) alignment between words in the two questions, which maximizes the similarity between aligned words, and computes the sentence similarity as the mean over the word similarities between aligned words.

The results are shown in Table 3. Example questions produced by the model with multiple placeholders are shown in Table 4. The neural network models outperform the template-based baseline by a clear margin across all metrics. The template-based baseline is already a relatively strong model, because it makes use of a separate template for each relationship. Qualitatively the neural networks outperform the baseline model in cases where they are able to levage additional knowledge about the entities (see first, third and fifth example in Table 4). On the other hand, for rare relationships the baseline model appears to perform better, because it is able to produce a reasonable question if only a single example with the same relationship exists in the training set (see eighth example in Table 4). Given enough training data this suggests that neural networks are generally better at the question generation task compared to hand-crafted template-based procedures, and therefore that they may be useful for generating question answering corpora. Furthermore, it appears that the best performing models are the models where TransE are trained on the largest set of triples (TransE++). This set contains, apart from the supporting triples described in Section 4.3, triples involving entities which are highly connected to the entities found in the SimpleQuestions facts. In total, around 30 millions of facts, which have been used to generate the 30M Factoid Question-Answer Corpus. Lastly, it is not clear whether the model with a single placeholder or the model with multiple placeholders performs best. This motivates the following human study.

[7] We use the Word2Vec embeddings pretrained on the Google News Corpus: `https://code.google.com/p/word2vec/`.

Model	BLEU	METEOR	Emb. Greedy
Baseline	31.36	33.12	74.02
SP Triples	33.27	35.07	76.72
MP Triples	32.76	34.97	76.70
SP Triples TransE++	**33.32**	**35.38**	76.78
MP Triples TransE++	33.28	35.29	**77.01**

Table 3: Test performance for all models w.r.t. BLEU, METEOR and Emb. Greedy performance metrics, where *SP* indicates models with a single placeholder and *MP* models with multiple placeholders. *TransE++* indicates models where the TransE embeddings have been pretrained on a larger set of triples. The best performance on each metric is marked in bold font.

Fact	Human	Baseline	MP Triples TransE++
bayuvi dupki – contained by – europe	where is bayuvi dupki?	what state is the city of bayuvi dupki located in?	what continent is bayuvi dupki in?
illinois – contains – ludlow township	what is in illinois?	what is a tributary found in illinois?	what is the name of a place within illinois?
neo contra – publisher – konami	who published neo contra?	which company published the game neo contra?	who is the publisher for the computer videogame neo contra?
fumihiko maki – structures designed – makuhari messe	fumihiko maki designed what structure?	what park did fumihiko maki help design?	what's a structure designed by fumihiko maki?
cheryl hickey – profession – actor	what is cheryl hickey's profession?	what is cheryl hickey?	what is cheryl hickey's profession in the entertainment industry?
cherry – drugs with this flavor – tussin expectorant for adults 100 syrup	name a cherry flavored drug?	what is a cherry flavored drug?	what's a drug that cherry shaped like?
pop music – artists – nikki flores	what artist is known for pop music?	An example of pop music is what artist?	who's an american singer that plays pop music?

Table 4: Test examples and corresponding questions.

5.2.2 Human Evaluation Study

We carry out pairwise preference experiments on Amazon Mechanical Turk.

Initially, we considered carrying out separate experiments for measuring relevancy and fluency respectively, since this is common practice in machine translation. However, the relevancy of a question is determined solely by a single factor, i.e. the relationship, since by construction the subject is always in the question. Measuring relevancy is therefore not very useful in our task. To verify this we carried out an internal pairwise preference experiment with human subjects, who were repeatedly shown a fact and two questions and asked to select the most relevant question. We found that 93% of the questions generated by the MP Triples TransE++ model were either judged better or at least as good as the human generated questions w.r.t. relevancy. The remaining 7% questions of the MP Triples TransE++ model questions were also judged relevant questions, although less so compared to the human generated questions. In the next experiment, we therefore measure the holistic quality of the questions.

We setup experiments comparing: Human-Baseline (human and baseline questions), Human-MP (human and MP Triples TransE++ questions) and Baseline-MP (baseline and MP Triples TransE++ questions). We show human evaluators a fact along with two questions, one question from each model for the corresponding fact, and ask the them to choose the question which is most relevant to the fact and most natural. The human evaluator also has the option of not choosing either question. This is important if both questions are equally good or if neither of the questions make sense. At the beginning of each experiment, we show the human evaluators two examples of statements and a corresponding pair of questions, where we briefly explain the form of the statements and how questions relate to those statements. Following the introductory examples, we present the facts and cor-

Model A	Model B	Model A Preference (%)	Model B Preference (%)	Fleiss' kappa
Human	Baseline	$^{*}\mathbf{56.329 \pm 5.469}$	34.177 ± 5.230	0.242
Baseline	MP Triples TransE++	32.484 ± 5.180	$^{*}\mathbf{60.828 \pm 5.399}$	0.234
Human	MP Triples TransE++	38.652 ± 5.684	$\mathbf{51.418 \pm 5.833}$	0.182

Table 5: Pairwise human evaluation preferences computed across evaluators with 95% confidence intervals. The preferred model in each experiment is marked in bold font. An asterisk next to the preferred model indicates a statistically significance likelihood-ratio test, which shows that the model is preferred in at least half of the presented examples with 95% confidence. The name *MP Triples TransE++* indicates the model with multiple placeholders and TransE embeddings pretrained on a larger set of triples. The last column shows the Fleiss' kappa averaged across batches (HITs) with different evaluators and questions.

responding pair of questions one by one. To avoid presentation bias, we randomly shuffle the order of the examples and the order in which questions are shown by each model. During each experiment, we also show four check facts and corresponding check questions at random, which any attentive human annotator should be able to answer easily. We discard responses of human evaluators who fail any of these four checks.

The preference of each example is defined as the question which is preferred by the majority of the evaluators. Examples where neither of the two questions are preferred by the majority of the evaluators, i.e. when there is an equal number of evaluators who prefer each question, are assigned to a separate preference class called "comparable".[8]

The results are shown in Table 5. In total, 3,810 preferences were recorded by 63 independent human evaluators. The questions produced by each model model pair were evaluated in 5 batches (HITs). Each human evaluated 44-75 examples (facts and corresponding question pairs) in each batch and each example was evaluated by 3-5 evaluators. In agreement with the automatic evaluation metrics, the human evaluators strongly prefer either the human or the neural network model over the template-based baseline. Furthermore, it appears that humans cannot distinguish between the human-generated questions and the neural network questions, on average showing a preference towards the later over the former ones. We hypothesize this is because our model penalizes uncommon and unnatural ways to frame questionsand sometimes, includes specific information about the target object that the humans do not (see last example in Table 4). This confirms our earlier assertion, that the neural network questions can be used for building question answering systems.

[8]The probabilities for the "comparable" class in Table 5 can be computed in each row as 100 minus the third and fourth column in the table.

6 Conclusion

We propose new neural network models for mapping knowledge base facts into corresponding natural language questions. The neural networks combine ideas from recent neural network architectures for statistical machine translation, as well as multi-relational knowledge base embeddings for overcoming sparsity issues and placeholder techniques for handling rare words. The produced question and answer pairs are evaluated using automatic evaluation metrics, including BLEU, METEOR and sentence similarity, and are found to outperform a template-based baseline model. When evaluated by untrained human subjects, the question and answer pairs produced by our best performing neural network appears to be comparable in quality to real human-generated questions. Finally, we use our best performing neural network model to generate a corpus of 30M question and answer pairs, which we hope will enable future researchers to improve their question answering systems.

Acknowledgments

The authors acknowledge IBM Research, NSERC, Canada Research Chairs and CIFAR for funding. The authors thank Yang Yu, Bing Xiang, Bowen Zhou and Gerald Tesauro for constructive feedback, and Antoine Bordes, Nicolas Usunier, Sumit Chopra and Jason Weston for providing the SimpleQuestions dataset. This research was enabled in part by support provided by Calcul Qubec (`www.calculquebec.ca`) and Compute Canada (`www.computecanada.ca`).

References

[Ali et al.2010] Husam Ali, Yllias Chali, and Sadid A Hasan. 2010. Automation of question generation from sentences. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 58–67.

[Bahdanau et al.2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

[Banerjee and Lavie2005] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *ACL, Workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

[Bengio2012] Yoshua Bengio. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade*, pages 437–478. Springer.

[Berant and Liang2014] Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*, pages 1415–1425.

[Bollacker et al.2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

[Bordes et al.2011] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *AAAI 2011*.

[Bordes et al.2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

[Bordes et al.2014] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases - European Conference, (ECML PKDD)*, pages 165–180.

[Bordes et al.2015] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

[Chen et al.2009] Wei Chen, Gregory Aist, and Jack Mostow. 2009. Generating questions automatically from informational text. In *Proceedings of the 2nd Workshop on Question Generation (AIED 2009)*, pages 17–24.

[Chen et al.2015] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft COCO captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

[Cho et al.2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

[Curto et al.2012] Sergio Curto, A Mendes, and Luisa Coheur. 2012. Question generation based on lexico-syntactic patterns learned from the web. *Dialogue and Discourse*, 3(2):147–175.

[Duma and Klein2013] Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. *ACL*, pages 83–94.

[Dumais et al.2002] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. 2002. Web question answering: Is more always better? In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–298.

[Graesser et al.1992] Arthur C Graesser, Sallie E Gordon, and Lawrence E Brainerd. 1992. QUEST: A model of question answering. *Computers and Mathematics with Applications*, 23(6):733–745.

[Greff et al.2015] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A search space odyssey. *arXiv preprint arXiv:1503.04069*.

[Kalady et al.2010] Saidalavi Kalady, Ajeesh Elikkottil, and Rajarshi Das. 2010. Natural language question generation using syntax and keywords. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 1–10. questiongeneration. org.

[Kingma and Ba2015] Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

[Lenat and Guha1989] Douglas B. Lenat and Ramanathan V. Guha. 1989. *Building large knowledge-based systems; representation and inference in the Cyc project*. Addison-Wesley Longman Publishing Co., Inc.

[Lopez et al.2011] Vanessa Lopez, Victoria Uren, Marta Sabou, and Enrico Motta. 2011. Is question answering fit for the semantic web? a survey. *Semantic Web*, 2(2):125–155.

[Luong et al.2015] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *ACL*, pages 11–19.

[Mannem et al.2010] Prashanth Mannem, Rashmi Prasad, and Aravind Joshi. 2010. Question generation from paragraphs at upenn: Qgstec system description. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 84–91.

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

[Olney et al.2012] Andrew M Olney, Arthur C Graesser, and Natalie K Person. 2012. Question generation from concept maps. *Dialogue and Discourse*, 3(2):75–99.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318.

[Rus and Lintean2012] Vasile Rus and Mihai Lintean. 2012. A comparison of greedy and optimal assessment of natural language student input using word-to-word similarity metrics. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP, NAACL*, pages 157–162.

[Rus et al.2010] Vasile Rus, Brendan Wyse, Paul Piwek, Mihai Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 251–257.

[Sutskever et al.2014] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*.

[Theano Development Team2016] Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

[Voorhees and Tice2000] Ellen M Voorhees and DM Tice. 2000. Overview of the trec-9 question answering track. In *TREC*.

[Vrandečić and Krötzsch2014] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

[Yao and Zhang2010] Xuchen Yao and Yi Zhang. 2010. Question generation with minimal recursion semantics. In *Proceedings of QG2010: The Third Workshop on Question Generation*, pages 68–75.

[Yao et al.2012] Xuchen Yao, Gosse Bouma, and Yi Zhang. 2012. Semantics-based question generation and implementation. *Dialogue and Discourse*, 3(2):11–42.

Latent Predictor Networks for Code Generation

Wang Ling♢ Edward Grefenstette♢ Karl Moritz Hermann♢
Tomáš Kočiský♢♣ Andrew Senior♢ Fumin Wang♢ Phil Blunsom♢♣

♢Google DeepMind ♣University of Oxford
{lingwang,etg,kmh,tkocisky,andrewsenior,awaw,pblunsom}@google.com

Abstract

Many language generation tasks require the production of text conditioned on both structured and unstructured inputs. We present a novel neural network architecture which generates an output sequence conditioned on an arbitrary number of input functions. Crucially, our approach allows both the choice of conditioning context and the granularity of generation, for example characters or tokens, to be marginalised, thus permitting scalable and effective training. Using this framework, we address the problem of generating programming code from a mixed natural language and structured specification. We create two new data sets for this paradigm derived from the collectible trading card games Magic the Gathering and Hearthstone. On these, and a third preexisting corpus, we demonstrate that marginalising multiple predictors allows our model to outperform strong benchmarks.

1 Introduction

The generation of both natural and formal languages often requires models conditioned on diverse predictors (Koehn et al., 2007; Wong and Mooney, 2006). Most models take the restrictive approach of employing a single predictor, such as a word softmax, to predict all tokens of the output sequence. To illustrate its limitation, suppose we wish to generate the answer to the question "Who wrote The Foundation?" as "The Foundation was written by Isaac Asimov". The generation of the words "Issac Asimov" and "The Foundation" from a word softmax trained on annotated data is unlikely to succeed as these words are sparse. A robust model might, for example, employ one predictor to copy "The Foundation" from the input, and a another one to find the answer "Issac Asimov" by searching through a database. However, training multiple predictors is in itself a challenging task, as no annotation exists regarding the predictor used to generate each output token. Furthermore, predictors generate segments of different granularity, as database queries can generate multiple tokens while a word softmax generates a single token. In this work we introduce *Latent Predictor Networks (LPNs)*, a novel neural architecture that fulfills these desiderata: at the core of the architecture is the exact computation of the marginal likelihood over latent predictors and generated segments allowing for scalable training.

Figure 1: Example MTG and HS cards.

We introduce a new corpus for the automatic generation of code for cards in Trading Card Games (TCGs), on which we validate our model [1]. TCGs, such as Magic the Gathering (MTG) and Hearthstone (HS), are games played between two players that build decks from an ever expanding pool of cards. Examples of such cards are shown in Figure 1. Each card is identified by its attributes

[1] Dataset available at https://deepmind.com/publications.html

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 599–609,
Berlin, Germany, August 7-12, 2016.

(e.g., name and cost) and has an effect that is described in a text box. Digital implementations of these games implement the game logic, which includes the card effects. This is attractive from a data extraction perspective as not only are the data annotations naturally generated, but we can also view the card as a specification communicated from a designer to a software engineer.

This dataset presents additional challenges to prior work in code generation (Wong and Mooney, 2006; Jones et al., 2012; Lei et al., 2013; Artzi et al., 2015; Quirk et al., 2015), including the handling of structured input—i.e. cards are composed by multiple sequences (e.g., name and description)—and attributes (e.g., attack and cost), and the length of the generated sequences. Thus, we propose an extension to attention-based neural models (Bahdanau et al., 2014) to attend over structured inputs. Finally, we propose a code compression method to reduce the size of the code without impacting the quality of the predictions.

Experiments performed on our new datasets, and a further pre-existing one, suggest that our extensions outperform strong benchmarks.

The paper is structured as follows: We first describe the data collection process (Section 2) and formally define our problem and our baseline method (Section 3). Then, we propose our extensions, namely, the structured attention mechanism (Section 4) and the LPN architecture (Section 5). We follow with the description of our code compression algorithm (Section 6). Our model is validated by comparing with multiple benchmarks (Section 7). Finally, we contextualize our findings with related work (Section 8) and present the conclusions of this work (Section 9).

2 Dataset Extraction

We obtain data from open source implementations of two different TCGs, MTG in Java[2] and HS in Python.[3] The statistics of the corpora are illustrated in Table 1. In both corpora, each card is implemented in a separate class file, which we strip of imports and comments. We categorize the content of each card into two different groups: *singular fields* that contain only one value; and *text fields*, which contain multiple words representing different units of meaning. In MTG, there are six singular fields (attack, defense, rarity, set, id, and health) and four text fields (cost, type, name, and description), whereas HS cards have eight singular fields (attack, health, cost and durability, rarity, type, race and class) and two text fields (name and description). Text fields are tokenized by splitting on whitespace and punctuation, with exceptions accounting for domain specific artifacts (e.g., Green mana is described as "{G}" in MTG). Empty fields are replaced with a "NIL" token.

	MTG	HS
Programming Language	Java	Python
Cards	13,297	665
Cards (Train)	11,969	533
Cards (Validation)	664	66
Cards (Test)	664	66
Singular Fields	6	4
Text Fields	8	2
Words In Description (Average)	21	7
Characters In Code (Average)	1,080	352

Table 1: Statistics of the two TCG datasets.

The code for the HS card in Figure 1 is shown in Figure 2. The effect of "drawing cards until the player has as many cards as the opponent" is implemented by computing the difference between the players' hands and invoking the draw method that number of times. This illustrates that the mapping between the description and the code is non-linear, as no information is given in the text regarding the specifics of the implementation.

```
class DivineFavor(SpellCard):
    def __init__(self):
        super().__init__("Divine Favor", 3,
    CHARACTER_CLASS.PALADIN, CARD_RARITY.RARE)

    def use(self, player, game):
        super().use(player, game)
        difference = len(game.other_player.hand)
       - len(player.hand)
        for i in range(0, difference):
            player.draw()
```

Figure 2: Code for the HS card "Divine Favor".

3 Problem Definition

Given the description of a card x, our decoding problem is to find the code $\hat{y}$ so that:

$$\hat{y} = \underset{y}{\operatorname{argmax}} \log P(y \mid x) \quad (1)$$

Here $\log P(y \mid x)$ is estimated by a given model. We define $y = y_1..y_{|y|}$ as the sequence of characters of the code with length $|y|$. We index each input field with $k = 1..|x|$, where $|x|$ quantifies the

[2] github.com/magefree/mage/

[3] github.com/danielyule/hearthbreaker/

number of input fields. $|x_k|$ denotes the number of tokens in x_k and x_{ki} selects the i-th token.

4 Structured Attention

Background When $|x| = 1$, the attention model of Bahdanau et al. (2014) applies. Following the chain rule, $\log P(y|x) = \sum_{t=1..|y|} \log P(y_t|y_1..y_{t-1}, x)$, each token y_t is predicted conditioned on the previously generated sequence $y_1..y_{t-1}$ and input sequence $x_1 = x_{11}..x_{1|x_1|}$. Probability are estimated with a softmax over the vocabulary Y:

$$p(y_t|y_1..y_{t-1}, x_1) = \underset{y_t \in Y}{\text{softmax}}(\mathbf{h}_t) \quad (2)$$

where $\mathbf{h}_t$ is the Recurrent Neural Network (RNN) state at time stamp t, which is modeled as $g(\mathbf{y}_{t-1}, \mathbf{h}_{t-1}, \mathbf{z}_t)$. $g(\cdot)$ is a recurrent update function for generating the new state $\mathbf{h}_t$ based on the previous token $\mathbf{y}_{t-1}$, the previous state $\mathbf{h}_{t-1}$, and the input text representation $\mathbf{z}_t$. We implement g using a Long Short-Term Memory (LSTM) RNNs (Hochreiter and Schmidhuber, 1997).

The attention mechanism generates the representation of the input sequence $\mathbf{x} = x_{11}..x_{1|x_1|}$, and $\mathbf{z}_t$ is computed as the weighted sum $\mathbf{z}_t = \sum_{i=1..|x_1|} a_i h(x_{1i})$, where a_i is the attention coefficient obtained for token x_{1i} and h is a function that maps each x_{1i} to a continuous vector. In general, h is a function that projects x_{1i} by learning a lookup table, and then embedding contextual words by defining an RNN. Coefficients a_i are computed with a softmax over input tokens $x_{11}..x_{1|x_1|}$:

$$a_i = \underset{x_{1i} \in x}{\text{softmax}}\left(v(h(x_{1i}), \mathbf{h}_{t-1})\right) \quad (3)$$

Function v computes the affinity of each token x_{1i} and the current output context $\mathbf{h}_{t-1}$. A common implementation of v is to apply a linear projection from $h(x_{1i}) : \mathbf{h}_{t-1}$ (where : is the concatenation operation) into a fixed size vector, followed by a $\tanh$ and another linear projection.

Our Approach We extend the computation of $\mathbf{z}_t$ for cases when x corresponds to multiple fields. Figure 3 illustrates how the MTG card "Serra Angel" is encoded, assuming that there are two singular fields and one text field. We first encode each token x_{ki} using the C2W model described in Ling et al. (2015), which is a replacement for lookup tables where word representations are learned at the

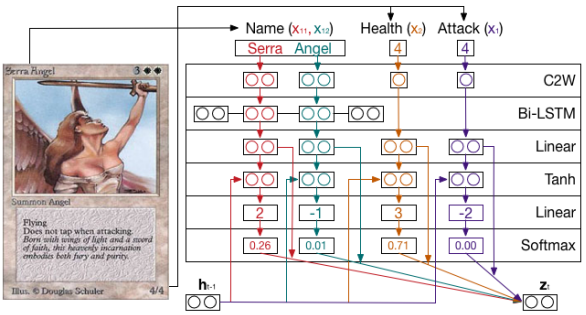


Figure 3: Illustration of the structured attention mechanism operating on a single time stamp t.

character level (cf. *C2W* row). A context-aware representation is built for words in the text fields using a bidirectional LSTM (cf. *Bi-LSTM* row). Computing attention over multiple input fields is problematic as each input field's vectors have different sizes and value ranges. Thus, we learn a linear projection mapping each input token x_{ki} to a vector with a common dimensionality and value range (cf. *Linear* row). Denoting this process as $f(x_{ki})$, we extend Equation 3 as:

$$a_{ki} = \underset{x_{ki} \in x}{\text{softmax}}(v(f(x_{ki}), \mathbf{h}_{t-1})) \quad (4)$$

Here a scalar coefficient a_{ki} is computed for each input token x_{ki} (cf. "Tanh", "Linear", and "Softmax" rows). Thus, the overall input representation $\mathbf{z}_t$ is computed as:

$$\mathbf{z}_t = \sum_{k=1..|x|, i=1..|x_k|} a_{ij} f(x_{ki}) \quad (5)$$

5 Latent Predictor Networks

Background In order to decode from x to y, many words must be copied into the code, such as the name of the card, the attack and the cost values. If we observe the HS card in Figure 1 and the respective code in Figure 2, we observe that the name "Divine Favor" must be copied into the class name and in the constructor, along with the cost of the card "3". As explained earlier, this problem is not specific to our task: for instance, in the dataset of Oda et al. (2015), a model must learn to map from `timeout = int ( timeout )` to "convert timeout into an integer.", where the name of the variable "timeout" must be copied into the output sequence. The same issue exists for proper nouns in machine translation

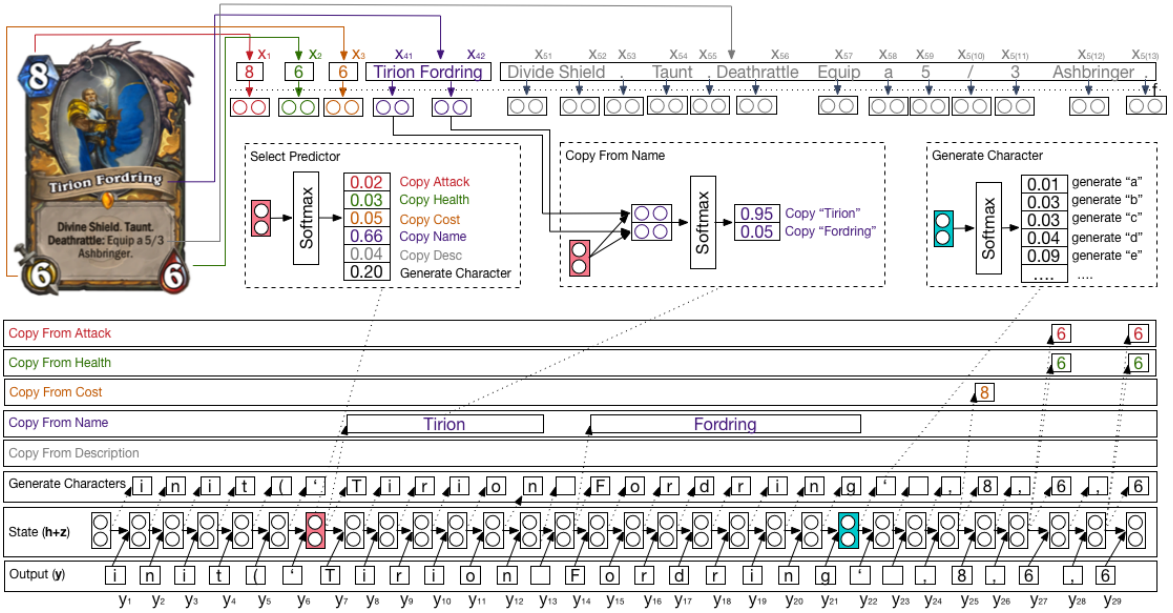


Figure 4: Generation process for the code `init('Tirion Fordring',8,6,6)` using LPNs.

which are typically copied from one language to the other. Pointer networks (Vinyals et al., 2015) address this by defining a probability distribution over a set of units that can be copied $c = c_1..c_{|c|}$. The probability of copying a unit c_i is modeled as:

$$p(c_i) = \underset{c_i \in c}{\text{softmax}}(v(h(c_i), \mathbf{q})) \quad (6)$$

As in the attention model (Equation 3), v is a function that computes the affinity between an embedded copyable unit $h(c_i)$ and an arbitrary vector $\mathbf{q}$.

Our Approach Combining pointer networks with a character-based softmax is in itself difficult as these generate segments of different granularity and there is no ground truth of which predictor to use at each time stamp. We now describe Latent Predictor Networks, which model the conditional probability $\log P(y|x)$ over the latent sequence of predictors used to generate y.

We assume that our model uses multiple predictors $r \in R$, where each r can generate multiple segments $s_t = y_t..y_{t+|s_t|-1}$ with arbitrary length $|s_t|$ at time stamp t. An example is illustrated in Figure 4, where we observe that to generate the code `init('Tirion Fordring',8,6,6)`, a pointer network can be used to generate the sequences y_7^{13}=`Tirion` and y_{14}^{22}=`Fordring` (cf. "Copy From Name" row). These sequences can also be generated using a character softmax (cf. "Generate Characters" row). The same applies to the generation of the attack, health and cost values as each of these predictors is an element in R. Thus, we define our objective function as a marginal log likelihood function over a latent variable ω:

$$\log P(y \mid x) = \log \sum_{\omega \in \bar{\omega}} P(y, \omega \mid x) \quad (7)$$

Formally, ω is a sequence of pairs r_t, s_t, where $r_t \in R$ denotes the predictor that is used at timestamp t and s_t the generated string. We decompose $P(y, \omega \mid x)$ as the product of the probabilities of segments s_t and predictors r_t:

$$P(y, \omega \mid x) = \prod_{r_t, s_t \in \omega} P(s_t, r_t \mid y_1..y_{t-1}, x) =$$

$$\prod_{r_t, s_t \in \omega} P(s_t \mid y_1..y_{t-1}, x, r_t) P(r_t \mid y_1..y_{t-1}, x)$$

where the generation of each segment is performed in two steps: select the predictor r_t with probability $P(r_t \mid y_1..y_{t-1}, x)$ and then generate s_t conditioned on predictor r_t with probability $\log P(s_t \mid y_1..y_{t-1}, x, r_t)$. The probability of each predictor is computed using a softmax over all predictors in R conditioned on the previous state $\mathbf{h_{t-1}}$ and the input representation $\mathbf{z}_t$ (cf. "Select Predictor" box). Then, the probability of generating the segment s_t depends on the predictor type. We define three types of predictors:

Character Generation Generate a single character from observed characters from the training data. Only one character is generated at each time stamp with probability given by Equation 2.

Copy Singular Field For singular fields only the field itself can be copied, for instance, the value of the attack and cost attributes or the type of card. The size of the generated segment is the number of characters in the copied field and the segment is generated with probability 1.

Copy Text Field For text fields, we allow each of the words x_{ki} within the field to be copied. The probability of copying a word is learned with a pointer network (cf. "Copy From Name" box), where $h(c_i)$ is set to the representation of the word $f(x_{ki})$ and $\mathbf{q}$ is the concatenation $\mathbf{h}_{t-1} : \mathbf{z}_t$ of the state and input vectors. This predictor generates a segment with the size of the copied word.

It is important to note that the state vector $\mathbf{h}_{t-1}$ is generated by building an RNN over the sequence of characters up until the time stamp $t-1$, i.e. the previous context $\mathbf{y}_{t-1}$ is encoded at the character level. This allows the number of possible states to remain tractable at training time.

5.1 Inference

At training time we use back-propagation to maximize the probability of observed code, according to Equation 7. Gradient computation must be performed with respect to each computed probability $P(r_t \mid y_1..y_{t-1}, x)$ and $P(s_t \mid y_1..y_{t-1}, x, r_t)$. The derivative $\frac{\partial \log P(y|x)}{\partial P(r_t|y_1..y_{t-1},x)}$ yields:

$$\frac{\partial \alpha_t P(r_t \mid y_1..y_{t-1}, x)\beta_{t,r_t} + \xi_{r_t}}{P(y \mid x)\partial P(r_t \mid y_1..y_{t-1}, x)} = \frac{\alpha_t \beta_{t,r_t}}{\alpha_{|y|+1}}$$

Here α_t denotes the cumulative probability of all values of ω up until time stamp t and $\alpha_{|y|+1}$ yields the marginal probability $P(y \mid x)$. $\beta_{t,r_t} = P(s_t \mid y_1..y_{t-1})\beta_{t+|s_t|-1}$ denotes the cumulative probability starting from predictor r_t at time stamp t, exclusive. This includes the probability of the generated segment $P(s_t \mid y_1..y_{t-1}, x, r_t)$ and the probability of all values of ω starting from timestamp $t+|s_t|-1$, that is, all possible sequences that generate segment y after segment s_t is produced. For completeness, ξ_r denotes the cumulative probabilities of all ω that do not include r_t. To illustrate this, we refer to Figure 4 and consider the timestamp $t = 14$, where the segment s_{14} =`Fordring` is generated. In this case, the cumulative probability α_{14} is the sum of the path that generates the sequence `init('Tirion` with characters alone, and the path that generates the word `Tirion` by copying from the input. β_{21} includes the probability of all paths that follow the generation of `Fordring`, which include $2\times3\times3$ different paths due to the three decision points that follow (e.g. generating 8 using a character softmax vs. copying from the cost). Finally, ξ_r refers to the path that generates `Fordring` character by character.

While the number of possible paths grows exponentially, α and β can be computed efficiently using the forward-backward algorithm for Semi-Markov models (Sarawagi and Cohen, 2005), where we associate $P(r_t \mid y_1..y_{t-1}, x)$ to edges and $P(s_t \mid y_1..y_{t-1}, x, r_t)$ to nodes in the Markov chain.

The derivative $\frac{\partial \log P(y|x)}{\partial P(s_t|y_1..y_{t-1},x,r_t)}$ can be computed using the same logic:

$$\frac{\partial \alpha_{t,s_t} P(s_t \mid y_1..y_{t-1}, x, r_t)\beta_{t+|s_t|-1} + \xi_{r_t}}{P(y \mid x)\partial P(s_t \mid y_1..y_{t-1}, x, r_t)} = \frac{\alpha_{t,r_t}\beta_{t+|s_t|-1}}{\alpha_{|y|+1}}$$

Once again, we denote $\alpha_{t,r_t} = \alpha_t P(r_t \mid y_1..y_{t-1}, x)$ as the cumulative probability of all values of ω that lead to s_t, exclusive.

An intuitive interpretation of the derivatives is that gradient updates will be stronger on probability chains that are more likely to generate the output sequence. For instance, if the model learns a good predictor to copy names, such as `Fordring`, other predictors that can also generate the same sequences, such as the character softmax will allocate less capacity to the generation of names, and focus on elements that they excel at (e.g. generation of keywords).

5.2 Decoding

Decoding is performed using a stack-based decoder with beam search. Each state S corresponds to a choice of predictor r_t and segment s_t at a given time stamp t. This state is scored as $V(S) = \log P(s_t \mid y_1..y_{t-1}, x, r_t) + \log P(r_t \mid y_1..y_{t-1}, x) + V(prev(S))$, where $prev(S)$ denotes the predecessor state of S. At each time stamp, the n states with the highest scores V are expanded, where n is the size of the beam. For each predictor r_t, each output s_t generates a new state. Finally, at each timestamp t, all states

which produce the same output up to that point are merged by summing their probabilities.

6 Code Compression

As the attention-based model traverses all input units at each generation step, generation becomes quite expensive for datasets such as MTG where the average card code contains 1,080 characters. While this is not the essential contribution in our paper, we propose a simple method to compress the code while maintaining the structure of the code, allowing us to train on datasets with longer code (e.g., MTG).

The idea behind that method is that many keywords in the programming language (e.g., `public` and `return`) as well as frequently used functions and classes (e.g., `Card`) can be learned without character level information. We exploit this by mapping such strings onto additional symbols X_i (e.g., `public class copy()` $\rightarrow$ "X_1 X_2 X_3()"). Formally, we seek the string $\hat{v}$ among all strings $V(max)$ up to length max that maximally reduces the size of the corpus:

$$\hat{v} = \underset{v \in V(max)}{\operatorname{argmax}} (len(v) - 1)C(v) \qquad (8)$$

where $C(v)$ is the number of occurrences of v in the training corpus and $len(v)$ its length. $(len(v) - 1)C(v)$ can be seen as the number of characters reduced by replacing v with a non-terminal symbol. To find $q(v)$ efficiently, we leverage the fact that $C(v) \leq C(v')$ if v contains v'. It follows that $(max - 1)C(v) \leq (max - 1)C(v')$, which means that the maximum compression obtainable for v at size max is always lower than that of v'. Thus, if we can find a $\bar{v}$ such that $(len(\bar{v}) - 1)C(\bar{v}) > (max - 1)C(v')$, that is $\bar{v}$ at the current size achieves a better compression rate than v' at the maximum length, then it follows that all sequences that contain v can be discarded as candidates. Based on this idea, our iterative search starts by obtaining the counts $C(v)$ for all segments of size $s = 2$, and computing the best scoring segment $\bar{v}$. Then, we build a list $L(s)$ of all segments that achieve a better compression rate than $\bar{v}$ at their maximum size. At size $s + 1$, only segments that contain a element in $L(s - 1)$ need to be considered, making the number of substrings to be tested to be tractable as s increases. The algorithm stops once s reaches max or the newly generated list $L(s)$ contains no elements.

X	v	size
X_1	card)⇓{⇓super(card);⇓}⇓@Override⇓public	1041
X_2	bility	1002
X_3	;⇓this.	964
X_4	(UUID_ownerId)⇓{⇓super(ownerId	934
X_5	public_	907
X_6	new_	881
X_7	_copy()	859
X_8	}")X_3expansionSetCode_=_"	837
X_9	X_6CardType[]{CardType.	815
X_{10}	ffect	794

Table 2: First 10 compressed units in MTG. We replaced newlines with ⇓ and spaces with _.

Once $\hat{v}$ is obtained, we replace all occurrences of $\hat{v}$ with a new non-terminal symbol. This process is repeated until a desired average size for the code is reached. While training is performed on the compressed code, the decoding will undergo an additional step, where the compressed code is restored by expanding the all X_i. Table 2 shows the first 10 replacements from the MTG dataset, reducing its average size from 1080 to 794.

7 Experiments

Datasets Tests are performed on the two datasets provided in this paper, described in Table 1. Additionally, to test the model's ability of generalize to other domains, we report results in the Django dataset (Oda et al., 2015), comprising of 16000 training, 1000 development and 1805 test annotations. Each data point consists of a line of Python code together with a manually created natural language description.

Neural Benchmarks We implement two standard neural networks, namely a sequence-to-sequence model (Sutskever et al., 2014) and an attention-based model (Bahdanau et al., 2014). The former is adapted to work with multiple input fields by concatenating them, while the latter uses our proposed attention model. These models are denoted as "Sequence" and "Attention".

Machine Translation Baselines Our problem can also be viewed in the framework of semantic parsing (Wong and Mooney, 2006; Lu et al., 2008; Jones et al., 2012; Artzi et al., 2015). Unfortunately, these approaches define strong assumptions regarding the grammar and structure of the output, which makes it difficult to generalize for other domains (Kwiatkowski et al., 2010). However, the work in Andreas et al. (2013) provides

evidence that using machine translation systems without committing to such assumptions can lead to results competitive with the systems described above. We follow the same approach and create a phrase-based (Koehn et al., 2007) model and a hierarchical model (or PCFG) (Chiang, 2007) as benchmarks for the work presented here. As these models are optimized to generate words, not characters, we implement a tokenizer that splits on all punctuation characters, except for the "_" character. We also facilitate the task by splitting CamelCase words (e.g., `class TirionFordring` → `class Tirion Fordring`). Otherwise all class names would not be generated correctly by these methods. We used the models implemented in Moses to generate these baselines using standard parameters, using IBM Alignment Model 4 for word alignments (Och and Ney, 2003), MERT for tuning (Sokolov and Yvon, 2011) and a 4-gram Kneser-Ney Smoothed language model (Heafield et al., 2013). These models will be denoted as "Phrase" and "Hierarchical", respectively.

Retrieval Baseline It was reported in (Quirk et al., 2015) that a simple retrieval method that outputs the most similar input for each sample, measured using Levenshtein Distance, leads to good results. We implement this baseline by computing the average Levenshtein Distance for each input field. This baseline is denoted "Retrieval".

Evaluation A typical metric is to compute the accuracy of whether the generated code exactly matches the reference code. This is informative as it gives an intuition of how many samples can be used without further human post-editing. However, it does not provide an illustration on the degree of closeness to achieving the correct code. Thus, we also test using BLEU-4 (Papineni et al., 2002) at the token level. There are clearly problems with these metrics. For instance, source code can be correct without matching the reference. The code in Figure 2, could have also been implemented by calling the `draw` function in an cycle that exists once both players have the same number of cards in their hands. Some tasks, such as the generation of queries (Zelle and Mooney, 1996), have overcome this problem by executing the query and checking if the result is the same as the annotation. However, we shall leave the study of these methologies for future work, as adapting these methods for our tasks is not trivial. For instance, the correctness cards with conditional (e.g. `if player has no cards, then draw a card`) or non-deterministc (e.g. `put a random card in your hand`) effects cannot be simply validated by running the code.

Setup The multiple input types (Figure 3) are hyper-parametrized as follows: The C2W model (cf. "C2W" row) used to obtain continuous vectors for word types uses character embeddings of size 100 and LSTM states of size 300, and generates vectors of size 300. We also report on results using word lookup tables of size 300, where we replace singletons with a special unknown token with probability 0.5 during training, which is then used for out-of-vocabulary words. For text fields, the context (cf. "Bi-LSTM" row) is encoded with a Bi-LSTM of size 300 for the forward and backward states. Finally, a linear layer maps the different input tokens into a common space with of size 300 (cf. "Linear" row). As for the attention model, we used an hidden layer of size 200 before applying the non-linearity (row "Tanh"). As for the decoder (Figure 4), we encode output characters with size 100 (cf. "output (y)" row), and an LSTM state of size 300 and an input representation of size 300 (cf. "State(h+z)" row). For each pointer network (e.g., "Copy From Name" box), the intersection between the input units and the state units are performed with a vector of size 200. Training is performed using mini-batches of 20 samples using AdaDelta (Zeiler, 2012) and we report results using the iteration with the highest BLEU score on the validation set (tested at intervals of 5000 mini-batches). Decoding is performed with a beam of 1000. As for compression, we performed a grid search over compressing the code from 0% to 80% of the original average length over intervals of 20% for the HS and Django datasets. On the MTG dataset, we are forced to compress the code up to 80% due to performance issues when training with extremely long sequences.

7.1 Results

Baseline Comparison Results are reported in Table 3. Regarding the retrieval results (cf. "Retrieval" row), we observe the best BLEU scores among the baselines in the card datasets (cf. "MTG" and "HS" columns). A key advantage of this method is that retrieving existing entities guarantees that the output is well formed, with no

	MTG		HS		Django	
	BLEU	Acc	BLEU	Acc	BLEU	Acc
Retrieval	54.9	0.0	62.5	0.0	18.6	14.7
Phrase	49.5	0.0	34.1	0.0	47.6	31.5
Hierarchical	50.6	0.0	43.2	0.0	35.9	9.5
Sequence	33.8	0.0	28.5	0.0	44.1	33.2
Attention	50.1	0.0	43.9	0.0	58.9	38.8
Our System	**61.4**	**4.8**	65.6	4.5	**77.6**	**62.3**
– C2W	60.9	4.4	**67.1**	4.5	75.9	60.9
– Compress	-	-	59.7	**6.1**	76.3	61.3
– LPN	52.4	0.0	42.0	0.0	63.3	40.8
– Attention	39.1	0.5	49.9	3.0	48.8	34.5

Table 3: BLEU and Accuracy scores for the proposed task on two in-domain datasets (HS and MTG) and an out-of-domain dataset (Django).

Compression	0%	20%	40%	60%	80%
Seconds Per Card					
Softmax	2.81	2.36	1.88	1.42	**0.94**
LPN	3.29	2.65	2.35	1.93	**1.41**
BLEU Scores					
Softmax	44.2	46.9	47.2	51.4	**52.7**
LPN	59.7	62.8	61.1	66.4	**67.1**

Table 4: Results with increasing compression rates with a regular softmax (cf. "Softmax") and a LPN (cf. "LPN"). Performance values (cf. "Seconds Per Card" block) are computed using one CPU.

syntactic errors such as producing a non-existent function call or generating incomplete code. As BLEU penalizes length mismatches, generating code that matches the length of the reference provides a large boost. The phrase-based translation model (cf. "Phrase" row) performs well in the Django (cf. "Django" column), where mapping from the input to the output is mostly monotonic, while the hierarchical model (cf. "Hierarchical" row) yields better performance on the card datasets as the concatenation of the input fields needs to be reordered extensively into the output sequence. Finally, the sequence-to-sequence model (cf. "Sequence" row) yields extremely low results, mainly due to the lack of capacity needed to memorize whole input and output sequences, while the attention based model (cf. "Attention" row) produces results on par with phrase-based systems. Finally, we observe that by including all the proposed components (cf. "Our System" row), we obtain significant improvements over all baselines in the three datasets and is the only one that obtains non-zero accuracies in the card datasets.

Component Comparison We present ablation results in order to analyze the contribution of each of our modifications. Removing the C2W model (cf. "– C2W" row) yields a small deterioration, as word lookup tables are more susceptible to sparsity. The only exception is in the HS dataset, where lookup tables perform better. We believe that this is because the small size of the training set does not provide enough evidence for the character model to scale to unknown words. Surprisingly, running our model compression code (cf. "– Compress" row) actually yields better results. Table 4 provides an illustration of the results for different compression rates. We obtain the best results with an 80% compression rate (cf. "BLEU Scores" block), while maximising the time each card is processed (cf. "Seconds Per Card" block). While the reason for this is uncertain, it is similar to the finding that language models that output characters tend to under-perform those that output words (Józefowicz et al., 2016). This applies when using the regular optimization process with a character softmax (cf. "Softmax" rows), but also when using the LPN (cf. "LPN" rows). We also note that the training speed of LPNs is not significantly lower as marginalization is performed with a dynamic program. Finally, a significant decrease is observed if we remove the pointer networks (cf. "– LPN" row). These improvements also generalize to sequence-to-sequence models (cf. "– Attention" row), as the scores are superior to the sequence-to-sequence benchmark (cf. "Sequence" row).

Result Analysis Examples of the code generated for two cards are illustrated in Figure 5. We obtain the segments that were copied by the pointer networks by computing the most likely predictor for those segments. We observe from the marked segments that the model effectively copies the attributes that match in the output, including the name of the card that must be collapsed. As expected, the majority of the errors originate from inaccuracies in the generation of the effect of the card. While it is encouraging to observe that a small percentage of the cards are generated correctly, it is worth mentioning that these are the result of many cards possessing similar effects. The "Madder Bomber" card is generated correctly as there is a similar card "Mad Bomber" in the training set, which implements the same effect, except that it deals 3 damage instead of 6. Yet, it is a promising result that the model was able to capture

this difference. However, in many cases, effects that radically differ from seen ones tend to be generated incorrectly. In the card "Preparation", we observe that while the properties of the card are generated correctly, the effect implements a unrelated one, with the exception of the value 3, which is correctly copied. Yet, interestingly, it still generates a valid effect, which sets a minion's attack to 3. Investigating better methods to accurately generate these effects will be object of further studies.

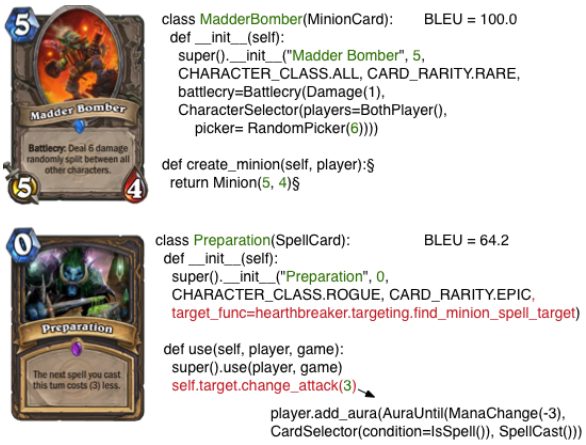


Figure 5: Examples of decoded cards from HS. Copied segments are marked in green and incorrect segments are marked in red.

8 Related Work

While we target widely used programming languages, namely, Java and Python, our work is related to studies on the generation of any executable code. These include generating regular expressions (Kushman and Barzilay, 2013), and the code for parsing input documents (Lei et al., 2013). Much research has also been invested in generating formal languages, such as database queries (Zelle and Mooney, 1996; Berant et al., 2013), agent specific language (Kate et al., 2005) or smart phone instructions (Le et al., 2013). Finally, mapping natural language into a sequence of actions for the generation of executable code (Branavan et al., 2009). Finally, a considerable effort in this task has focused on semantic parsing (Wong and Mooney, 2006; Jones et al., 2012; Lei et al., 2013; Artzi et al., 2015; Quirk et al., 2015). Recently proposed models focus on Combinatory Categorical Grammars (Kushman and Barzilay, 2013; Artzi et al., 2015), Bayesian Tree Transducers (Jones et al., 2012; Lei et al., 2013) and Probabilistic Context Free Grammars (Andreas et al., 2013). The work in natural language programming (Vadas and Curran, 2005; Manshadi et al., 2013), where users write lines of code from natural language, is also related to our work. Finally, the reverse mapping from code into natural language is explored in (Oda et al., 2015).

Character-based sequence-to-sequence models have previously been used to generate code from natural language in (Mou et al., 2015). Inspired by these works, LPNs provide a richer framework by employing attention models (Bahdanau et al., 2014), pointer networks (Vinyals et al., 2015) and character-based embeddings (Ling et al., 2015). Our formulation can also be seen as a generalization of Allamanis et al. (2016), who implement a special case where two predictors have the same granularity (a sub-token softmax and a pointer network). Finally, HMMs have been employed in neural models to marginalize over label sequences in (Collobert et al., 2011; Lample et al., 2016) by modeling transitions between labels.

9 Conclusion

We introduced a neural network architecture named *Latent Prediction Network*, which allows efficient marginalization over multiple predictors. Under this architecture, we propose a generative model for code generation that combines a character level softmax to generate language-specific tokens and multiple pointer networks to copy keywords from the input. Along with other extensions, namely structured attention and code compression, our model is applied on on both existing datasets and also on a newly created one with implementations of TCG game cards. Our experiments show that our model out-performs multiple benchmarks, which demonstrate the importance of combining different types of predictors.

References

M. Allamanis, H. Peng, and C. Sutton. 2016. A Convolutional Attention Network for Extreme Summarization of Source Code. *ArXiv e-prints*, February.

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 47–52, August.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, September.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 82–90.

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51th Annual Meeting on Association for Computational Linguistics*, pages 690–696.

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

Bevan Keeley Jones, Mark Johnson, and Sharon Goldwater. 2012. Semantic parsing with bayesian tree transducers. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 488–496.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Rohit J. Kate, Yuk Wah Wong, and Raymond J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 1062–1068, Pittsburgh, PA, July.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 826–836, Atlanta, Georgia, June.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233.

G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer. 2016. Neural Architectures for Named Entity Recognition. *ArXiv e-prints*, March.

Vu Le, Sumit Gulwani, and Zhendong Su. 2013. Smartsynth: Synthesizing smartphone automation scripts from natural language. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, pages 193–206.

Tao Lei, Fan Long, Regina Barzilay, and Martin Rinard. 2013. From natural language specifications to program input parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1294–1303, Sofia, Bulgaria, August.

Wang Ling, Tiago Luís, Luís Marujo, Rámon Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 783–792, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mehdi Hafezi Manshadi, Daniel Gildea, and James F. Allen. 2013. Integrating programming by example and natural language programming. In Marie desJardins and Michael L. Littman, editors, *AAAI*. AAAI Press.

Lili Mou, Rui Men, Ge Li, Lu Zhang, and Zhi Jin. 2015. On end-to-end program generation from user intention by deep neural networks. *CoRR*, abs/1510.07211.

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

Yusuke Oda, Hiroyuki Fudaba, Graham Neubig, Hideaki Hata, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Learning to generate pseudo-code from source code using statistical machine translation. In *30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Lincoln, Nebraska, USA, November.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 878–888, Beijing, China, July.

Sunita Sarawagi and William W. Cohen. 2005. Semimarkov conditional random fields for information extraction. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1185–1192. MIT Press.

Artem Sokolov and François Yvon. 2011. Minimum Error Rate Semi-Ring. In Mikel Forcada and Heidi Depraetere, editors, *Proceedings of the European Conference on Machine Translation*, pages 241–248, Leuven, Belgium.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215.

David Vadas and James R. Curran. 2005. Programming with unrestricted natural language. In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 191–199, Sydney, Australia, December.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2674–2682. Curran Associates, Inc.

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

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

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR, August. AAAI Press/MIT Press.

Easy Things First: Installments Improve Referring Expression Generation for Objects in Photographs

Sina Zarrieß David Schlangen
Dialogue Systems Group // CITEC // Faculty of Linguistics and Literary Studies
Bielefeld University, Germany
first.last@uni-bielefeld.de

Abstract

Research on generating referring expressions has so far mostly focussed on "one-shot reference", where the aim is to generate a single, discriminating expression. In interactive settings, however, it is not uncommon for reference to be established in "installments", where referring information is offered piecewise until success has been confirmed. We show that this strategy can also be advantageous in technical systems that only have uncertain access to object attributes and categories. We train a recently introduced model of grounded word meaning on a data set of REs for objects in images and learn to predict semantically appropriate expressions. In a human evaluation, we observe that users are sensitive to inadequate object names - which unfortunately are not unlikely to be generated from low-level visual input. We propose a solution inspired from human task-oriented interaction and implement strategies for avoiding and repairing semantically inaccurate words. We enhance a word-based REG with context-aware, referential installments and find that they substantially improve the referential success of the system.

1 Introduction

A speaker who wants to refer to an object in a visual scene will try to produce a referring expression (RE) that (i) is semantically adequate, i.e. accurately describes the visual properties of the target referent, and (ii) is pragmatically and contextually appropriate, i.e. distinguishes the target from other objects in the scene but does not overload the listener with unnecessary information. Figure 1 illustrates this with two examples from a corpus of REs collected from human subjects for objects in images (Kazemzadeh et al., 2014).

Figure 1: Example images and REs from the ReferIt corpus (Kazemzadeh et al., 2014)

Research on referring expression generation (REG) has mostly focussed on (ii), modeling pragmatic adequacy in attribute selection tasks, using as input a fully specified, symbolic representation of the visual attributes of an object and its distractors in a scene (Dale and Reiter, 1995; Krahmer and Van Deemter, 2012).

In this paper, we follow a more recent trend (Kazemzadeh et al., 2014; Gkatzia et al., 2015) and investigate REG on real-world images. In this setting, a low-level visual representation of an image (a scene) segmented into regions (objects), including the region of the target referent, constitutes the input. This task is closely related to the recently very active field of image-to-text generation, where deep learning approaches have been used to directly map low-level visual input to natural language sentences, e.g. (Vinyals et al., 2015; Chen and Lawrence Zitnick, 2015; Devlin et al., 2015). Similarly, we propose to cast REG on images as a word selection task. Thus, we base this work on a model of perceptually grounded word meaning, which associates words with classifiers that predict their semantic appropriateness given

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 610–620,
Berlin, Germany, August 7-12, 2016.

the low-level visual features of an object (Kennington and Schlangen, 2015). As our first contribution, we train this model on the ReferIt corpus (Kazemzadeh et al., 2014) and define decoding mechanisms tailored to REG.

Large-scale recognition of objects and their attributes in images is still a non-trivial task. Consequently, REG systems now face the challenge of dealing with semantically inadequate expressions. For instance, in Figure 1, the system might not precisely distinguish between *man* or *woman* and generate an inadequate, confusing RE like *man in the middle*. Therefore, we focus on evaluating our system in an object identification task with users, in contrast to previous approaches to REG on images (Mao et al., 2015). In order to assess possible sources of misunderstanding more precisely, our set-up also introduces a restricted form of interaction: instead of measuring "one-shot" performance only, users have three trials for identifying a referent. In this set-up, we find that different parameter settings of the systems (e.g. their visual inputs) have a clear effect on the referential success rates, while automatic evaluation measures reflect the interactive effectiveness rather poorly.

Research on reference in human interaction has noticed that conversation partners try to minimize their joint effort and often prefer to present simple expressions that can be expanded on or repaired, if necessary (Clark and Wilkes-Gibbs, 1986). This strategy, called "referring in installments" is very effective for achieving common ground in task-oriented interaction (Fang et al., 2014) and is attested in dialogue data (Striegnitz et al., 2012). The connection between reference in installments on the one and the status of distractors and distinguishing expressions on the other hand is relatively unexplored, though it seems natural to combine the two perspectives (DeVault et al., 2005). Figure 1 shows an example for very a simple but highly effective expression - it mentions color as a salient and distinguishing property while avoiding a potentially unclear object name.

As our second contribution, we extend our probabilistic word selection model to work in a simple interactive installment component that tries to avoid semantically inadequate words as much as possible and only expands the expression in case of misunderstanding. We present an algorithm that generates these installments depending on the context, based on ideas from traditional REG algorithms like (Dale and Reiter, 1995). We find that a context-aware installment strategy greatly improves referential success as it helps to avoid and repair misunderstandings and offers a combined treatment of semantic and pragmatic adequacy.

2 Background

2.1 Approaches to REG

"One-shot REG" Foundational work in REG has investigated attribute selection algorithms (Dale and Reiter, 1995) that compute a *distinguishing* referring expression for an object in a visual scene, which is defined as a target object r, set of distractor objects $D = \{d_1, d_2, d_3, ...\}$ and a set of attributes $A = \{type, position, size, color, ...\}$. A manually specified database typically associates the target and distractors in D with atomic values for each attribute, cf. (Krahmer and Van Deemter, 2012). In this setting, an attribute $a_1 \in A$ is said to *rule out* a distractor object from D, if the target and distractor have different values. This is mostly based on the assumption that we have objects of particular types (e.g. people, furniture, etc.) and that the system has perfect knowledge about these object types and, consequently, about potential distractors of the target. This does not apply to REG on real-world images which, as we will show in this paper, triggers some new challenges and research questions for this field. Subsequent work has shown that human speakers do not necessarily produce minimally distinguishing expressions (van Deemter et al., 2006; Viethen and Dale, 2008; Koolen et al., 2011), and has tried to account for the wide range of factors - such as different speakers, modalities, object categories - that are related to attribute selection, cf. (Mitchell et al., 2010; Koolen and Krahmer, 2010; Clarke et al., 2013; Tarenskeen et al., 2015).

Task-oriented REG has looked at reference as a collaborative process where a speaker and a listener try to reach a common goal (Clark and Wilkes-Gibbs, 1986; Heeman and Hirst, 1995; DeVault et al., 2005). Given the real-time constraints of situated interaction, a speaker often has to start uttering before she has found the optimal expression, but at the same time, she can tailor, extend, adapt, revise or correct her referring expressions in case the listener signals that he did not understand. Thus, human speakers can flex-

ibly split and adapt their REs over several utterances during an interaction, a phenomenon called "reference in installments". In a corpus analysis of the S-GIVE domain, (Striegnitz et al., 2012) showed that installments are pervasive in human-human interaction in a task-oriented environment. However, while there has been research on goal-oriented and situated REG (Stoia et al., 2006; Kelleher and Kruijff, 2006; Striegnitz et al., 2011; Garoufi and Koller, 2013), installments have been rarely implemented and empirically tested in interactive systems. A noticeable exception is the work by Fang et al. (2014) who use reinforcement learning to induce an installment strategy that is targeted at robots that have uncertain knowledge about the objects in their environment. Using relatively simple computer-generated scenes and a standard representations of objects as sets of attributes, they learn a strategy that first guides the user to objects that the system can recognize with high confidence. Our work is targeted at more complex scenes in real-world images and large domains where no a priori knowledge about object types and their attributes is given. Mao et al. (2015) use a convolutional neural network and an LSTM to generate REs directly and on the same data sets as we do in this paper, but they only report automatic evaluation results.

2.2 The ReferIt corpus

We train and evaluate our system on the ReferIt data set collected by Kazemzadeh et al. (2014). The basis of the corpus is a collection of "20,000 still natural images taken from locations around the world" (Grubinger et al., 2006), which was augmented by Escalante et al. (2010) with segmentation masks identifying objects in the images (an average of 5 objects per image). This dataset also provides manual annotations of region labels and a vector of visual features for each region (e.g. region area, width, height, and color-related features). There are 256 types of objects (i.e. labels), out of which 140 labels are used for more than 50 regions (Escalante et al., 2010). Kazemzadeh et al. (2014) collected a large number of expressions referring to objects (for which segmentations exist) from these images (130k REs for 96k objects), using a game-based crowd-sourcing approach, and they have assembled an annotated test set.

2.3 The WAC model

Given a corpus of REs aligned with objects in images, we can train a model that predicts semantically appropriate words given the visual representation of an image region. We adopt the WAC ("words-as-classifiers") model (Kennington and Schlangen, 2015), which was originally used for reference resolution in situated dialogue. However, WAC is essentially a task-independent approach to predicting semantic appropriateness of words in visual contexts and can be flexibly combined with task-dependent decoding procedures.

The WAC model pairs each word w in its vocabulary V with an individual classifier that maps the low-level, real-valued visual properties of an object o to a semantic appropriateness score. In order to learn the meaning of e.g. the word *red*, the visual properties of all objects described as *red* in a corpus of REs are given as positive instances to a supervised (logistic regression) learner. Negative instances are randomly samples from the complementary set of utterances (e.g. not containing *red*).

We used this relatively simple model in our work, because first of all we wanted to test wether it scales from a controlled domain of typical reference game scenes (Kennington and Schlangen, 2015) to real-world images. Second, as compared to standard object recognisers that predict abstract image labels annotated in e.g. ImageNet (Deng et al., 2009), this model directly captures the relation between actual words used in REs and visual properties of the corresponding referents. Following (Schlangen et al., 2016), we can easily base our classifiers on such a high-performance convolutional neural network (Szegedy et al., 2015), by applying it on our images and extracting the final fully-connected layer before the classification layer (see Section 3.1).

3 Word-based REG for Image Objects

We describe a word selection model for REG on images, which reverses the decoding procedure of our reference resolution model (Kennington and Schlangen, 2015; Schlangen et al., 2016). The main question we pursue here is whether we can predict semantically adequate words for visually represented target objects in real-world images and achieve communicative success in a task-oriented evaluation.

3.1 A Basic Algorithm for REG with WAC

Given a visual representation of an object, we can apply all word classifiers from the vocabulary of our WAC model and obtain an appropriateness ranking over words. As these WAC scores do not reflect appropriateness in the linguistic context, i.e. the previously generated words, we combine them with simple language model (bigram) probabilities (LM) computed on our corpus. The combination of WAC and LM scores is used to rank our vocabulary with respect to appropriateness given the visual features of the target referent and linguistic context.

Algorithm 1 shows our implementation of the decoding step, a beam search that iteratively adds n words with the highest combined LM and WAC score to a its agenda and terminates after a prespecified number of maximum steps.

The algorithm takes the number of iterations as input, so it searches for the optimal RE given a fixed length. Deciding how many words have to be generated is very related to deciding how many attributes to include in more traditional REG. As a first approach, we have trained an additional regression classifier that predicts the length of the RE, given the number of objects in the scene and the visual properties of the target.

Algorithm 1 Baseline REG with WAC

function WORD-GEN($object, maxsteps, V$)
  $Agenda \leftarrow \{`start'\}$
  for $n \in 0..maxsteps$ **do**
    $Beam \leftarrow \emptyset$
    for $re \in Agenda$ **do**
      $w_{-1} \leftarrow$ LAST(re)
      for $w \in$ BIGRAMS(w_{-1}, V) **do**
        $s =$ WAC($w, object$) + LM(w, w_{-1})
        $re_{new} \leftarrow$ APPEND($re, word$)
        $Beam \leftarrow Beam \cup \{(re_{new}, s)\}$
      end for
    end for
    $Agenda \leftarrow$ K-BEST($Beam, k$)
  end for
  return K-BEST($Agenda, 1$)
end function

3.2 Experimental Set-up

Data We use the same test set as Kazemzadeh et al. (2014) that is divided into the 3 subsets, each containing 500 objects: "**Set A** contains objects randomly sampled from the entire dataset, **Set B** was sampled from the most frequently occurring object categories in the dataset, excluding the less interesting categories, **Set C** contains objects sampled from images that contain at least 2 objects of the same category, excluding the less interesting categories."[1] For each object, there are 3 human-generated reference REs. We train the WAC model on the set of images that are not contained in the test set, which amounts to 100384 REs.

The classifiers We use Schlangen et al. (2016)'s WAC model that is a trained on the REFERIT data (Kazemzadeh et al., 2014) based on the SAIAPR collection (Grubinger et al., 2006). We train binary logistic regression classifiers (with $\ell 1$ regularisation) for the 400 most frequent words from the training set.[2] During training, we only consider non-relational expressions, as words from relational expressions would introduce further noise. Each classifier is trained with the same balance of positive and negative examples, a fixed ratio of 1 positive to 7 negative. Additionally, we train a regression classifier that predicts the expected length of the RE given the visual features of the target object and the number of objects in the entire scene. We also train a simple bigram language model on the data.

Feature sets In this experiment, we manipulate the features sets of the underlying word classifiers. We train it on (i) a small set of 27 low-level visual features extracted and provided by Escalante et al. (2010), called SAIAPR features below, and (ii) a larger set of features automatically learned by a state-of-the-art convolutional neural network, "GoogLeNet" (Szegedy et al., 2015). We derive representations of our visual inputs with this CNN, that was trained on data from the ImageNet corpus (Deng et al., 2009), and extract the final fully-connected layer before the classification layer, to give us a 1024 dimensional representation of the region. We augment this with 7 features that encode information about the region relative to the image: the (relative) coordinates of two corners, its (relative) area, distance to the center, and orientation of the image. The full representation hence is a vector of 1031 features. The feature extraction for (ii) is described in more detail in (Schlangen et al., 2016). Generally, the SAIAPR features represent interpretable visual information on position, area, and color of an image region, they could be associated with particular visual attributes. This is not possible with the GoogLeNet features.

[1] Where objects mostly located in the background like 'sky', 'mountain' are considered to be less interesting.

[2] We used `scikit learn` (Pedregosa et al., 2011).

3.3 Automatic Evaluation

To the best of our knowledge, end-to-end REG performance has not been reported on the ReferIt data set before. Table 1 shows corpus-based BLEU and NIST measures calculated on the test set (using 3 references for each RE). The results indicate a minor gain of the GoogLeNet features. We also evaluate a version of the GoogLeNet-based system that instantiates the beam search with the gold length of the RE from the corpus (GoogLeNet$_{glen}$). This leads to a small improvement in BLEU and NIST, indicating that the length prediction is not a critical factor.

	BLEU		NIST	
	1-gram	2-gram	1-gram	2-gram
SAIAPR	0.33	0.19	1.5	1.7
GoogLeNet	0.35	0.21	1.9	2.3
GoogLeNet$_{glen}$	0.38	0.19	2.0	2.6

Table 1: Automatic evaluation for word-based REG systems

3.4 A Game-based Human Evaluation

Set-up In parallel to the reference game in (Kazemzadeh et al., 2014), we set up a game between a computer that generates REs and a human player who clicks on the location of the described object that he identifies based on the RE. After each click, the GUI presents some canned feedback and informs the player whether he clicked on the intended object. In case of an unsuccessful click, the player has two more trials. In the following, we report the success rates with respect to each trial and the different test sets. This set-up will trigger a certain amount of user guesses such that the success rates do not correspond perfectly to semantic accuracies. But it accounts for the increased difficulty as well as the interactive nature of the task. See Section 4.4 for an analysis of learning effects in this set-up and (Gatt et al., 2009; Belz and Hastie, 2014) for general discussion on REG and NLG evaluation.

	Success rate/ trial			Error
	1st	2nd	3rd	red.
SAIAPR	32.2	40.3	46.3	20.8
GoogLeNet	41.6	53.4	59.1	29.9
GoogLeNet$_{glen}$	37.6	51	58.7	33.8
human	90.6	94.6	98.3	81.9

Table 2: Human success and error reduction rates in object identification task, for different sets of visual features

For each player, we randomly sampled the games from the entire test set, but balanced the items so that they were equally distributed across the 3 test subsets A, B, C (see above) and the three systems. We also included human REs from the corpus. In total, we collected 1201 games played by 8 participants.

Results In Table 2, we report the cumulative success rates for the different systems across the different trials, i.e. the success rate in the 3rd trial corresponds to the overall proportion of successfully identified referents. First of all, this suggests that the differences in performance between the systems is much bigger in terms of their communicative effectiveness as in terms of the corpus-based measures (Table 1). Thus, on the one hand, the GoogLeNet features are clearly superior to SAIAPR, whereas differences between GoogLeNet and GoogLeNet$_{glen}$ are minor. Interestingly, the GoogLeNet features improve 1st trial as well as overall success, leading to a much better error reduction rate[3] in object identification between the first and third trial. This means that, here, humans are more likely to recover from misunderstandings and indicates that REs generated by the SAIAPR system are more semantically inadequate.

	Success rate (3rd trial)		
	Set A	Set B	Set C
SAIAPR	35.7	63.8	40.7
GoogLeNet	57	67.7	53.1
GoogLeNet$_{glen}$	50	74	53
human	99.1	99	96.5

Table 3: Human success rates for baseline REG systems trained on different visual feature sets

In Table 3, we report the overall success rates for the different test sets. All systems have a clearly higher performance on the B Set which contains the most frequent object types. Surprisingly, all systems have a largely comparable performance on Set A and C whereas only C contains images with distractors in the sense of traditional REG. This shows that describing objects which belong to an infrequent type in a semantically adequate way, which is necessary in Set A, is equally challenging as reaching pragmatic adequacy which is called for in Set C.

[3] Calculated as $(\#error_{1st} - \#error_{3rd})/\#error_{1st}$

3.5 Error Analysis

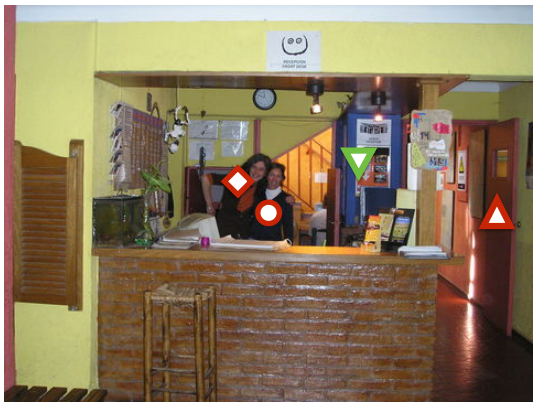
When users click on a distractor object instead of the intended referent in our object identification task, there can be several sources of misunderstanding. For instance, it is possible that the system produced REs that are interpretable but not specific and distinguishing enough to identify the target. It is also possible that the system selected words that are semantically inadequate such that the expression becomes completely misleading. We can get some insight into possible sources of misunderstanding by comparing the the clicked-on distractor objects to their intended target, using the object labels annotated for each image region (see Section 2.2).

The analysis of mismatches between the expected label of the target and the label of the object actually clicked on by the user reveals that many errors are due to semantic inadequacies and apparently severe interpretation failures: Looking at the total number of clicks on distractor objects, 80% are clicks on a distractor with a different label than the target.[4], e.g. the user clicked on a 'tree' instead of a 'man'. This is clear evidence for semantic inadequacies, suggesting that the systems often generate an inadequate noun for the object type. An example for such a label mismatch is shown in Figure 2 where the system generated "person" for referring to a "sign", such that the user first clicked on distractor objects that are persons.

Similarly, we can get some evidence about how users try to repair misunderstandings, by comparing a distractor clicked on in the first trial to another distractor clicked on in the subsequent second, or third trial. Interestingly, we find that users do not seem to be aware of the fact that the system does not always generate the correct noun and do not generally try to click on objects with a different label. Only in 39% of the unsuccessful second trials, users decided for a distractor object with a different label, even though the first click had been unsuccessful. For instance, in Figure 2, the user clicked on the other person in the image in the second trial, although this referent is clearly not on the right. This suggests that users do not easily revise their RE interpretation with respect to the intended type of referent.

Moreover, we can compare the different distractor clicks with respect to their spatial distance

[4]The percentage varies between saiapr (86%), GoogLeNet (71%)



re$_{generated}$: "person on the right"
re$_{human}$: "sign on the blue shelf in the back"

Figure 2: Example for an unsuccessful trial in object identification; first click: ◯, second click: ⋄, third click: △, target: ▽)

to the target. We find that after an unsuccessful first trial, users click on an object that has a greater distance to the target in 70% of the cases (as e.g. in Figure 2). This means that users often try to repair the misunderstanding with respect to the intended location, rather than with respect to the intended object type. Intuitively, this behaviour makes sense: a human speaker is more likely to confuse e.g. *left* and *right* than e.g. *man* and *tree*. From the perspective of the system this is a problematic situation: words like *left* and *right* are much easier to generate (based on simple positional features) than nouns like *man* and *tree*.

4 Towards interactive, contextual REG

In this Section, we extend our word-based REG to deal with semantic inadequacies. We take a first step towards interactive REG and implement installments, a pervasive strategy in human task-oriented interaction. The main idea is that the system should try to avoid semantically inadequate expressions wherever possible and, if misunderstanding occurs, try to react appropriately.

4.1 Procedure

When a speaker or system *refers in installments*, they do not need to generate an RE in one shot, but can start with an initial, simple RE that is extended or reformulated if this becomes necessary in the interaction, i.e. if the listener cannot identify the referent. This setting is a straightforward extension of our game-based evaluation in Section 3.4, where users had 3 trials for identifying a referent: instead of generating a single RE for the target

and presenting it in every trial, we now produce a triple (re_1, re_2, re_3), where re_1 will be used in the first trial, re_2 in the second trial, etc.

In this set-up, we want to investigate whether installments and reformulations help to avoid semantic inadequacies and improve referential success, i.e. whether a dynamic approach to REG compares favourably to the non-dynamic version of our system (see Section 3). This question is, however, closely linked to another, more intricate question: what is the best strategy to realize installments that, on the one hand, provide enough information so that a user can eventually identify the referent and, on the other hand, avoid misleading words? To date, even highly interactive systems do not generally treat installments, or if they do, only realise them via templates, e.g. (Stoia et al., 2006; Staudte et al., 2012; Garoufi and Koller, 2013; Dethlefs and Cuayáhuitl, 2015). As pointed out by Liu et al. (2012), data-driven approaches are not straightforward to set-up, due to the "mismatched perceptual basis" between a human listener and an REG system.

Based on the insights of our error analysis in Section 3.4, we will rely on a general installment strategy that is mostly targeted at avoiding semantically inadequate object names, and emphasizing the fact that location words generated by the system convey more reliable information. We have implemented two versions of this general strategy: (i) pattern-based installments that always avoid object names in their initial expression and dynamically extend this if necessary, (ii) context-dependent installments that condition the initial expression on the complexity of the scene and extend the initial expression accordingly, inspired by standard approaches to attribute selection in REG (Krahmer and Van Deemter, 2012). Thus, we do not test initial or reformulated expressions in isolation, but the strategy as a whole, which is similar to (Fang et al., 2014).

4.2 Pattern-based Installments

This system generates a triple of REs for each image, corresponding to the respective trials in the object identification task. The triple for pattern-based installments is defined as follows:

- re_1: a short RE that only contains location words, e.g. *bottom left*
- re_2: a longer RE that contains location words and an object name, e.g. *the car on the left*
- re_3: a reformulation of re_2 that hedges the object name and suggests an alternative object name, e.g. *vehicle or bottle on the left*

Figure 3(a) illustrates a case where this pattern is useful: the target is a horse, the biggest and most salient object in the image, which can be easily identified with a simple locative expression. As horses are not frequent in the training data, the system unfortunately generates *hat guy* as the most likely object name. This RE would be very misleading indeed if presented to a listener, as one of the distractors actually is a person with a hat.

Generation Procedure In order to generate the above installment triples with our REG system, we simply restrict the vocabulary of the underlying WAC-model. Thus, we divided the 400 word classifiers into the following subsets:

- V_1: 20 location words (manually defined)
- V_2: V_1 + 183 object names (extracted from annotated section of the ReferIt corpus)
- V_3: entire vocabulary

This basic installment-based system does not use V_3 (but see below). For generating the hedge of the object name in the third trial (re_3) we use the top-second and top-third word from the ranking that WAC produces over all object type words given the visual features of the target.

4.3 Context-dependent Installments

Our context-dependent installment strategy determines the initial RE (re_1) based on the surrounding scene and generates subsequent reformulations (re_2,re_3) accordingly.

Initial REs and Distractors As we do not have a symbolic representation of the distractor objects and their properties, we use the word-based REG system to decide whether an RE can be expected to be distinguishing for the target in the scene. This is similar to (Roy, 2002). Algorithm 2 shows the procedure for determining the initial RE (re_1). Same as before, we restrict the vocabulary of the underlying WAC model, e.g. to contain only location words. But now, we apply the word generation function to the target object and to all the other objects in the set of distractors (D). If the algorithm generates an identical chunk for the target and one of its distractors, it continues with a less restricted vocabulary and a longer expression. It terminates when it has found an RE that is optimal only for the target. This algorithm proceeds on the level of

chunks, instead of single words, as e.g. location is often described by several words (e.g. *bottom left*).

Algorithm 2 A Context-aware REG Algorithm

```
1: function INC-GEN(object, maxsteps, D, V)
2:     for n ∈ 2..maxsteps do
3:         V_n ← RESTRICT(V, n)
4:         re ← WORD-GEN(object, V_n)
5:         for d ∈ D do
6:             re_d ← WORD-GEN(d, V_n)
7:             if re_d = re then
8:                 break
9:             end if
10:        end for
11:        return re
12:    end for
13: end function
```

As we found that the linguistic quality degrades for longer REs, we limit the maximal RE length to 6 words. We obtain 3 types of initial REs predicted to be distinguishing for a target by Algorithm 2:

- ref_{loc}: 2 word RE, only location words (V_1), Figure 3(a)
- ref_{object}: 4 word RE, location words and object names (V_2), Figure 3(b)
- ref_{att}: 6 word RE, all attributes from the entire vocabulary (V_3), Figure 3(c)

On our test set, this produces distinguishing REs for all targets, except 4 cases for which we use an initial 6 word RE as well.

Reformulations We have several options for generating the reformulation REs (re_2,re_3) - e.g. hedging the object name, extending the RE with more words, removing potentially misleading words, etc. - which are more or less appropriate, depending on the initial RE predicted by Algorithm 2. Therefore, we implemented the following types of installment triples that dynamically extend or reduce the initial RE:

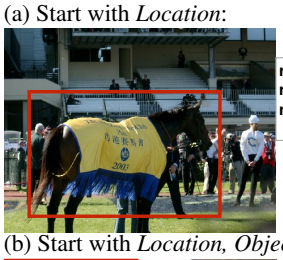
1. (ref_{loc}, ref_{object}, $ref_{object,hedge}$), this corresponds to the pattern in Section 4.2
2. (ref_{object}, $ref_{object,hedge}$,ref_{att})
3. (ref_{att}, $ref_{att,hedge}$,ref_{loc})

Figure 3 shows examples for each triple.

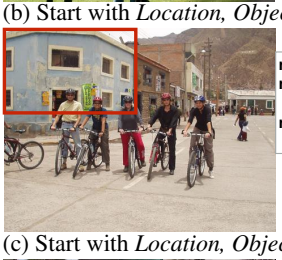
4.4 Human Evaluation

Set-up We use the task-oriented setup from Section 3.4 with 3 trials per image. But instead of presenting the same RE in each trial, the system now updates the phrases according to the RE triples described above. We have recruited 5 players and collected 1200 games, split equally between

(a) Start with *Location*:



(b) Start with *Location, Object Type*:



(c) Start with *Location, Object Type,Other*:

Figure 3: Examples for context-dependent installments

the pattern-based installment (Section 4.2) and the context-dependent installment strategy (Section 4.3). In this evaluation, we only use word classifiers trained on GoogLeNet features.

Results Table 4 shows that even the simple, pattern-based installment system improves the 1st trial success rate compared to the non-interactive baseline (the GoogLeNet-based system from Section 3) and is clearly superior with respect to its overall success and error reduction rate over trials. This suggests that a fair amount of target objects can be identified by users based on very simple, locative REs as semantically inadequate object names are avoided. Another important finding here is the high rate of error reduction during the 2nd and 3rd trial achieved by the installment-based system. In the non-interactive system, users did not have additional cues for repairing their misunderstanding and probably guessed other possible targets in individual, more or less systematic ways. Apparently, even simple strategies for extending and hedging the initially presented RE provide very helpful cues for repairing initial misunderstandings.

As we expected, the pattern-based installment system is clearly improved by our context-dependent approach to generating installments. This systems seems to strike a much better balance between generating simple expressions that avoid

	Success rate/ trial			Error
	1st	2nd	3rd	red.
No install.	41.6	53.4	59.1	29.9
Pattern install.	46.8	69.2	80.9	64.1
Contextual install.	50.5	74.9	86	71.71

Table 4: Human evaluation for installment-based REG systems

inadequate object names on the one and contextually appropriate expressions on the other hand. It improves the pattern-based installments in terms of 1st trial success rate and overall success and error reduction rate.

The finding that installment strategies should be combined with insights from traditional distractor-oriented REG is further corroborated when we compare the success rates on the different subsets of our test set, see Table 5. Thus, the performance of the context-dependent installment system is much more stable on the different subsets than the pattern-based system which has a clear dip in success rate on Set C, which contains target referents with distractors of the same object type. This result suggests that our approach to determine distinguishing REs based purely on predictions of word-based REG (Section 4.3) presents a viable solution for REG on images, where information on distractors is not directly assessable in the low-level representation of the scene.

	Success rate (3rd trial)		
	Set A	Set B	Set C
No install.	57	67.7	53.1
Pattern install.	80.8	84.3	77.5
Contextual install.	86	87.5	84.5

Table 5: Human evaluation on different test sets for installment-based REG systems

Finally, the graph in Figure 4 shows the average success rates over time and provides more evidence for the effectiveness of installments. We observe a clear learning effect in the non-interactive system, meaning that users faced unexpected interpretation problems due to inaccurate expressions, but adapted to the situation to some extent. In contrast, both installment systems have stable performance over time, which indicates that system behaviour is immediately understandable and predictable for human users.

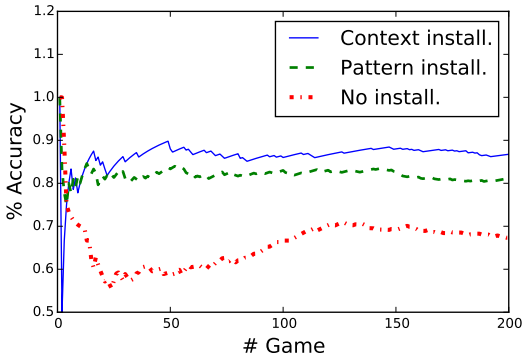


Figure 4: Participants' success rates in object identification over time

5 Discussion and Conclusion

We have presented an REG system that approaches the task as a word selection problem and circumvents manual specification of attributes in symbolic scene representations as required in traditional REG (Krahmer and Van Deemter, 2012), or manual specification of attribute-specific functions that map particular low-level visual features to attributes or words as in (Roy, 2002; Kazemzadeh et al., 2014). This knowledge-lean approach allows us to use automatically learned ConvNet features and obtain a promising baseline that predicts semantically appropriate words based on visual object representations.

We have argued and demonstrated that REG in more realistic settings greatly benefits from a task-oriented, interactive account and should explore principled strategies for repairing and avoiding misunderstandings due to semantically inaccurate REs. In order to achieves this, we have augmented our approach with some manually designed installment strategies. An obvious direction for future work is to automatically induce such a strategy, based on confidence measures that automatically predict the trust-worthiness of a word for an object.

Another extension that we have planned for future work is to implement relational expressions, similar to (Kennington and Schlangen, 2015). Based on relational expressions, we will be able to generate reformulations and installments tailored to the interaction with the user. For instance, a very natural option for installments is to relate the wrong target object clicked on by the user to the intended target, e.g. something like *to the left of that one, the bigger object*.

Acknowledgments

We acknowledge support by the Cluster of Excellence "Cognitive Interaction Technology" (CITEC; EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

Anja Belz and Helen Hastie. 2014. Comparative evaluation and shared tasks for nlg in interactive systems. In Amanda Stent and Srinivas Bangalore, editors, *Natural Language Generation in Interactive Systems*, pages 302–350. Cambridge University Press. Cambridge Books Online.

Xinlei Chen and C Lawrence Zitnick. 2015. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2422–2431.

Herbert H Clark and Deanna Wilkes-Gibbs. 1986. Referring as a collaborative process. *Cognition*, 22(1):1–39.

Alasdair DF Clarke, Micha Elsner, and Hannah Rohde. 2013. Where's wally: the influence of visual salience on referring expression generation. *Frontiers in psychology*, 4.

Robert Dale and Ehud Reiter. 1995. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive Science*, 19(2):233–263.

Jia Deng, W. Dong, Richard Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Nina Dethlefs and Heriberto Cuayáhuitl. 2015. Hierarchical reinforcement learning for situated natural language generation. *Natural Language Engineering*, 21(03):391–435.

David DeVault, Natalia Kariaeva, Anubha Kothari, Iris Oved, and Matthew Stone. 2005. An information-state approach to collaborative reference. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 1–4.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 100–105, Beijing, China, July. Association for Computational Linguistics.

Hugo Jair Escalante, Carlos a. Hernández, Jesus a. Gonzalez, a. López-López, Manuel Montes, Eduardo F. Morales, L. Enrique Sucar, Luis Villaseñor, and Michael Grubinger. 2010. The segmented and annotated IAPR TC-12 benchmark. *Computer Vision and Image Understanding*, 114(4):419–428.

Rui Fang, Malcolm Doering, and Joyce Y. Chai. 2014. Collaborative Models for Referring Expression Generation in Situated Dialogue. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Konstantina Garoufi and Alexander Koller. 2013. Generation of effective referring expressions in situated context. *Language and Cognitive Processes*.

Albert Gatt, Anja Belz, and Eric Kow. 2009. The tuna-reg challenge 2009: Overview and evaluation results. In *Proceedings of the 12th European Workshop on Natural Language Generation*, pages 174–182. Association for Computational Linguistics.

Dimitra Gkatzia, Verena Rieser, Phil Bartie, and William Mackaness. 2015. From the virtual to the real world: Referring to objects in real-world spatial scenes. In *Proceedings of EMNLP 2015*. Association for Computational Linguistics.

Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The IAPR TC-12 benchmark: a new evaluation resource for visual information systems. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2006)*, pages 13–23, Genoa, Italy.

Peter A Heeman and Graeme Hirst. 1995. Collaborating on referring expressions. *Computational linguistics*, 21(3):351–382.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 787–798, Doha, Qatar.

John D Kelleher and Geert-Jan M Kruijff. 2006. Incremental generation of spatial referring expressions in situated dialog. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 1041–1048.

Casey Kennington and David Schlangen. 2015. Simple learning and compositional application of perceptually grounded word meanings for incremental reference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 292–301, Beijing, China, July. Association for Computational Linguistics.

Ruud Koolen and Emiel Krahmer. 2010. The d-tuna corpus: A dutch dataset for the evaluation of referring expression generation algorithms. In *LREC*.

Ruud Koolen, Albert Gatt, Martijn Goudbeek, and Emiel Krahmer. 2011. Factors causing overspecification in definite descriptions. *Journal of Pragmatics*, 43(13):3231–3250.

Emiel Krahmer and Kees Van Deemter. 2012. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218.

Changsong Liu, Rui Fang, and Joyce Y Chai. 2012. Towards mediating shared perceptual basis in situated dialogue. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 140–149. Association for Computational Linguistics.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2015. Generation and comprehension of unambiguous object descriptions. *CoRR*, abs/1511.02283.

Margaret Mitchell, Kees van Deemter, and Ehud Reiter. 2010. Natural reference to objects in a visual domain. In *Proceedings of the 6th international natural language generation conference*, pages 95–104. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Deb K Roy. 2002. Learning visually grounded words and syntax for a scene description task. *Computer Speech & Language*, 16(3):353–385.

David Schlangen, Sina Zarriess, and Casey Kennington. 2016. Resolving references to objects in photographs using the words-as-classifiers model. In *Proceedings of the 54rd Annual Meeting of the Association for Computational Linguistics (ACL 2016)*.

Maria Staudte, Alexander Koller, Konstantina Garoufi, and Matthew W Crocker. 2012. Using listener gaze to augment speech generation in a virtual 3d environment. In *Proceedings of the 34th Annual Conference of the Cognitive Science Society. To appear*.

Laura Stoia, Darla Magdalene Shockley, Donna K Byron, and Eric Fosler-Lussier. 2006. Noun phrase generation for situated dialogs. In *Proceedings of the fourth international natural language generation conference*, pages 81–88. Association for Computational Linguistics.

Kristina Striegnitz, Alexandre Denis, Andrew Gargett, Konstantina Garoufi, Alexander Koller, and Mariët Theune. 2011. Report on the second second challenge on generating instructions in virtual environments (give-2.5). In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 270–279.

Kristina Striegnitz, Hendrik Buschmeier, and Stefan Kopp. 2012. Referring in installments: a corpus study of spoken object references in an interactive virtual environment. In *Proceedings of the Seventh International Natural Language Generation Conference*, pages 12–16.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR 2015*, Boston, MA, USA, June.

Sammie Tarenskeen, Mirjam Broersma, and Bart Geurts. 2015. hand me the yellow stapler or hand me the yellow dress: Colour overspecification depends on object category. page 140.

Kees van Deemter, Ielka van der Sluis, and Albert Gatt. 2006. Building a semantically transparent corpus for the generation of referring expressions. In *Proceedings of the Fourth International Natural Language Generation Conference*, pages 130–132. Association for Computational Linguistics.

Jette Viethen and Robert Dale. 2008. The use of spatial relations in referring expression generation. In *Proceedings of the Fifth International Natural Language Generation Conference*, pages 59–67. Association for Computational Linguistics.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*.

Collective Entity Resolution with Multi-Focal Attention

Amir Globerson[*] and **Nevena Lazic** and **Soumen Chakrabarti**[†] and
Amarnag Subramanya and **Michael Ringgaard** and **Fernando Pereira**
Google, Mountain View CA, USA
gamir@post.tau.ac.il, nevena@google.com, soumen@cse.iitb.ac.in,
{asubram, ringgaard, pereira}@google.com

Abstract

Entity resolution is the task of linking each mention of an entity in text to the corresponding record in a knowledge base (KB). Coherence models for entity resolution encourage all referring expressions in a document to resolve to entities that are related in the KB. We explore attention-like mechanisms for coherence, where the evidence for each candidate is based on a small set of strong relations, rather than relations to all other entities in the document. The rationale is that document-wide support may simply not exist for non-salient entities, or entities not densely connected in the KB. Our proposed system outperforms state-of-the-art systems on the CoNLL 2003, TAC KBP 2010, 2011 and 2012 tasks.

1 Introduction

Entity resolution (ER) is the task of mapping mentions of entities in text to corresponding records in a knowledge base (KB) (Bunescu and Pasca, 2006; Cucerzan, 2007; Kulkarni et al., 2009; Dredze et al., 2010; Hoffart et al., 2011; Hachey et al., 2013). ER is a challenging problem because mentions are often ambiguous on their own, and can only be resolved given appropriate context. For example, the mention `Beirut` may refer to the capital of Lebanon, the band from New Mexico, or a drinking game (Figure 1). Names may also refer to entities that are not in the KB, a problem known as NIL *detection*.

Most ER systems consist of a *mention model*, a *context model*, and a *coherence model* (Milne and Witten, 2008; Cucerzan, 2007; Ratinov et al., 2011; Hoffart et al., 2011; Hachey et al., 2013). The mention model associates each entity with its possible textual representations (also known as aliases or surface forms). The context model helps resolve an ambiguous mention using textual features extracted from the surrounding context. The coherence model, the focus of this work, encourages all mentions to resolve to entities that are related to each other. Relations may be established via the KB, Web links, embeddings, or other resources.

Coherence models often define an objective function that includes local and pairwise candidate scores, where the pairwise scores correspond to some notion of coherence or relation strength.[1] Support for a candidate is typically aggregated over relations to all other entities in the document. One problem with this approach is that it may dilute evidence for entities that are not salient in the document, or not well-connected in the KB. Our work aims to address this issue.

We introduce a novel coherence model with an attention mechanism, where the score for each candidate only depends on a small subset of mentions. Attention has recently been used with considerable empirical success in tasks such as translation (Bahdanau et al., 2014) and image caption generation (Xu et al., 2015). We argue that attention is also desirable for collective ER due to the discussed imbalance in the number of relations for different entities.

Attention models typically have a single focus, implemented using the softmax function. Our model allows each candidate to focus on multiple mentions, and, to implement it, we introduce a novel smooth version of the multi-focus attention

[*] Currently at Tel Aviv University
[†] Currently at IIT Bombay

[1] An exception to this framework are topic models in which a topic may generate both entities and words, e.g., (Kataria et al., 2011; Han and Sun, 2012; Houlsby and Ciaramita, 2014).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 621–631,
Berlin, Germany, August 7-12, 2016.

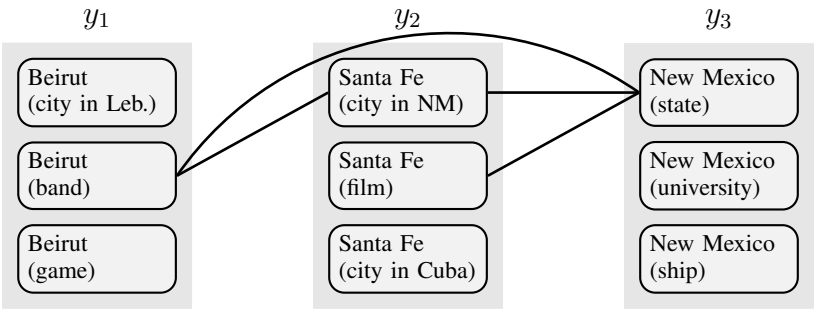


Figure 1: Illustration of the ER problem for three mentions "Beirut", "New Mexico" and "Santa Fe". each mention has three possible disambiguations. Edges link disambiguations that have Wikipedia links between their respective pages.

function, which generalizes soft-max.

Our system uses mention and context models similar to those of Lazic et al. (2015), along with our novel multi-focal attention model to enforce coherence, leading to significant performance improvements on CoNLL 2003 (Hoffart et al., 2011) and TAC KBP 2010–2012 tasks (Ji et al., 2010; Ji et al., 2011; Mayfield et al., 2012). In particular, we achieve a 20% relative reduction in error from Chisholm and Hachey (2015) on CoNLL, and a 22% error reduction from Cucerzan (2012) on TAC 2012. Our contributions thus consist of defining a novel multi-focal attention model and applying it successfully to an entity resolution system.

2 Definitions and notation

We are given a document with n mentions, where each mention i has a set of n_i candidate entities $\mathcal{C}_i = \{c_{i,1}, ..., c_{i,n_i}\}$. The goal is to assign a label $y_i \in \mathcal{C}_i$ to each mention.

Similarly to previous work, our approach to disambiguation relies on local and pairwise candidate scores, which we denote by $s_i(y_i)$ and $s_{ij}(y_i, y_j)$ respectively. The local score is based only on local evidence, such as the mention phrase and textual features, while the pairwise score is based on the relatedness of the two candidates. In Sections 3.2 and 3.3 we discuss how these scores may be parameterized and learned. Many systems (Cucerzan, 2007; Milne and Witten, 2008; Kulkarni et al., 2009) simply hardwire pairwise scores.

Coherence models typically attempt to maximize a *global* objective function that assigns a score to each complete labeling $\mathbf{y} = (y_1, \ldots, y_n)$. An example of such a function is the sum of all singleton and pairwise scores for each label:[2]

$$g(\mathbf{y}) = \sum_i s_i(y_i) + \sum_i \sum_{j:j \neq i} s_{ij}(y_i, y_j). \quad (1)$$

One disadvantage of this approach is that maximizing g corresponds to finding the MAP assignment of a general pairwise Markov random field, and is hence NP hard for the general case (Wainwright and Jordan, 2008). Another limitation is that non-salient entities may be related to very few other entities mentioned in the document, and summing over all mentions may dilute the evidence for such entities. In this paper we explore alternative objectives, relying on attention and tractable inference.

3 Attention model

We now describe our multi-focal attention model. We first introduce the inference approach and optimization objective, and then provide details on how scores are calculated and learned.

3.1 Inference

As noted earlier, the global score function in Eq. (1) is hard to maximize. Here we simplify inference by decomposing the task over mentions, which makes it easy to integrate attention in terms of both inference and learning.

3.1.1 Star model

We start by considering a simple attention-free model in which inference is tractable, which we call a *star model*. For a particular mention i, the star model is a graphical model that contains y_i,

[2] The scores usually depend not only on the labels, but also on the input text. We omit this dependence for brevity.

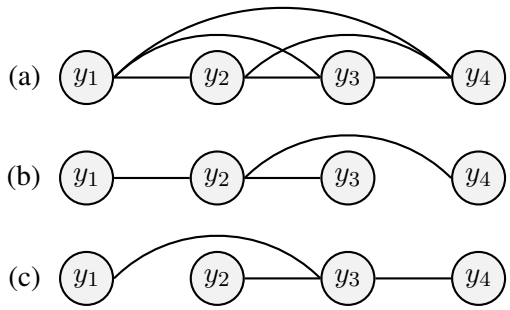


Figure 2: (a) The complete graph corresponding to Eq. (1). (b) A star shaped subgraph corresponding to y_2. This will be used to obtaining the label y_2. (c) The star graph for y_3.

all interactions between y_i and other labels, and no other interactions, as illustrated in Fig. 2.

While the star graph centered at i contains up to n variables, we will only use it to infer the label of mention i. Let $q_{ij}(y_i)$ be the support for label y_i from mention j, defined as follows:

$$q_{ij}(y_i) = \max_{y_j} s_{ij}(y_i, y_j) + s_j(y_j), \qquad (2)$$

and we also define $q_{ii}(y_i) = -\infty$ to simplify notation for later. We define the following score function for mention i:

$$f_i(y_i) = s_i(y_i) + \sum_{j:j\neq i} q_{ij}(y_i) \qquad (3)$$

and predict the label $y_i = \arg\max_y f_i(y)$.

Due to the structure of the star graph, inference is easy and can be done in $O(nC^2)$, where C is the maximum number of candidates. A similar decomposition has previously been used in the context of approximate learning for structured prediction (Sontag et al., 2011). Note that we do not view this approach as an approximation to the global problem, but rather as our inference procedure.

3.1.2 Adding attention

The score function in Eq. (3) aggregates pairwise scores for each label y_i over all mentions. In this section, we restrict this to only consider K mentions with the strongest relations to y_i.[3] Let $\text{amx}_K(\mathbf{z})$ be the sum of the largest K values in the vector $\mathbf{z} = (z_1, \ldots, z_n)$. For each label y_i, we redefine the score function to be

$$f_i(y_i) = s_i(y_i) + \text{amx}_K(\mathbf{q}_i(y_i)), \qquad (4)$$

where $\mathbf{q}_i(y_i) = (q_{i1}(y_i), \ldots, q_{in}(y_i))$ and $q_{ij}(y_i)$ is as defined in Eq. (2). The inference rule is again $y_i = \arg\max_y f_i(y)$, and the computational cost is $O(nC^2 + n \log n)$ since sorting is required.[4]

3.1.3 Soft attention

Previous work on attention has shown that it is advantageous to use a soft form of attention, where the level of attention is not zero or one, but can rather take intermediate values. Existing attention models focus on a single object, such as a single word (Bahdanau et al., 2014) or a single image window (Xu et al., 2015). In such models, it is natural to change the max function in the attention operator to a soft-max. In our case, the attention beam contains K elements, and we require a different notion of a soft-max, which we develop below.

To obtain a soft version of the function $\text{amx}_K(\mathbf{z})$, we first use an alternative definition. Denote by $\mathcal{S}$ the set $\mathbf{u} = (u_1, \ldots, u_n)$ such that $0 \leq u_i \leq 1$ and $\sum_i u_i = K$. Then $\text{amx}_K(\mathbf{z})$ is equivalent to the optimization problem:

$$\max_{\cdot \mathbf{u} \in \mathcal{S}} \mathbf{z} \cdot \mathbf{u} \qquad (5)$$

The optimization problem above is a linear program, whose solution is the sum of top K elements of $\mathbf{z}$ as required. This follows since the optimal u_i can easily be shown to attain only integral values.

Given this optimization view of $\text{amx}_K(\mathbf{z})$ it is natural to smooth it (Nesterov, 2005) by adding a non-linearity to the optimization. Since the variables are non-negative, one possible choice is an entropy-like regularizer. We shall see that this choice results in a closed form solution, and also recovers the standard soft-max case for $K = 1$. Consider the optimization problem:

$$\text{smx}_K(\mathbf{z}) = \max_{\mathbf{u}\in\mathcal{S}} \sum_i z_i u_i - \beta^{-1} \sum_i u_i \log u_i, \qquad (6)$$

where β is a tuned hyperparameter.[5] The following proposition provides a closed form solution for smx_K, as well as its gradient.

Proposition 3.1. *Assume w.l.o.g. that* $\mathbf{z}$ *is sorted such that* $z_1 \geq \ldots \geq z_n$. *Denote by* R *the maximum index* $r \in \{1, \ldots, K-1\}$ *such that:*

$$z_r \geq \beta^{-1} \log \frac{\sum_{j=r+1}^{n} \exp(\beta z_j)}{K - r} \qquad (7)$$

[3] It is possible to relax this to allow *up to* K relations, but we focus on exactly K for simplicity.

[4] Note that if $K < \log n$, we spend only nK instead of $n \log n$ time.

[5] Note that $-\sum_i u_i \log u_i$ is different from the entropy function since variables u_i sum to K and not to 1.

If this doesn't hold for any r, set $R = 0$. Then:

$$smx_K(\mathbf{z}) = \sum_{j=1}^{R} z_j + \frac{K-R}{\beta} \log \frac{\sum_{j=R+1}^{n} \exp(\beta z_j)}{K-R} \tag{8}$$

The function $smx_K(\mathbf{z})$ is differentiable with a gradient $\boldsymbol{v}$ given by:

$$v_i = \left\{ \begin{array}{ll} 1 & 1 \le i \le R \\ (K-R)\frac{\exp(\beta z_i)}{\sum_{j=R+1}^{n} \exp(\beta z_j)} & R < i \le n \end{array} \right\} \tag{9}$$

Proof is provided in the appendix.

As noted, $K = 1$ recovers the standard soft-max function.[6] As $\beta \to \infty$, smx_K will approach the sum of the top K elements as expected. For finite β we have a soft version of amx_K.

Our soft attention based model will therefore consider the soft-variant of Eq. (4):

$$f_i(y_i) = s_i(y_i) + \text{smx}_K(\mathbf{q}_i(y_i)) \,, \tag{10}$$

and maximize $f(y_i)$ to obtain the label.

3.2 Score parameterization

Thus far we assumed the singleton and pairwise scores were given. We next discuss how to parameterize and learn these scores. As in other structured prediction work, we will assume that the scores are functions of the features of the input x and labels. Specifically, denote a set of singleton features for mention i and label y_i by $\boldsymbol{\phi}_i^s(x, y_i) \in \mathbb{R}^{n_s}$ and a set of pairwise features for mentions i and j and their labels by $\boldsymbol{\phi}_{ij}^p(x, y_i, y_j) \in \mathbb{R}^{n_p}$. Then the model has two sets of weights $\boldsymbol{w}_s$ and $\boldsymbol{w}_p$ and the scores are obtained as a linear combination of the features. Namely:[7]

$$\begin{aligned} s_i(y_i; \boldsymbol{w}_s) &= \boldsymbol{w}_s \cdot \boldsymbol{\phi}_i^s(x, y_i) \\ s_{ij}(y_i, y_j; \boldsymbol{w}_p) &= \boldsymbol{w}_p \cdot \boldsymbol{\phi}_{ij}^p(x, y_i, y_j) \,, \end{aligned}$$

where we have explicitly denoted the dependence of the scores on the weight vectors. See Sec. 6.2.2 for details on how the features are chosen. It is of course possible to consider non-linear alternatives for the score function, as in recent deep learning parsing models (Chen and Manning, 2014; Weiss et al., 2015), but we focus on the linear case for simplicity.

3.3 Parameter learning

The parameters $\boldsymbol{w}_s, \boldsymbol{w}_p$ are learned from labeled data, as explained next. Since inference decomposes over mentions, we use a simple hinge loss for each mention. Denote by y_i^* the ground truth label for mention i, and let $\mathbf{s}_i(y_i) \equiv (s_{i1}(y_i), \ldots, s_{in}(y_i))$. Then the hinge loss for mention i is:

$$\begin{aligned} L_i \;=\; & \max_{y_i} [s_i(y_i) + \text{smx}_K(\mathbf{s}_i(y_i)) \\ & - s_i(y_i^*) - \text{smx}_K(\mathbf{s}_i(y_i^*)) + \Delta(y_i, y_i^*)] \end{aligned}$$

where $\Delta(y_i, y_i^*)$ is zero if $y_i = y_i^*$ and one otherwise. If there are unlabeled mentions in the training data, we add those to the star graph, and maximize over the unknown labels in the positive and negative part of the hinge loss. The overall loss is simply the sum of losses for all the mentions, plus ℓ_2 regularization over $\boldsymbol{w}_s, \boldsymbol{w}_p$. We minimize the loss using AdaGrad (Duchi et al., 2011) with learning rate $\eta = 0.1$.

4 Single-link model

To motivate our modeling choices of using multi-focal attention and decomposed inference, we additionally consider a simple baseline model with single-focus attention and global inference. In this approach, which we name *single-link*, each mention i attends to exactly one other mention that maximizes the pairwise relation score. The corresponding objective can be written as

$$g^{SL}(\mathbf{y}) = \sum_i \left(s_i(y_i) + \max_j s_{ij}(y_i, y_j) \right) \tag{11}$$

where $s_{ij}(y_i, y_j) = -\infty$ if there is no relation between y_i and y_j, and we set $s_{ii}(y_i, y_i) = 0$.

While exact inference in this model remains intractable, we can find approximate solutions using max-sum belief propagation (Kschischang et al., 2001). As a reminder, max-sum is an iterative algorithm for MAP inference which can be described in terms of messages sent from model factors $g_a(\mathbf{y}_a)$ to each of their variables $y \in \mathbf{y}_a$. At convergence, each variable is assigned to the value that maximizes *belief* $b(y)$, defined as the sum of incoming messages. The message updates

[6] When we refer to the *soft-max function*, we mean the function $\beta^{-1} \log \sum \exp(\beta a_i)$, which is an often used differentiable convex upper bound of the max function (e.g., see (Gimpel and Smith, 2010)). Soft-max sometimes also refers to the activation function $\frac{\exp(a_i)}{\sum_j \exp(a_j)}$. The latter is in fact the gradient of the former (for $\beta = 1$).

[7] We again omit the dependence of the scores on the input x for brevity.

have the following form:

$$\mu_{g_a \to Y}(y) = \max_{\mathbf{y}_a \backslash y} \left[g_a(\mathbf{y}_a) + \sum_{j \neq i} q_j^{\backslash a}(y_j) \right] \quad (12)$$

where $q_j^{\backslash a}(y_j)$ is the sum of all messages to y_j except the one from factor g_a. While the single-link model contains high-order factors over n variables, computing the messages from these factors is tractable and requires sorting.

5 Related work

Ji (2016) and Ling et al. (2015) provide summaries of recent ER research. Here we review work related to the three main facets of our approach.

5.1 Coherence scores

Several systems (Milne and Witten, 2008; Kulkarni et al., 2009; Hoffart et al., 2011) use the "Milne and Witten" measure for relatedness between a pair of entities, which is based on the number of Wikipedia articles citing each entity page, and the number of articles citing both; Cucerzan (2007) has also relied on the Wikipedia category structure. Internal links from one entity page to another in Wikipedia also provide direct evidence of relatedness between them. Another (possibly more noisy) source of information are Web pages containing links (Singh et al., 2012) to Wikipedia pages of both entities. Such links have been used in several recent systems (Cheng and Roth, 2013; Chisholm and Hachey, 2015). Yamada et al. (2016) train embedding vectors for entities, and use them to define similarities.

5.2 Collective inference for ER

Optimizing most global coherence objectives is intractable. Milne and Witten (2008) and Ferragina and Scaiella (2010) decompose the problem over mentions and select the candidate that maximizes their relatedness score, which includes relations to all other mentions. Hoffart et al. (2011) use an iterative heuristic to remove unpromising mention-entity edges. Cucerzan (2007) creates a relation vector for each candidate, and disambiguates each entity to the candidate whose vector is most similar to the aggregate (which includes both correct and incorrect labels). Cheng and Roth (2013) use an integer linear program solver and Kulkarni et al. (2009) use a convex relaxation. Ratinov et al. (2011) use relation scores as features in a ranking SVM. Belief propagation without attention has been used by Ganea et al. (2015). Personalized PageRank (PPR) (Jeh and Widom, 2003) is another tractable alternative, adopted by several recent systems (Han and Sun, 2011; He et al., 2013; Alhelbawy and Gaizauskas, 2014; Pershina et al., 2015). Laplacian smoothing (Huang et al., 2014) is closely related.

5.3 Attention models

Attention models have shown great promise in several applications, including machine translation (Bahdanau et al., 2014) and image caption generation (Xu et al., 2015). We address a new application of attention, and introduce a significantly different attention mechanism, which allows each variable to focus on *multiple* objects. We develop a novel smooth version of the multi-focus attention function, which generalizes the single focus softmax-function. While some existing entity resolution systems (Jin et al., 2014; Lazic et al., 2015) may be viewed as having attention mechanisms, these are intended for single textual features and not readily extensible to structured inference.

6 Experiments

6.1 Evaluation data

CoNLL: The CoNLL dataset (Hoffart et al., 2011) contains 1393 articles with about 34K mentions, and the standard performance metric is mention-averaged accuracy. The documents are partitioned into train, test-a and test-b. Like most authors, we report performance on the 231 test-b documents with 4483 linkable mentions.

TAC KBP: The TAC KBP 2010, 2011, and 2012 evaluation datasets (Ji et al., 2010; Ji et al., 2011; Mayfield et al., 2012) include 2250, 2250, and 2226 mentions respectively, of which roughly half are linkable to the reference KB. The competition evaluation includes NIL entities; participants are required to cluster NIL mentions across documents so that all mentions of each unknown entity are assigned a unique identifier. For these datasets, we report in-KB accuracy, overall accuracy (with all NILs in one cluster), and the competition metric $B^{3+}F_1$ which evaluates NIL clustering.

6.2 Experimental setup

6.2.1 KB and entity aliases

Our KB is derived from the Wikipedia subset of Freebase (Bollacker et al., 2008), with about 4M

entities. To obtain our mention prior (the probability of candidate entities given a mention), we collect alias counts from Wikipedia page titles (including redirects and disambiguation pages), Freebase aliases, and Wikipedia anchor text. 99.31% of CoNLL test-b mentions are covered by the KB, and 96.19% include the gold entity in the candidates.

We optionally use the mapping from aliases to candidate entities released by Hoffart et al. (2011), obtained by extending the "means" tables of YAGO (Hoffart et al., 2013). When released, it had 100% mention and gold recall on CoNLL, i.e. every annotated mention could be mapped to at least one entity, and the set of entities included the gold entity. However, changes in canonical Wikipedia URLs, accented characters and unicode usually result in mention losses over time, as not all URLs can be mapped to the KB (Hasibi et al., 2016, Sec. 4).

For CoNLL only, we experiment with a third alias-entity mapping derived from Hoffart et al. (2011) by Pershina et al. (2015); we call it "HP". It is not known how candidates were pruned, but it has high recall and very low ambiguity: only 12.6 on CoNLL test-b, compared to 22.34 in our KB and 65.9 in YAGO. Unsurprisingly, using only this source of aliases results in high accuracy on CoNLL (Pershina et al., 2015; Yamada et al., 2016).

Table 1 lists the statistics of the three alias-entity mappings and some of their combinations on the CoNLL test-b dataset. Table 2 provides the same statistics on the TAC KBP datasets (restricted to non-NIL mentions) for the of the YAGO+KB alias-entity mapping.

Alias map	Mention recall	Gold recall	Uniq. %	Avg. ambig.
KB	99.31	96.19	17.93	22.3
YAGO	97.17	96.30	15.50	65.9
+KB	99.84	99.51	16.28	73.6
HP	99.87	99.84	17.98	12.6
+KB	99.87	99.87	16.40	28.7
All	99.87	99.87	15.37	78.7

Table 1: Alias-entity map statistics on CoNLL test-b, 4483 gold mentions. Mention recall is the percentage of mentions with at least one known entity; gold recall is the percentage of mentions where the gold entity was included in the candidates. Unique aliases map to exactly one entity. The last column shows the number of candidates averaged over test-b mentions.

Dataset	Mention recall	Gold recall	Uniq. %	Avg. ambig.
TAC 2010	98.14	93.04	22.45	45.34
TAC 2011	98.40	89.23	27.82	49.13
TAC 2012	97.36	87.83	20.00	68.93

Table 2: YAGO+KB alias-entity map statistics on the TAC KBP datasets, restricted to non-NIL mentions.

6.2.2 Local and pairwise scores

Our baseline system is similar in design and accuracy to Plato (Lazic et al., 2015). Given the referent phrase m_i and textual context features $\mathbf{b}_i$, it computes the probability of a candidate entity as $p_i(c) \propto p(c|m_i)p(\mathbf{b}_i|c)$. The system resolves mentions independently and does not have an explicit coherence model; however, it does capture some coherence information indirectly as referent phrases are included as string context features. We experiment with several versions of the mention prior $p(c|m_i)$ as described in the previous section.

Scores for single-link model: In the single-link model, we simply set the local score for mention i and candidate c to $s_i(c) = \ln \frac{p_i(c)}{1-p_i(c)}$, so that likely candidates get positive scores. We set the pairwise score between two candidates heuristically to $s_{ij}(y_i, y_j) = \ln o(y_i, y_j) + 2.3$, where $o(y_i, y_j)$ is the number of outlinks from the Wikipedia page of y_i to the page of y_j. We consider up to three candidates for each mention for CONLL, and ten for TAC; if the baseline probability of the top candidate exceeds 0.9, we only consider the top candidate. Including more candidates did not make a difference in performance, as additional candidates had low baseline scores and were almost never chosen in practice.

Scores for attention model: Local features $\boldsymbol{\phi}_i^s(x, y_i)$ for the attention model are derived from $p_i(c)$. As the attention models have no probabilistic interpretation, we inject as features $\log p_i(c)$ and $\log(1 - p_i(c))$. We set $\log 0 = 0$ by convention, and handle the case where $\log$ is undefined by introducing two additional binary indicator features for $p_i(c) = 0$ and $p_i(c) = 1$.

Edge features $\boldsymbol{\phi}_{ij}^p$ are set based on three sources of information: (1) number of Freebase relations

System	Alias map	In-KB acc. %
Lazic (2015)	N/A	86.4
Our baseline	KB	87.9
Single link	KB	88.2
Attention	KB	**89.5**
Chisholm (2015)	YAGO	88.7
Ganea (2015)	YAGO	87.6
Our baseline	KB+YAGO	85.2
Single link	KB+YAGO	86.6
Attention	KB+YAGO	**91.0**
Our baseline	KB+HP	89.9
Single link	KB+HP	89.9
Attention	KB+HP	**91.7**
Our baseline	KB+HP*	91.9
Single link	KB+HP*	92.1
Attention	KB+HP*	92.7
Pershina (2015)	HP	91.8
Yamada (2016)	HP	**93.1**

Table 3: CoNLL test-b evaluation for recent competitive systems and our models, using different alias-entity maps. "KB+HP*" means we train and score entities using KB+HP, but output entities only in HP.

between y_i and y_j, (2) number of hyperlinks between Wikipedia pages of y_i and y_j (in either direction), and (3) number of mentions of y_i on the Wikipedia page of y_j and vice versa, after annotating Wikipedia with our baseline resolver. We cap each count to five and encode it using five binary indicator features, where the j^{th} feature is set to 1 if the count is j and 0 otherwise. Additionally, for each count c we add a feature $\log(1+c)$. We also added a binary feature which is one if $y_i = y_j$.

We train the scores for the attention model on the 946 CoNLL train documents for CoNLL, and on the TAC 2009 evaluation and TAC 2010 training documents for TAC.

6.3 Results

CoNLL: Table 3 compares our models to recent competitive systems on CoNLL test-b in terms of mention-averaged (micro) accuracy. We also note the alias-entity map used in each system, as the corresponding gold recall is an upper bound on accuracy, and alias ambiguity determines the difficulty of the task. Therefore performance is not strictly comparable between maps.

Our baseline is slightly better than Lazic et al. (2015), but degrades after adding YAGO aliases which increase ambiguity. The attention model provides a substantial gain over the baseline, and outperforms Chisholm and Hachey (2015) by 2.3% in absolute accuracy.

The extremely low ambiguity (Tab. 1) of the HP alias mapping, coupled with guaranteed gold recall, makes the task too easy to be considered a realistic benchmark. Although we match Pershina et al. (2015) using KB+HP, for completeness, we provide the performance of our system with candidate entities restricted to those in HP (KB+HP*), but this is not equivalent to using only HP during training and inference. With KB+HP*, we outperform Pershina et al. (2015), and are competitive with recent unpublished work by Yamada et al. (2016), which uses entity and word embeddings. Including embeddings as features in our system may lead to further gains.

TAC KBP: Table 4 shows our results for the TAC KBP 2010, 2011, and 2012 evaluation datasets, where we used the KB+YAGO entity-alias map for all our experiments. To compute NIL clusters required for $B^3 + F_1$, we simply rely on the fact that our KB is larger than the TAC reference KB, similarly to previous work. We assign a unique NIL label to all mentions of an entity that is in our KB but not in TAC. For mentions that cannot be linked to our KB, we simply use the mention string as the NIL identifier. Once again, our attention models improve the performance over the baseline system in nearly all experiments, with multi-focus attention outperforming single-link. Compared to prior work, we achieve competitive performance on TAC 2010 and the best results to date on TAC 2011 and TAC 2012.

Table 5 shows two examples from the TAC 2011 dataset in which our multi-focus attention model improves over the baseline, along with the focus mentions in the document.

6.4 Effect of K and β on attention

We set the size of the multi-focus attention beam K based on accuracy on CoNLL test-a (for CoNLL) and training accuracy (for TAC). Fig. 3 shows the effect of K on the performance on CoNLL test-a dataset. Performance peaks for $K = 6$, with a sharp decrease after $K = 10$. This validates our central premise: all-pairs label coupling may hurt accuracy.

In Sec. 3.1.3 we proposed an extension of softmax smoothing to the K attention case. In our

System	In-KB acc.(%)	Overall acc.(%)	$B^{3+}F_1$
Chisholm (2015)	80.7	-	-
Ling (2015)	-	**88.8**	-
Yamada (2016)	85.2	-	-
Our baseline	84.5	87.6	83.0
Single link	84.3	87.5	82.8
Attention	**87.2**	88.7	**84.4**
Cucerzan (2011)	-	86.8	84.1
Lazic (2015)	79.3	86.5	84.0
Ling (2015)	-	-	81.6
Our baseline	81.5	86.8	84.3
Single link	82.8	87.3	84.9
Attention	**84.3**	**88.0**	**85.6**
Cucerzan (2012) R1	72.0	76.2	72.1
Cucerzan (2012) R3	71.2	76.6	73.0
Lazic (2015)	74.2	76.6	71.2
Ling (2015)	-	-	66.7
Our baseline	78.8	80.3	76.9
Single link	79.7	80.7	77.3
Attention	**82.4**	**81.9**	**78.9**

Table 4: Results on the TAC 2010 (top), TAC 2011 (middle), and TAC 2012 bottom evaluation datasets.

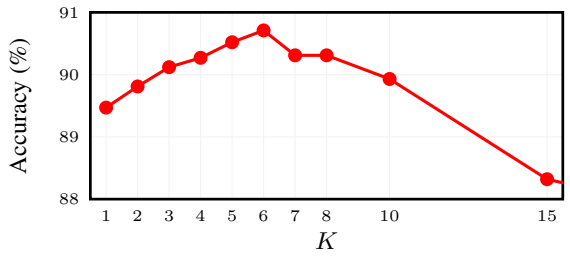


Figure 3: Effect of parameter K on entity linking accuracy. Trained on CoNLL train and tested on CoNLL test-a.

experiments we cross-validated over a wide range of β values, including $\beta = \infty$ which corresponds to taking the exact sum of K largest values. We found that the optimal value in most cases was large: $\beta = 10, 100$, or even ∞. This suggests that a *hard* attention model, where exactly K mentions are picked is adequate in the current settings.

7 Conclusion

We have described an attention-based approach to collective entity resolution, motivated by the observation that a non-salient entity in a long document may only have relations to a small subset of other entities. We explored two approaches to attention: a multi-focus attention model with tractable inference decomposed over mentions, and a single-focus model with global inference implemented using belief propagation. Our empirical results show that the methods results in significant performance gains across several benchmarks.

Experiments in varying the size of the attention beam K in the star-shaped model suggest that multi-focus attention is beneficial. It is of course possible to extend the global single-link model to the multi-focus case, by modifying the model factors and resulting messages. However, the simplicity of the star-shaped model, its empirical effectiveness, and ease of learning parameters make it an attractive approach for easily incorporating attention into existing resolution models. The model can also readily be applied to other structured prediction problems in language processing, such as selecting antecedents in coreference resolution.

Deep learning has recently been used in mutliple NLP applications, including parsing (Chen and Manning, 2014) and translation (Bahdanau et al., 2014). Learning the local and pairwise scores in our model using a deep architecture rather than a linear model would likely lead to performance improvements. The star-shaped model is particularly amenable to this architecture, as it can be implemented via a feed-forward sequence of operations (including sorting, which can be implemented with soft-max gates).

Finally, one may consider a more elaborate model in which attention depends on the current state of the system; for example, the state can summarize the mention context. The dynamics of the underlying state can be modeled by recurrent neural networks or LSTMs (Bahdanau et al., 2014).

In conclusion, we have shown that attention is an effective mechanism for improving entity resolution models, and that it can be implemented via a simple inference mechanism, where model parameters can be easily learned.

8 Proof of Proposition 3.1

Begin with the optimization problem in Eq. (6). Introduce the following Lagrange multipliers: λ for the $\sum_i u_i = K$ constraint, and $\alpha_i \geq 0$ for the $u_i \leq 1$ constraint. We can ignore the $u_i \geq 0$ constraint, as it will turn out to be satisfied. Denote

Sentence with mention	Entity	Attn. focus mentions
Caroline has dropped her name from consideration for the seat that Hillary has left vacant.	base: Caroline (given name) attn: Caroline Kennedy	Democratic Party New York Robert Kennedy
Chris Johnson had just 13 tackles last season, and the Raiders currently have have 11 defensive backs on their roster.	base: Chris Johnson (running back) attn: Chris Johnson (cornerback)	Oakland Raiders Oakland Raiders Oakland Raiders

Table 5: Examples of gains by our algorithm, showing the resolved mention, the entities it resolves to in the baseline and the attention models, and the mentions in the document that are attended to (here $K = 3$). In the first example, the baseline labels the mention "Caroline" as the given name, whereas the attention model attends to mentions that identify it as the diplomat Caroline Kennedy. In the second example, both models resolve "Chris Johnson" to football players, but the attention model finds the correct one by attending to three mentions of his former team, the Oakland Raiders.

the corresponding Lagrangian by $L(\mathbf{u}, \lambda, \alpha)$. We will show the result by using the dual $g(\lambda, \alpha) = \max_{\mathbf{u}} L(\mathbf{u}, \lambda, \alpha)$ and the fact that the solution of Eq. (6) is $\min_{\lambda,\alpha} g(\lambda, \alpha)$.

Maximizing L with respect to u_i yields:

$$u_i = e^{\beta z_i - 1 + \beta\lambda - \beta\alpha_i} \tag{13}$$

From this we can obtain the convex dual $g(\lambda, \alpha)$, and after minimizing over λ we arrive at:

$$g(\alpha) = K\beta^{-1} \log \frac{\sum_i e^{\beta z_i - \beta\alpha_i}}{K} + \sum_i \alpha_i \tag{14}$$

Next, we maximize the above with respect to $\alpha \geq 0$. Introduce Lagrange multipliers γ_i for the constraint $\alpha_i \geq 0$ and the corresponding Lagrangian $\bar{L}(\alpha, \gamma)$. We propose a solution for α, γ and show that it satisfies the KKT conditions. Minimizing $\bar{L}$ wrt α we can characterize the optimal γ as:

$$\gamma_i = -K \frac{e^{\beta z_i - \beta\alpha_i}}{\sum_i e^{\beta z_i - \beta\alpha_i}} + 1 \tag{15}$$

Set α_i as follows:

$$\alpha_i = \begin{cases} z_i - \frac{1}{\beta} \log \frac{\sum_{i=R+1}^{n} e^{\beta z_i}}{K-R} & 1 \leq i \leq R \\ 0 & R < i \leq n \end{cases} \tag{16}$$

It can now be confirmed that the α, γ from Equations 16 and 15 satisfy the KKT conditions. Plugging the α value into $g(\alpha)$ yields the solution in the proposition. Differentiability follows from Nesterov (2005) and the gradient is u_i in Eq. (13).

References

[Alhelbawy and Gaizauskas2014] Ayman Alhelbawy and Robert Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. In *Proc. 52nd Annual Meeting of the Association for Computational Linguistics*, ACL 14, pages 75–80.

[Bahdanau et al.2014] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bollacker et al.2008] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250. ACM.

[Bunescu and Pasca2006] Razvan C. Bunescu and Marius Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proc. 11th Conference of the European Chapter of the Association for Computational Linguistics*, EACL 06.

[Chen and Manning2014] Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750.

[Cheng and Roth2013] Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *EMNLP Conference*, pages 1787–1796.

[Chisholm and Hachey2015] Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics*, 3:145–156.

[Cucerzan2007] Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proc. of EMNLP-CoNLL 2007*, pages 708–716.

[Cucerzan2012] Silviu Cucerzan. 2012. The MSR system for entity linking at TAC 2012. In *In Proc. of the Text Analysis Conference*, TAC 12.

[Dredze et al.2010] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. 2010.

Entity disambiguation for knowledge base population. In *Proc. of the 23rd International Conference on Computational Linguistics*, COLING 10, pages 277–285.

[Duchi et al.2011] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

[Ferragina and Scaiella2010] Paolo Ferragina and Ugo Scaiella. 2010. TAGME: on-the-fly annotation of short text fragments (by Wikipedia entities). In *Proc. of the 19th ACM International Conference on Information Knowledge and Management*, CIKM 10, pages 1625–1628. ACM.

[Ganea et al.2015] Octavian-Eugen Ganea, Marina Horlescu, Aurelien Lucchi, Carsten Eickhoff, and Thomas Hofmann. 2015. Probabilistic bag-of-hyperlinks model for entity linking. *arXiv preprint arXiv:1509.02301*.

[Gimpel and Smith2010] Kevin Gimpel and Noah A Smith. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736. Association for Computational Linguistics.

[Hachey et al.2013] Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with Wikipedia. *Artificial Intelligence*, 194(0):130 – 150.

[Han and Sun2011] Xianpei Han and Le Sun. 2011. A generative entity-mention model for linking entities with knowledge base. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, volume 1 of *ACLHLT 11*. ACL.

[Han and Sun2012] Xianpei Han and Le Sun. 2012. An entity-topic model for entity linking. In *EMNLP-CoNLL*, pages 105–115.

[Hasibi et al.2016] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. On the reproducibility of the TagMe entity linking system. In *Advances in Information Retrieval*, pages 436–449. Springer.

[He et al.2013] Zhengyan He, Shujie Liu, Mu Li, Ming Zhou, Longkai Zhang, and Houfeng Wang. 2013. Learning entity representation for entity disambiguation. In *Proc. of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL 13, pages 30–34.

[Hoffart et al.2011] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust disambiguation of named entities in text. In *Proc. of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP11. ACL.

[Hoffart et al.2013] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

[Houlsby and Ciaramita2014] Neil Houlsby and Massimiliano Ciaramita. 2014. A scalable Gibbs sampler for probabilistic entity linking. In *Advances in Information Retrieval*, pages 335–346. Springer.

[Huang et al.2014] Hongzhao Huang, Yunbo Cao, Xiaojiang Huang, Heng Ji, and Chin-Yew Lin. 2014. Collective tweet wikification based on semi-supervised graph regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 380–390, Baltimore, Maryland, June. Association for Computational Linguistics.

[Jeh and Widom2003] Glen Jeh and Jennifer Widom. 2003. Scaling personalized web search. In *Proceedings of the 12th international conference on World Wide Web*, pages 271–279. ACM.

[Ji et al.2010] Heng Ji, Ralph Grishman, Hoa Trang Dang, Kira Griffitt, and Joe Ellis. 2010. Overview of the TAC 2010 knowledge base population track. In *Proc. of the 3rd Text Analysis Conference*, TAC 10.

[Ji et al.2011] Heng Ji, Ralph Grishman, and Hoa Trang Dang. 2011. Overview of the TAC 2011 knowledge base population track. In *Proc. of the 4th Text Analysis Conference*, TAC 11.

[Ji2016] Heng Ji. 2016. Entity discovery and linking and Wikification reading list. Online. http://nlp.cs.rpi.edu/kbp/2014/elreading.html.

[Jin et al.2014] Yuzhe Jin, Emre Kiciman, Kuansan Wang, and Ricky Loynd. 2014. Entity linking at the tail: sparse signals, unknown entities, and phrase models. In *Proc. of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 453–462, New York, NY, USA. ACM.

[Kataria et al.2011] Saurabh S. Kataria, Krishnan S. Kumar, Rajeev R. Rastogi, Prithviraj Sen, and Srinivasan H. Sengamedu. 2011. Entity disambiguation with hierarchical topic models. In *Proc. of the 17th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1037–1045. ACM.

[Kschischang et al.2001] Frank R Kschischang, Brendan J Frey, and Hans-Andrea Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519.

[Kulkarni et al.2009] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, and Soumen Chakrabarti. 2009. Collective annotation of Wikipedia entities in web text. In *Proc. of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 457–466. ACM.

[Lazic et al.2015] Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

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

[Mayfield et al.2012] James Mayfield, Javier Artiles, and Hoa Trang Dang. 2012. Overview of the TAC 2012 knowledge base population track. In *Proc. of the 5th Text Analysis Conference*, TAC 12.

[Milne and Witten2008] David N. Milne and Ian H. Witten. 2008. Learning to link with Wikipedia. In *Proc. of the 17th ACM Conference on Information and Knowledge Management*, CIKM 07, pages 509–518.

[Nesterov2005] Yu Nesterov. 2005. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152.

[Pershina et al.2015] Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized Page Rank for named entity disambiguation. In *Proc. 2015 Annual Conference of the North American Chapter of the ACL*, NAACL HLT 14, pages 238–243.

[Ratinov et al.2011] Lev-Arie Ratinov, Dan Roth, Doug Downey, and Mike Anderson. 2011. Local and global algorithms for disambiguation to Wikipedia. In *Proc. of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, ACLHLT 11, pages 1375–1384. ACL.

[Singh et al.2012] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2012. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst.

[Sontag et al.2011] D. Sontag, O. Meshi, T. Jaakkola, and A. Globerson. 2011. More data means less inference: A pseudo-max approach to structured learning. In R. Zemel and J. Shawe-Taylor, editors, *Advances in Neural Information Processing Systems 23*, pages 2181–2189. MIT Press, Cambridge, MA.

[Wainwright and Jordan2008] Martin J Wainwright and Michael I Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305.

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

[Xu et al.2015] Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

[Yamada et al.2016] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2016. Joint learning of the embedding of words and entities for named entity disambiguation. *CoRR*, abs/1601.01343.

Which Coreference Evaluation Metric Do You Trust? A Proposal for a Link-based Entity Aware Metric

Nafise Sadat Moosavi and **Michael Strube**
Heidelberg Institute for Theoretical Studies gGmbH
Schloss-Wolfsbrunnenweg 35
69118 Heidelberg, Germany
{nafise.moosavi|michael.strube}@h-its.org

Abstract

Interpretability and discriminative power are the two most basic requirements for an evaluation metric. In this paper, we report the *mention identification effect* in the B^3, *CEAF*, and *BLANC* coreference evaluation metrics that makes it impossible to interpret their results properly. The only metric which is insensitive to this flaw is *MUC*, which, however, is known to be the least discriminative metric. It is a known fact that none of the current metrics are reliable. The common practice for ranking coreference resolvers is to use the average of three different metrics. However, one cannot expect to obtain a reliable score by averaging three unreliable metrics. We propose LEA, a Link-based Entity-Aware evaluation metric that is designed to overcome the shortcomings of the current evaluation metrics. *LEA* is available as branch `LEA-scorer` in the reference implementation of the official CoNLL scorer.

1 Introduction

There exists a variety of models (e.g. pairwise, entity-based, and ranking) and feature sets (e.g. string match, lexical, syntactic, and semantic) to be used in coreference resolution. There is no known formal way to prove which coreference model is superior to the others and which set of features is more beneficial/less useful in coreference resolution. The only way to compare different models, features or implementations of coreference resolvers is to compare the values of the existing coreference resolution evaluation metrics. By comparing the evaluation scores, we determine which system performs best, which model suits coreference resolution better, and which feature set is useful for improving the recall or precision of a coreference resolver. Therefore, evaluation metrics play an important role in the advancement of the underlying technology. It is imperative for the evaluation metrics to be reliable. However, it is not a trivial task to score output entities with various kinds of coreference errors.

Several evaluation metrics have been introduced for coreference resolution (Vilain et al., 1995; Bagga and Baldwin, 1998; Luo, 2005; Recasens and Hovy, 2011; Tuggener, 2014). Metrics that are being used widely are *MUC* (Vilain et al., 1995), B^3 (Bagga and Baldwin, 1998), *CEAF* (Luo, 2005), and *BLANC* (Recasens and Hovy, 2011). There are known flaws for each of these metrics. Besides, the agreement between all these metrics is relatively low (Holen, 2013), and it is not clear which metric is the most reliable. The CoNLL-2011/2012 shared tasks (Pradhan et al., 2011; Pradhan et al., 2012) ranked participating systems using an *average* of three metrics, i.e. *MUC*, B^3, and *CEAF*, following a proposal by (Denis and Baldridge, 2009a). Averaging three unreliable scores does not result in a reliable one. Besides, when an average score is used for comparisons, it is not possible to analyse recall and precision to determine which output is more precise and which one covers more coreference information. This is indeed a requirement for coreference resolvers to be used in end-tasks. Therefore, averaging individual metrics is nothing but a compromise.

As mentioned by Luo (2005), interpretability and discriminative power are two basic requirements for a reasonable evaluation metric. In regard to the *interpretability* requirement a high score should indicate that the vast majority of coreference relations and entities are detected correctly. Similarly, a system that resolves none of the coreference relations or entities should get a zero score.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 632–642,
Berlin, Germany, August 7-12, 2016.

	MUC			B^3			$CEAF_e$			BLANC		
	R	P	F_1	R	P	F_1	R	P	F_1	R	P	F_1
Base	69.31	76.23	72.60	55.83	66.07	60.52	54.88	**59.41**	57.05	57.46	65.77	61.31
More precise	69.31	**82.29**	**75.24**	53.94	**69.32**	60.67	50.92	55.85	53.27	53.60	66.45	59.25
Less precise[a]	69.31	69.46	69.38	60.53	63.98	62.21	64.82	53.06	**58.35**	68.68	67.02	67.68
Less precise[b]	69.31	74.70	71.90	**60.61**	69.14	**64.60**	**69.50**	47.61	56.51	**68.74**	**67.37**	**67.87**

Table 1: Counterintuitive values of B^3, CEAF and BLANC recall and precision.

An evaluation metric should also be *discriminative*. It should be able to discriminate between good and bad coreference decisions. In this paper, we report on a drawback for *B^3*, *CEAF*, and *BLANC* which violates the interpretability requirement. We also show that this flaw invalidates the recall/precision analysis of coreference outputs based on these three metrics. We then review the current evaluation metrics with their known flaws to explain why we cannot trust them and need a new reliable one. Finally, we propose *LEA*, a Link-based Entity Aware evaluation metric that is designed to overcome problems of the existing metrics. We have begun the process of integrating the *LEA* metric in the official CoNLL scorer[1] so as to continue the progress made in recent years to produce replicable evaluation metrics. In order to use the *LEA* metric, there is no additional requirement than that of the CoNLL scorer v8.01 [2].

2 The Mention Identification Effect

All the proposed evaluation metrics for coreference resolution use recall, precision and F_1 for reporting the performance of a coreference resolver. Recall is an indicator of the fraction of correct coreference information, i.e. coreference links or entities, that is resolved. Precision is an indicator of the fraction of resolved coreference information that is correct. F_1 is the weighted harmonic mean of recall and precision.

While we usually use F_1 for comparing coreference resolution systems, it is also important for the corresponding recall and precision values to be interpretable and discriminative. Coreference resolution is not an end-task itself but it is an important step toward text understanding. Depending on the task, recall or precision may be more important. For example, as Stuckhardt (2003) argues, a coreference resolver needs high precision to meet the specific requirements of text summarization and question answering.

In this section, we show that the recall and precision of the *B^3*, *CEAF* and *BLANC* metrics are neither interpretable nor reliable. We choose the output of the state-of-the-art coreference resolver of Wiseman et al. (2015) on the CoNLL 2012 English test set as the base output. The CoNLL 2012 English test set contains 222 documents (comprising 348 partially annotated sections). This test set contains 19,764 coreferring mentions that belong to 4,532 different entities.

In Table 1, *Base* represents the scores of (Wiseman et al., 2015) on the CoNLL 2012 test set. All reported scores in this paper are computed by the official CoNLL scorer v8.01 (Pradhan et al., 2014).

Assume $M_{k,r}$ is the set of mentions that exists in both key and response entities. Let $L_k(m)$ and $L_r(m)$ be the set of coreference links of mention m in the key and response entities, respectively. Mention m is an incorrectly resolved mention if $m \in M_{k,r}$ and $L_k(m) \cap L_r(m) = \emptyset$. Therefore, m is a coreferent mention that has at least one coreference link in the response entities. However, none of its detected coreference links in the response entities are correct.

By removing the incorrectly resolved mentions, the response entities will become more precise. The precision improves because the wrong links that are related to the incorrectly resolved mentions have been removed. Besides, the recall will not change because no correct coreference relations or entities have been added or removed.

We make the *Base* output more precise by removing all 1075 incorrectly resolved mentions from the response entities. The score for this more precise output is shown as *More precise* in Table 1. As can be seen, (1) recall changes for all the metrics except for *MUC*; (2) both *$CEAF_e$* recall and precision significantly decrease; and (3) *BLANC* recall notably decreases so that F_1 drops significantly in comparison to *Base*.

On the other hand, adding completely incorrect

[1] Currently available as branch `LEA-scorer` in `https://github.com/conll/reference-coreference-scorers`.

[2] *LEA* scores will be obtained by running the command `perl scorer.pl lea goldFile systemFile`.

entities to the response entities should not affect the recall and it should decrease the precision.

Assume $M_{d,k,\bar{r}}$ is the set of mentions of document d that exists in the key entities but is missing from the response entities. We can add completely incorrect entities to the *Base* output as follows: (1) By linking $m_1 \in M_{d,k,\bar{r}}$ to mention $m_2 \in M_{d,k,\bar{r}}$ that is non-coreferent with m_1. All the new wrong entities are of size two (*Less precise*^{a}). (2) By linking $m_1 \in M_{d,k,\bar{r}}$ to all mentions of $M_{d,k,\bar{r}}$ that are non-coreferent with m_1. In this case the new entities are larger but their number is smaller (*Less precise*^{b}). The number of new entities is 1350 and 283 for the first and second case, respectively. As can be seen from the results of Table 1, (1) recall changes for all metrics except for *MUC*; and (2) the B^3, *CEAF* and *BLANC* scores improve significantly over those of *Base* when the output is doubtlessly worse.

These experiments show that B^3, *CEAF* and *BLANC* are not reliable for recall-precision analysis. We refer to the problem that is causing these contradictory results as the **mention identification effect**.

3 Reasons for the Unreliable Results

In this section, we briefly give an overview of the common evaluation metrics for coreference resolution. We also discuss the shortcomings of each metric, including the **mention identification effect**, that may lead to counterintuitive and unreliable results. In all metrics, K is the key entity set and R is the response entity set.

3.1 MUC

MUC is the earliest systematic coreference evaluation metric and is introduced by Vilain et al. (1995). *MUC* is a link-based metric. It computes recall based on the minimum number of missing links in the response entities in comparison to the key entities. *MUC* recall is defined as:

$$\text{Recall} = \frac{\sum_{k_i \in K}(|k_i| - |p(k_i)|)}{\sum_{k_i \in K}(|k_i| - 1)}$$

where $p(k_i)$ is the set of partitions that is created by intersecting k_i with the corresponding response entities. *MUC* precision is computed by switching the role of the key and response entities.

It is not trivial to determine which evaluation metric discriminates coreference responses best. However, *MUC* is known to be the **least discriminative** coreference resolution metric (Bagga and Baldwin, 1998; Luo, 2005; Recasens and Hovy, 2011). The *MUC* evaluation is only based on the minimum number of missing/extra links in the response compared to the key entities. For instance, *MUC* does not differentiate whether an extra link merges two singletons or the two most prominent entities of the text. However, the latter error does more damage than the first one.

Another major problem with *MUC* is that it has an **incorrect preference in ranking** coreference outputs. *MUC* favors the outputs in which entities are over-merged (Luo, 2005). For instance, if we link all the key mentions of the CoNLL 2012 test set into a single response entity, the corresponding *MUC* scores, i.e. Recall=100, Precision=78.44 and F_1=87.91, will be all higher than those of the state-of-the-art system (*Base* in Table 1).

3.2 BCUBED

The B^3 score is introduced by Bagga and Baldwin (1998). B^3 is a mention-based metric, i.e., the overall recall/precision is computed based on the recall/precision of the individual mentions. For each mention m in the key entities, B^3 recall considers the fraction of the correct mentions that are included in the response entity of m. B^3 recall is computed as follows:

$$\text{Recall} = \frac{\sum_{k_i \in K} \sum_{r_j \in R} \frac{|k_i \cap r_j|^2}{|k_i|}}{\sum_{k_i \in K} |k_i|}$$

Similar to MUC, B^3 precision is computed by switching the role of the key and response entities.

The **mention identification effect** arises in B^3, because B^3 uses mentions instead of coreference relations to evaluate the response entities. Therefore, if a mention exists in a response entity, it is considered as a resolved mention regardless of whether it has a correct coreference relation in the response entity.

Luo (2005) argues that B^3 leads to **counter-intuitive results for boundary cases**: (1) consider a system that makes no decision and leaves every key mention as a singleton. B^3 precision for this system is 100%. However, not all of the recognized system entities (i.e. singletons), or the detected coreference relations (i.e. every mention only coreferent with itself) are correct; (2) consider a system that merges all key mentions into a single entity. B^3 recall for this system is 100%.

Luo (2005) interprets this recall as counterintuitive because the key entities have not been found in the response. The intuitiveness or counterintuitiveness of this recall value depends on the evaluator's point of view. From one point of view, all of the key mentions, that are supposed to be in the same entity, are indeed in the same entity.

Finally, as discussed by Luo and Pradhan (2016), B^3 cannot properly handle **repeated mentions in the response entities**. If a gold mention is repeated in several response entities, B^3 receives credit for all the repetitions. The repeated response mentions issue is not an imaginary problem (Luo and Pradhan, 2016). It can happen if system mentions are read from a parse tree where an NP node has a single child, a pronoun, and where both the nodes are considered as candidate mentions.

3.3 CEAF

The *CEAF* metric is introduced by Luo (2005). *CEAF*'s main assumption is that each key entity should only be mapped to one reference entity, and vice versa. *CEAF* uses a similarity measure (ϕ) to evaluate the similarity of two entities. It uses the Kuhn-Munkres algorithm to find the best one-to-one mapping of the key to the response entities (g^*) using the given similarity measure. Assuming K^* is the set of key entities that is included in the optimal mapping, recall is computed as:

$$\text{Recall} = \frac{\sum_{k_i \in K^*} \phi(k_i, g^*(k_i))}{\sum_{k_i \in K} \phi(k_i, k_i)} \qquad (1)$$

For computing *CEAF* precision, the denominator of Equation 1 is changed to $\sum_{R_i \in R} \phi(r_i, r_i)$. Based on ϕ, there are two variants of CEAF: (1) mention-based *CEAF* (*CEAF*$_m$), which computes the similarity as the number of common mentions between two entities, i.e. $\phi(k_i, r_j) = |k_i \cap r_j|$; and (2) entity-based *CEAF* (*CEAF*$_e$), in which $\phi(k_i, r_j) = \frac{2 \times |k_i \cap r_j|}{|k_i| + |r_j|}$. The denominator of Equation 1 for *CEAF*$_e$ is the number of key entities.

Similar to B^3, the **mention identification effect** of *CEAF* is caused by both similarity measures of *CEAF* using the number of common mentions between two entities, i.e. $|k_i \cap r_j|$. In this way, even if the two mapped entities (k_i and r_j) have only one mention in common, *CEAF*$_m$ rewards recall and precision by $\frac{1}{\sum_{k_i} |k_i|}$ and $\frac{1}{\sum_{r_j} |r_j|}$, respectively. *CEAF*$_e$ rewards recall and precision by $\frac{2}{(|k_i| + |r_j|) \times |K|}$ and $\frac{2}{(|k_i| + |r_j|) \times |R|}$, respectively. If instead of the number of common mentions, we would use the number of common coreference links between two entities in both *CEAF*$_m$ and *CEAF*$_e$ similarity measures, this problem would be solved. However, even if we handle the mention identification effect by using coreference relations rather than mentions in the similarity measures, *CEAF* may still result in counterintuitive results. As mentioned by Denis and Baldridge (2009b), *CEAF* **ignores all correct decisions of unaligned response entities** that may lead to unreliable results. In order to illustrate this, we use a sample text from the CoNLL 2012 development set as an example (Figure 1). Gold mentions are enclosed in square brackets. Mentions with the same text are marked with different indices. The indices in parentheses denote to which key entity the mentions belong Consider cr_1 and cr_2 in Table 2, which are different responses for entity (1) of Figure 1. cr_1 resolves many coreference relations of entity (1). However, it misses that *they*$_1$ could refer to an entity which is already referred to by 'it'. Therefore cr_1 produces two entities instead of one because of this missing relation. On the other hand, cr_2 only recognizes half of the correct coreference relations of entity (1).

As can be seen from Table 3, CEAF prefers cr_2 over cr_1 even though cr_1 makes more correct decisions. CEAF only selects one of the output entities of cr_1 for giving credit to the correct decisions.

[The American administration]$_{(1)}$ committed a fatal mistake when [it$_1$]$_{(1)}$ [executed]$_{(2)}$ [this man]$_{(3)}$, in a way for which [it$_2$]$_{(1)}$ will pay a hefty price in the near future. [[His$_1$]$_{(3)}$ survival]$_{(4)}$ would have benefited [it$_3$]$_{(1)}$ much more than [[his$_2$]$_{(3)}$ execution]$_{(2)}$ if [they$_1$]$_{(1)}$ understood politics as [they$_2$]$_{(1)}$ should, because [[his$_3$]$_{(3)}$ survival]$_{(4)}$ could have been a card to threaten [the sectarians]$_{(5)}$ and keep [them$_1$]$_{(5)}$ as servants to [them$_1$]$_{(1)}$ and [their]$_{(1)}$ schemes.

Figure 1: Sample text from CoNLL 2012.

	Response entities
cr_1	r_1={the American administration, it$_1$, it$_2$, it$_3$} , r_2={they$_1$, they$_2$, them, their}
cr_2	r_1={the American administration, it$_1$, it$_2$, it$_3$}

Table 2: Different system outputs for Figure 1.

	MUC	B^3	CEAF$_m$	CEAF$_e$	BLANC
cr_1	92.30	66.66	50.00	44.44	60.00
cr_2	60.00	40.00	66.66	66.66	32.29

Table 3: F_1 scores for Table 2's response entities.

The other response entity is only used for penalizing the precision of cr_1. This counterintuitive result is only because of the stringent constraint of CEAF that the mapping of key to response entities should be one-to-one.

Another problem with *CEAF*$_e$, mentioned by Stoyanov et al. (2009), is that it **weights entities equally regardless of their sizes**. The system that does not detect entity (1), the most prominent entity of Figure 1, gets the same score as that of a system which does not detect entity (4) of size 2.

3.4 BLANC

BLANC (Recasens and Hovy, 2011; Luo et al., 2014) is a link-based metric that adapts the Rand index (Rand, 1971) to coreference resolution evaluation. Let C_k and C_r be the sets of coreference links in the key and response entities, respectively. Assume N_k and N_r are the sets of non-coreference links in the key and response entities, respectively. Recall and precision of coreference links are computed as:

$$R_c = \frac{|C_k \cap C_r|}{|C_k|}, \quad P_c = \frac{|C_k \cap C_r|}{|C_r|}$$

Recall and precision of non-coreference links are computed as:

$$R_n = \frac{|N_k \cap N_r|}{|N_k|}, \quad P_n = \frac{|N_k \cap N_r|}{|N_r|}$$

BLANC recall and precision are computed by averaging the recall and precision of coreference and non-coreference links, e.g. Recall$= \frac{R_c+R_n}{2}$.

The *BLANC* measure is the newest but the least popular metric for evaluating coreference resolvers. Because of considering non-coreferent relations, the **mention identification effect** affects *BLANC* most strongly. When the number of gold mentions that exist in the response entities is larger, the number of detected non-coreference links will also get larger. Therefore, it results in higher values for *BLANC* recall and precision ignoring whether those gold mentions are resolved.

4 LEA

In this section, we present our new evaluation metric, namely the Link-Based Entity-Aware metric (*LEA*). *LEA* is designed to overcome the shortcomings of the current evaluation metrics.

For each entity, *LEA* considers how important the entity is and how well it is resolved. Therefore, *LEA* evaluates a set of entities as follows:

$$\frac{\sum_{e_i \in E}(importance(e_i) \times resolution\text{-}score(e_i))}{\sum_{e_k \in E} importance(e_k)}$$

We consider the size of an entity as a measure of importance, i.e. $importance(e) = |e|$. Therefore, the more prominent entities of the text get higher importance values. However, according to the end-task or domain used, one can choose other importance measures based on factors besides e_i's size, e.g. e_i's entity type or e_i's mention types. For example, as suggested by Holen (2013), each mention carries different information values, and considering this information could benefit the quantitative evaluation of coreference resolution. The $importance$ measure of *LEA* is the appropriate place to incorporate this kind of information.

Entity e with n mentions has $link(e) = n \times (n-1)/2$ unique coreference links. The resolution score of key entity k_i is computed as the fraction of correctly resolved coreference links of k_i:

$$resolution\text{-}score(k_i) = \sum_{r_j \in R} \frac{link(k_i \cap r_j)}{link(k_i)}$$

For each k_i, *LEA* checks all the response entities to see whether they are partial matches for k_i. r_j is a partial match for k_i, if it contains at least one of the coreference links of k_i. Thus, if a response entity only contains one mention of k_i, it is not a partial mapping of k_i.

Having the definitions of $importance$ and $resolution\text{-}score$, *LEA* recall is computed as:

$$\text{Recall} = \frac{\sum_{k_i \in K}(|k_i| \times \sum_{r_j \in R} \frac{link(k_i \cap r_j)}{link(k_i)})}{\sum_{k_z \in K} |k_z|}$$

LEA precision is computed by switching the role of the key and response entities:

$$\text{Precision} = \frac{\sum_{r_i \in R}(|r_i| \times \sum_{k_j \in K} \frac{link(r_i \cap k_j)}{link(r_i)})}{\sum_{r_z \in R} |r_z|}$$

LEA handles singletons by self-links. A self-link is a link connecting a mention to itself. Self-links indicate that a mention is only coreferent with itself and not with other mentions. By considering self-links, the number of links in a singleton is one. If entity k_i is a singleton, $link(k_i \cap r_j)$ is one only if r_j is a singleton and contains the same mention as k_i.

In summary, *LEA* is a link-based metric with the following properties:

– *LEA* takes into account all coreference links instead of only extra/missing links. Therefore, it has more discriminative power than MUC.
– *LEA* evaluates resolved coreference relations instead of resolved mentions. *LEA* also does not rely on non-coreferent links in order to detect entity structures or singletons. Therefore, the **mention identification effect** does not apply to *LEA* recall and precision. As a result, one can trust *LEA* recall or precision.
– *LEA* allows one-to-many mappings of entities. Unlike *CEAF*, all correct coreference relations are rewarded by *LEA*. More splits (or similarly merges) in entity k_i result in a smaller $\sum_{r_j \in R} link(k_i \cap r_j)$. Therefore, splitting (merging) of an entity in several entities will be penalized implicitly in $resolution\text{-}score$.
– *LEA* takes the importance of missing/extra entities into account. Therefore, unlike $CEAF_e$, it differentiates between the outputs missing the most prominent and the smallest entities.
– *LEA* considers resolved coreference relations instead of resolved mentions. Therefore, the existence of repeated mentions in different response entities is not troublesome for *LEA*.

5 An Illustrative Example

In this section, we use the example from Pradhan et al. (2014) to show the process of computing the LEA scores. In this example, $K = \{k_1 = \{a, b, c\}, k_2 = \{d, e, f, g\}\}$ is the set of key entities and $R = \{r_1 = \{a, b\}, r_2 = \{c, d\}, r_3 = \{f, g, h, i\}\}$ is the set of response entities.

Here we assume that $importance$ corresponds to entity size. Hence, $importance(k_1) = 3$ and $importance(k_2) = 4$. The sets of coreference links in k_1 and k_2 are $\{ab, ac, bc\}$ and $\{de, df, dg, ef, eg, fg\}$, respectively. Therefore, $link(k_1) = 3$ and $link(k_2) = 6$. ab is the only common link between k_1 and r_1. There are no common links between k_1 and the two other response entities. Similarly, k_2 has one common link with r_3 and it has no common links with r_1 or r_2. Therefore, $resolution\text{-}score(k_1) = \frac{1+0+0}{3}$ and $resolution\text{-}score(k_2) = \frac{0+0+1}{6}$. As a result *LEA* recall is computed as:

$$\frac{\sum importance(k_i) \times resolution\text{-}score(k_i)}{\sum importance(k_j)} = \frac{3 \times \frac{1}{3} + 4 \times \frac{1}{6}}{3+4} \approx 0.24$$

By changing the roles of key and response entities, *LEA* precision is computed as:

$$\frac{2 \times \frac{1+0}{1} + 2 \times \frac{0+0}{1} + 4 \times \frac{0+1}{6}}{2+2+4} \approx 0.33$$

6 Evaluation on Real Data

Table 4 shows the scores of the state-of-the-art coreference resolvers developed by Wiseman et al. (2015), Martschat and Strube (2015), and Peng et al. (2015). Clark and Manning (2015)'s resolver is also among the state-of-the-art systems but we did not have access to their output. Considering the average score of *MUC*, B^3, and $CEAF_e$, *Martschat*, and *Peng* perform equally. However, according to *LEA*, *Martschat* performs significantly better based on an approximate randomization test (Noreen, 1989). $CEAF_e$ also agrees with *LEA* for this ranking. However, $CEAF_e$ recall and precision are similar for *Peng* while based on *LEA*, *Peng*'s precision is marginally better than recall.

In addition to the state-of-the-art systems, we report the scores of boundary cases in the CoNLL 2012 test set in Table 4: (1) *sys-sing*: all system mentions as singletons; and (2) *sys-1ent*: all system mentions in a single entity.

Table 5 presents the evaluations of the participating systems in the CoNLL 2012 shared task (closed task with predicted mentions). The rankings are specified in parentheses. For the *LEA* rankings we also perform a significance test. The systems without significant differences have the same ranking. The main difference between the rankings of *avg.* and *LEA* is the rank of *xu*. Based on *LEA*, *xu* is significantly better than *chen* and *chunyuang*, while *avg.* ranks these two above *xu*. The recall values of *chen* and *chunyuang* for mention identification are 75.08 and 75.23, which are higher than those of the best performing systems, i.e 72.75 for *fernandes*, and 74.23 for *martschat*. *chen* and *chunyuang* include 1850 and 1735 gold mentions in their outputs that have not a single correct coreference link. On the other hand, the number of these gold mentions in *xu* is 757. Therefore, these different rankings could be a direct result of the **mention identification effect**.

Overall, using one reliable metric instead of an average score benefits us in two additional ways: (1) we can perform a significance test to check whether there is a meaningful difference, and (2) the recall and precision values are meaningful.

	MUC			B^3			$CEAF_e$			CoNLL	LEA		
	R	P	F_1	R	P	F_1	R	P	F_1	Avg. F_1	R	P	F_1
Wiseman	69.31	76.23	72.60	55.83	66.07	60.52	54.88	59.41	57.05	63.39	51.78	62.12	56.48
Martschat	68.55	77.22	72.63	54.64	66.78	60.11	52.85	60.30	56.33	63.02	50.64	62.87	56.10
Peng	69.54	75.80	72.53	56.91	65.40	60.86	55.49	55.98	55.73	63.04	51.91	58.97	55.21
sys-sing	0.00	0.00	0.00	19.72	39.05	26.20	50.32	4.99	9.08	11.76	0.00	0.00	0.00
sys-1ent	88.01	29.58	44.28	84.87	2.53	4.91	1.50	19.63	2.80	17.33	82.31	2.27	4.43

Table 4: Results on the CoNLL 2012 test set.

	MUC	B^3	$CEAF_m$	$CEAF_e$	BLANC	CoNLL avg.	LEA
fernandes	70.51 (1)	57.58 (1)	61.42	53.86 (1)	58.75	60.65 (1)	53.28 (1)
martschat	66.97 (3)	54.62 (2)	58.77	51.46 (2)	55.04	57.68 (2)	49.99 (2)
bjorkelund	67.58 (2)	54.47 (3)	58.19	50.21(3)	55.42	57.42 (3)	49.98 (2)
chang	66.38 (4)	52.99 (4)	57.10	48.94 (4)	53.86	56.10 (4)	48.50 (4)
chen	63.71 (7)	51.76 (5)	55.77	48.10 (5)	52.87	54.52 (5)	46.24 (6)
chunyuang	63.82 (6)	51.21 (6)	55.10	47.58 (6)	52.65	54.20 (6)	45.84 (6)
shou	62.91 (8)	49.44 (9)	53.16	46.66 (7)	50.44	53.00 (7)	43.97 (8)
yuan	62.55 (9)	50.11 (8)	54.53	45.99 (8)	52.10	52.88 (8)	44.76 (8)
xu	66.18 (5)	50.30 (7)	51.31	41.25 (11)	46.47	52.58 (9)	46.83 (5)
uryupina	60.89 (10)	46.24 (10)	49.31	42.93 (9)	46.04	50.02 (10)	41.15 (10)
songyang	59.83 (12)	45.90 (11)	49.58	42.36 (10)	45.10	49.36 (11)	41.25 (10)
zhekova	53.52 (13)	35.66 (13)	39.66	32.16 (12)	34.80	40.45 (12)	29.98 (12)
xinxin	48.27 (14)	35.73 (12)	37.99	31.90 (13)	36.54	38.63 (13)	29.22 (12)
li	50.84 (11)	32.29 (14)	36.28	25.21 (14)	31.85	36.11 (14)	27.32 (14)

Table 5: The results of the CoNLL 2012 shared task.

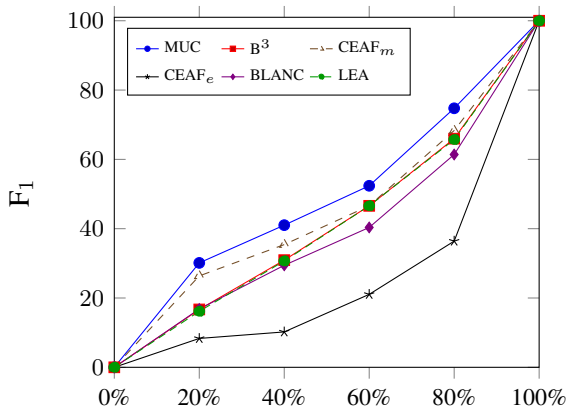


Figure 2: Resolved coreference links ratio without incorrect links.

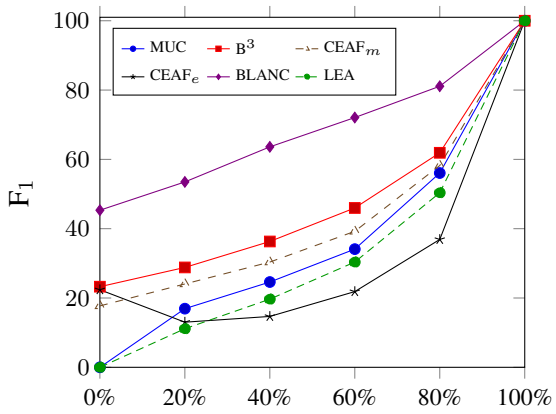


Figure 3: Resolved coreference links ratio in the presence of incorrect links.

7 Analysis

In this section we analyze the behavior of the evaluation metrics based on various coreference resolution errors. The set of key entities in all experiments contains: one entity of size 20, two entities of size 10, three entities of size 5, one entity of size 4, and ten entities of size 2.

7.1 Correct Links

We analyze different metrics based on the ratio of correctly resolved coreference links: (1) without wrong coreference links (Figure 2), and (2) with wrong coreference links (Figure 3). In the experiments of Figure 2, only mentions that are correctly resolved exist in the response. In Figure 3, apart from the mentions that are resolved correctly, other mentions are linked to at least one non-coreferent mention. Therefore, mention detection F_1 is always 100%.

The following observations can be drawn from these experiments: (1) *MUC* and *LEA* are the only measures which give a zero score to the response that contains no correct coreference relations; (2) in our experiments, $CEAF_e$ shows an unreasonable drop when the correct link ratio changes from 0% to 20%; and (3), in Figure 2, the *BLANC*

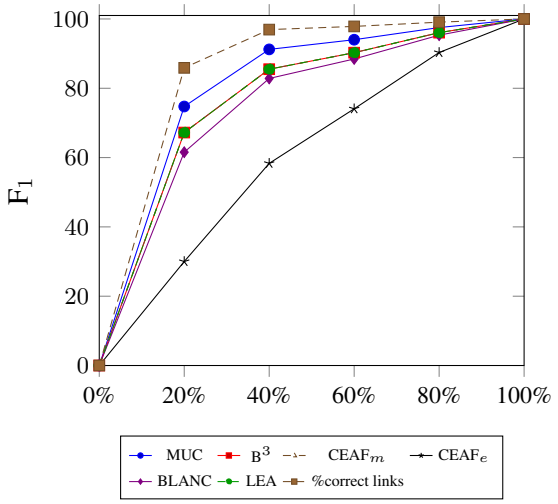


Figure 4: Resolving entities in decreasing order. F_1 of B^3, CEAF, and LEA are the same.

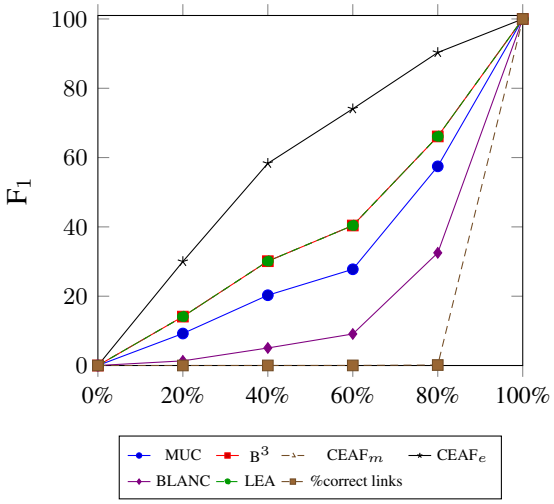


Figure 5: Resolving entities in increasing order. F_1 of B^3, CEAF, and LEA are the same.

F_1 values are less than or equal to those of B^3 and *LEA*. However, in Figure 3 that contains both coreferent and non-coreferent links, *BLANC* F_1 is at least 20% higher than that of other metrics.

7.2 Correct Entities

Apart from the correctly resolved links, a coreference metric should also take into account the resolved entities. In this section, we analyze the coreference resolution metrics based on the number and the size of the correctly resolved entities. In these experiments, each entity is either resolved completely, or all of its mentions are absent from the response. In Figure 4, the key entities are added to the response in decreasing order of their size. Figure 5 shows the experiments in which the entities are resolved in increasing order. The ratio of the correctly resolved coreference links is shown in both figures.

We can observe the following points from Figure 4 and Figure 5: (1) $CEAF_e$ results in the same F_1 values regardless of the size of entities that are resolved or are missing; (2) B^3, $CEAF_m$ and *LEA* result in the same F_1 values; and (3) *BLANC* is very sensitive to the total number of links.

7.3 Splitting/Merging Entities

The effect of splitting a single entity into two or more entities is studied in Figure 6. The overall effect of merging entities would be similar to that of splitting if the roles of the key and response entities change. In each experiment, only one key entity is split in a way that no singletons are created. For example, 18-2 in the horizontal axis indicates

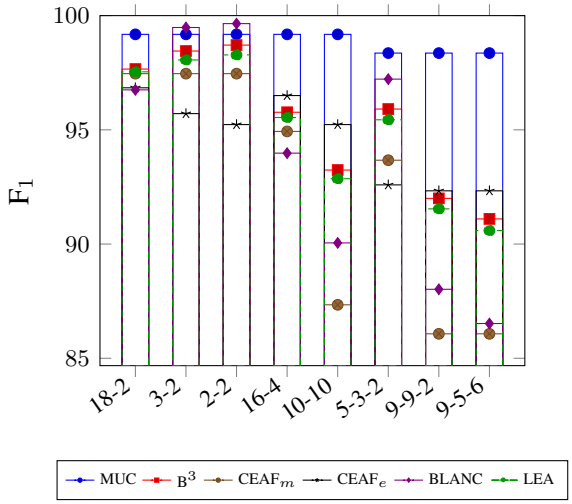


Figure 6: Effect of splitting entities.

that an entity of size 20 is split into two entities of size 18 and 2.

The following observations can be drawn from Figure 6: (1) *MUC* only recognizes the number of splits regardless of the size of entities; (2) $CEAF_e$ does not differentiate 2-2 from 10-10, and 9-9-2 from 9-5-6; and (3) the highest disagreement is for ranking different numbers of splits in entities with different sizes, i.e., B^3: 18-2>5-3-2>16-4, *BLANC*: 5-3-2>18-2>16-4, *CEAF*: 18-2>16-4>5-3-2, and *LEA*: 18-2>16-4>5-3-2. These are the cases that are even for humans hard to rank.

7.4 Extra/Missing Mentions

Figure 7 shows the effect of extra mentions, i.e. mentions that are not included in any key entity. If we change the roles of the key and response enti-

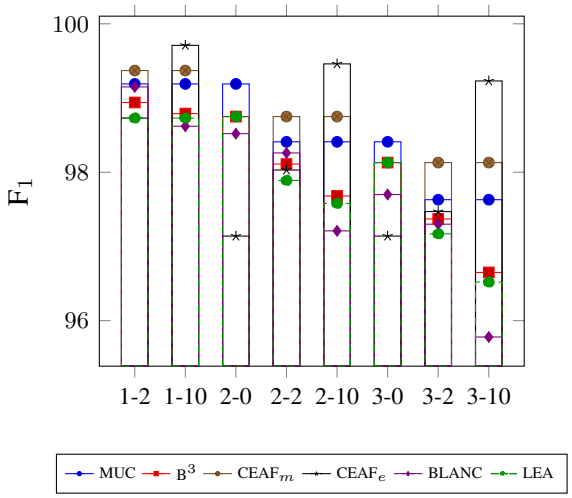


Figure 7: Effect of extra mentions.

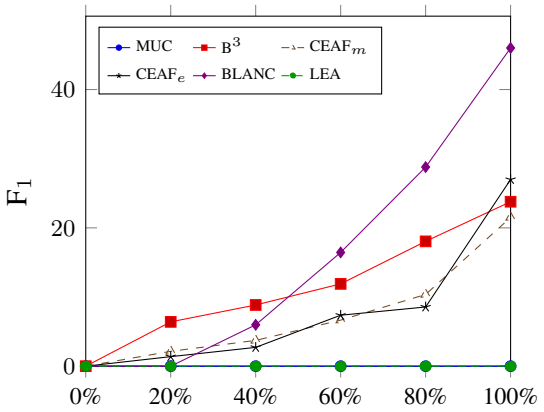


Figure 8: Effect of mention identification.

ties, the overall effect of missing mentions would be similar. In the horizontal axis, the first number shows the number of extra mentions. The second number shows the size of the entity to which extra mentions are added. A zero entity size indicates the extra mentions are linked together.

The following points are worth noting from the results of Figure 7: (1) *MUC* and $CEAF_m$ are the least discriminative metrics when the system output includes extra mentions; (2) except for $CEAF_e$, other metrics rank 3-10 as the worst output;(3) $CEAF_e$ recognizes both 2-0 and 3-0 as the worst outputs. However, in these outputs the extra mentions are linked together and therefore no incorrect information is added to the correctly resolved entities; and (4) *LEA* is the only metric that recognizes error 2-0 is less harmful than 1-2 or 1-10. However, *LEA* does not discriminate the different outputs in which only one extra mention is added to an entity. If k extra mentions are added to an entity of size n, the corresponding resolution error multiplied by the importance of the response entity is $(n+k) \times (1 - \frac{n \times (n-1)}{(n+k) \times (n+k-1)})$. If $k = 1$, this equation is 2 regardless of n's value.

7.5 Mention Identification

The **mention identification effect** is shown in Figure 8. In all experiments, the number of correct coreference links is *zero*. The horizontal axis shows the mention identification accuracy in the system output. The F_1 of B^3, *CEAF* and *BLANC* in these experiments clearly contrast the interpretability requirement. A coreference resolver with a non-zero score should have resolved some of the coreference relations.

8 Conclusions

Current coreference resolution evaluation metrics have flaws which make them unreliable for comparing coreference resolvers. There is also a low agreement between the rankings of different metrics. The current solution is to use an average value of different metrics for comparisons. Averaging unreliable scores does not result in a reliable one. Indeed, recall and precision comparisons of coreference resolvers are not possible based on an average score. We first report the **mention identification effect** on B^3, *CEAF* and *BLANC* which causes these metrics to report misleading values. The only metric that is resistant to the mention identification effect is the least discriminative one, i.e. *MUC*. We introduce *LEA*, the Link-based Entity-Aware metric, as a new evaluation metric for coreference resolution. *LEA* is a simple intuitive metric that overcomes the drawbacks of the current metrics. It can be easily adapted for entity evaluation in different domains or applications in which entities with various attributes are of different importance.

Acknowledgments

The authors would like to thank Sameer Pradhan, Mark-Christoph Müller, Mohsen Mesgar and Sebastian Martschat for their helpful comments. We would also like to thank Sam Wiseman and Haoruo Peng for providing us with their coreference system outputs. This work has been funded by the Klaus Tschira Foundation, Heidelberg, Germany. The first author has been supported by a Heidelberg Institute for Theoretical Studies PhD. scholarship.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *Proceedings of the 1st International Conference on Language Resources and Evaluation,* Granada, Spain, 28–30 May 1998, pages 563–566.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1405–1415.

Pascal Denis and Jason Baldridge. 2009a. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, (42):87–96.

Pascal Denis and Jason Baldridge. 2009b. Global joint models for coreference resolution and named entity classification. *Procesamiento del Lenguaje Natural*, 42:87–96, March.

Gordana Ilic Holen. 2013. Critical reflections on evaluation practices in coreference resolution. In *Proceedings of the 2013 NAACL HLT Student Research Workshop,* Atlanta, Georgia, 9-14 June 2013, pages 1–7.

Xiaoqiang Luo and Sameer Pradhan. 2016. Evaluation metrics. In M. Poesio, R. Stuckardt, and Y. Versley, editors, *Anaphora Resolution: Algorithms, Resources, and Applications*. Springer. To appear.

Xiaoqiang Luo, Sameer Pradhan, Marta Recasens, and Eduard Hovy. 2014. An extension of BLANC to system mentions. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 24–29, Baltimore, Maryland, June. Association for Computational Linguistics.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Proceedings of the Human Language Technology Conference and the 2005 Conference on Empirical Methods in Natural Language Processing,* Vancouver, B.C., Canada, 6–8 October 2005, pages 25–32.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics*, 3:405–418.

Eric W. Noreen. 1989. *Computer Intensive Methods for Hypothesis Testing: An Introduction*. Wiley, New York, N.Y.

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015. A joint framework for coreference resolution and mention head detection. In *Proceedings of the 19th Conference on Computational Natural Language Learning,* Beijing, China, 30–31 July 2015, pages 12–21.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. CoNLL-2011 Shared Task: Modeling unrestricted coreference in OntoNotes. In *Proceedings of the Shared Task of the 15th Conference on Computational Natural Language Learning,* Portland, Oreg., 23–24 June 2011, pages 1–27.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 Shared Task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Shared Task of the 16th Conference on Computational Natural Language Learning,* Jeju Island, Korea, 12–14 July 2012, pages 1–40.

Sameer Pradhan, Xiaoqiang Luo, Marta Recasens, Eduard Hovy, Vincent Ng, and Michael Strube. 2014. Scoring coreference partitions of predicted mentions: A reference implementation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers),* Baltimore, Md., 22–27 June 2014, pages 30–35.

William R. Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850.

Marta Recasens and Eduard Hovy. 2011. BLANC: Implementing the Rand index for coreference evaluation. *Natural Language Engineering*, 17(4):485–510.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing,* Singapore, 2–7 August 2009, pages 656–664.

Roland Stuckhardt. 2003. Coreference-based summarization and question answering: A case for high precision anaphor resolution. In *Proceedings of the 2003 International Symposium on Reference Resolution and Its Applications to Question Answering and Summarization,* Venice, Italy, 23–24 June 2003, pages 33–42.

Don Tuggener. 2014. Coreference resolution evaluation for higher level applications. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, Volume 2: Short Papers,* Gothenburg, Sweden, 26–30 April 2014, pages 231–235.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th Message Understanding Conference (MUC-6)*, pages 45–52, San Mateo, Cal. Morgan Kaufmann.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* Beijing, China, 26–31 July 2015, pages 1416–1426.

Improving Coreference Resolution by Learning Entity-Level Distributed Representations

Kevin Clark
Computer Science Department
Stanford University
kevclark@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
manning@cs.stanford.edu

Abstract

A long-standing challenge in coreference resolution has been the incorporation of entity-level information – features defined over clusters of mentions instead of mention pairs. We present a neural network based coreference system that produces high-dimensional vector representations for pairs of coreference clusters. Using these representations, our system learns when combining clusters is desirable. We train the system with a learning-to-search algorithm that teaches it which local decisions (cluster merges) will lead to a high-scoring final coreference partition. The system substantially outperforms the current state-of-the-art on the English and Chinese portions of the CoNLL 2012 Shared Task dataset despite using few hand-engineered features.

1 Introduction

Coreference resolution, the task of identifying which mentions in a text refer to the same real-world entity, is fundamentally a clustering problem. However, many recent state-of-the-art coreference systems operate solely by linking pairs of mentions together (Durrett and Klein, 2013; Martschat and Strube, 2015; Wiseman et al., 2015).

An alternative approach is to use agglomerative clustering, treating each mention as a singleton cluster at the outset and then repeatedly merging clusters of mentions deemed to be referring to the same entity. Such systems can take advantage of entity-level information, i.e., features between clusters of mentions instead of between just two mentions. As an example for why this is useful, it is clear that the clusters {*Bill Clinton*} and {*Clinton*, *she*} are not referring to the same entity, but it is ambiguous whether the pair of mentions *Bill Clinton* and *Clinton* are coreferent.

Previous work has incorporated entity-level information through features that capture hard constraints like having gender or number agreement between clusters (Raghunathan et al., 2010; Durrett et al., 2013). In this work, we instead train a deep neural network to build distributed representations of pairs of coreference clusters. This captures entity-level information with a large number of learned, continuous features instead of a small number of hand-crafted categorical ones.

Using the cluster-pair representations, our network learns when combining two coreference clusters is desirable. At test time it builds up coreference clusters incrementally, starting with each mention in its own cluster and then merging a pair of clusters each step. It makes these decisions with a novel easy-first cluster-ranking procedure that combines the strengths of cluster-ranking (Rahman and Ng, 2011) and easy-first (Stoyanov and Eisner, 2012) coreference algorithms.

Training incremental coreference systems is challenging because the coreference decisions facing a model depend on previous decisions it has already made. We address this by using a learning-to-search algorithm inspired by SEARN (Daumé III et al., 2009) to train our neural network. This approach allows the model to learn which action (a cluster merge) available from the current state (a partially completed coreference clustering) will eventually lead to a high-scoring coreference partition.

Our system uses little manual feature engineering, which means it is easily extended to multiple languages. We evaluate our system on the English and Chinese portions of the CoNLL 2012 Shared Task dataset. The cluster-ranking model significantly outperforms a mention-ranking model that

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 643–653,
Berlin, Germany, August 7-12, 2016.

does not use entity-level information. We also show that using an easy-first strategy improves the performance of the cluster-ranking model. Our final system achieves CoNLL F_1 scores of 65.29 for English and 63.66 for Chinese, substantially outperforming other state-of-the-art systems.[1]

2 System Architecture

Our cluster-ranking model is a single neural network that learns which coreference cluster merges are desirable. However, it is helpful to think of the network as being composed of distinct subnetworks. The *mention-pair encoder* produces distributed representations for pairs of mentions by passing relevant features through a feedforward neural network. The *cluster-pair encoder* produces distributed representations for pairs of clusters by applying a pooling operation over the representations of relevant mention pairs, i.e., pairs where one mention is in each cluster. The *cluster-ranking model* then scores pairs of clusters by passing their representations through a single neural network layer.

We also train a *mention-ranking model* that scores pairs of mentions by passing their representations through a single neural network layer. Its parameters are used to initialize the cluster-ranking model, and the scores it produces are used to prune which candidate cluster merges the cluster-ranking model considers, allowing the cluster-ranking model to run much faster. The system architecture is summarized in Figure 1.

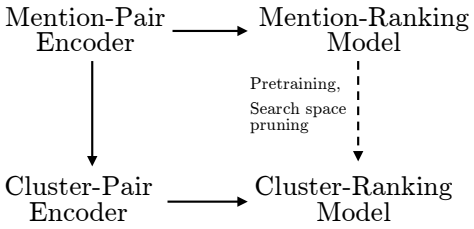


Figure 1: System architecture. Solid arrows indicate one neural network is used as a component of the other; the dashed arrow indicates other dependencies.

3 Building Representations

In this section, we describe the neural networks producing distributed representations of pairs of mentions and pairs of coreference clusters. We assume that a set of mentions has already been extracted from each document using a method such as the one in Raghunathan et al. (2010).

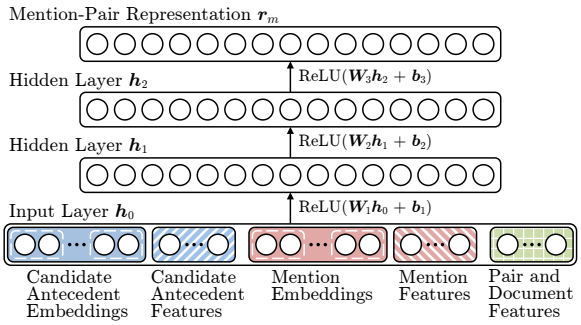


Figure 2: Mention-pair encoder.

3.1 Mention-Pair Encoder

Given a mention m and candidate antecedent a, the mention-pair encoder produces a distributed representation of the pair $\boldsymbol{r}_m(a, m) \in \mathbb{R}^d$ with a feedforward neural network, which is shown in Figure 2. The candidate antecedent may be any mention that occurs before m in the document or NA, indicating that m has no antecedent. We also experimented with models based on Long Short-Term Memory recurrent neural networks (Hochreiter and Schmidhuber, 1997), but found these to perform slightly worse when used in an end-to-end coreference system due to heavy overfitting to the training data.

Input Layer. For each mention, the model extracts various words and groups of words that are fed into the neural network. Each word is represented by a vector $\boldsymbol{w}_i \in \mathbb{R}^{d_w}$. Each group of words is represented by the average of the vectors of each word in the group. For each mention and pair of mentions, a small number of binary features and distance features are also extracted. Distances and mention lengths are binned into one of the buckets $[0, 1, 2, 3, 4, 5\text{-}7, 8\text{-}15, 16\text{-}31, 32\text{-}63, 64+]$ and then encoded in a one-hot vector in addition to being included as continuous features. The full set of features is as follows:

Embedding Features: Word embeddings of the head word, dependency parent, first word, last word, two preceding words, and two following words of the mention. Averaged word embeddings of the five preceding words, five following

[1] Code and trained models are available at `https://github.com/clarkkev/deep-coref`.

words, all words in the mention, all words in the mention's sentence, and all words in the mention's document.

Additional Mention Features: The type of the mention (pronoun, nominal, proper, or list), the mention's position (index of the mention divided by the number of mentions in the document), whether the mentions is contained in another mention, and the length of the mention in words.

Document Genre: The genre of the mention's document (broadcast news, newswire, web data, etc.).

Distance Features: The distance between the mentions in sentences, the distance between the mentions in intervening mentions, and whether the mentions overlap.

Speaker Features: Whether the mentions have the same speaker and whether one mention is the other mention's speaker as determined by string matching rules from Raghunathan et al. (2010).

String Matching Features: Head match, exact string match, and partial string match.

The vectors for all of these features are concatenated to produce an I-dimensional vector $\boldsymbol{h}_0$, the input to the neural network. If $a = \text{NA}$, the features defined over mention pairs are not included. For this case, we train a separate network with an identical architecture to the pair network except for the input layer to produce anaphoricity scores.

Our set of hand-engineered features is much smaller than the dozens of complex features typically used in coreference systems. However, we found these features were crucial for getting good model performance. See Section 6.1 for a feature ablation study.

Hidden Layers. The input gets passed through three hidden layers of rectified linear (ReLU) units (Nair and Hinton, 2010). Each unit in a hidden layer is fully connected to the previous layer:

$$\boldsymbol{h}_i(a, m) = \max(0, \boldsymbol{W}_i \boldsymbol{h}_{i-1}(a, m) + \boldsymbol{b}_i)$$

where $\boldsymbol{W}_1$ is a $M_1 \times I$ weight matrix, $\boldsymbol{W}_2$ is a $M_2 \times M_1$ matrix, and $\boldsymbol{W}_3$ is a $d \times M_2$ matrix.

The output of the last hidden layer is the vector representation for the mention pair: $\boldsymbol{r}_m(a, m) = \boldsymbol{h}_3(a, m)$.

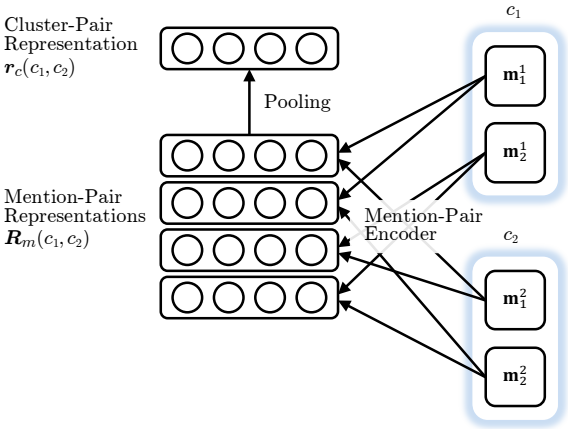


Figure 3: Cluster-pair encoder.

3.2 Cluster-Pair Encoder

Given two clusters of mentions $c_i = \{m_1^i, m_2^i, ..., m_{|c_i|}^i\}$ and $c_j = \{m_1^j, m_2^j, ..., m_{|c_j|}^j\}$, the cluster-pair encoder produces a distributed representation $\boldsymbol{r}_c(c_i, c_j) \in \mathbb{R}^{2d}$. The architecture of the encoder is summarized in Figure 3.

The cluster-pair encoder first combines the information contained in the matrix of mention-pair representations $\boldsymbol{R}_m(c_i, c_j) = [\boldsymbol{r}_m(m_1^i, m_1^j), \boldsymbol{r}_m(m_1^i, m_2^j), ..., \boldsymbol{r}_m(m_{|c_i|}^i, m_{|c_j|}^j)]$ to produce $\boldsymbol{r}_c(c_i, c_j)$. This is done by applying a pooling operation. In particular it concatenates the results of max-pooling and average-pooling, which we found to be slightly more effective than using either one alone:

$$\boldsymbol{r}_c(c_i, c_j)_k = \begin{cases} \max\{\boldsymbol{R}_m(c_i, c_j)_{k,\cdot}\} & \text{for } 0 \le k < d \\ \text{avg}\{\boldsymbol{R}_m(c_i, c_j)_{k-d,\cdot}\} & \text{for } d \le k < 2d \end{cases}$$

4 Mention-Ranking Model

Rather than training a cluster-ranking model from scratch, we first train a mention-ranking model that assigns each mention its highest scoring candidate antecedent. There are two key advantages of doing this. First, it serves as pretraining for the cluster-ranking model; in particular the mention-ranking model learns effective weights for the mention-pair encoder. Second, the scores produced by the mention-ranking model are used to provide a measure of which coreference decisions are easy (allowing for an easy-first clustering strategy) and which decisions are clearly wrong (these decisions can be pruned away, significantly reducing the search space of the cluster-ranking model).

The mention-ranking model assigns a score $s_m(a, m)$ to a mention m and candidate an-

tecedent a representing their compatibility for coreference. This is produced by applying a single fully connected layer of size one to the representation $\boldsymbol{r}_m(a, m)$ produced by the mention-pair encoder:

$$s_m(a, m) = \boldsymbol{W}_m \boldsymbol{r}_m(a, m) + b_m$$

where $\boldsymbol{W}_m$ is a $1 \times d$ weight matrix. At test time, the mention-ranking model links each mention with its highest scoring candidate antecedent.

Training Objective. We train the mention-ranking model with the slack-rescaled max-margin training objective from Wiseman et al. (2015), which encourages separation between the highest scoring true and false antecedents of the current mention. Suppose the training set consists of N mentions $m_1, m_2, ..., m_N$. Let $\mathcal{A}(m_i)$ denote the set of candidate antecedents of a mention m_i (i.e., mentions preceding m_i and NA), and $\mathcal{T}(m_i)$ denote the set of true antecedents of m_i (i.e., mentions preceding m_i that are coreferent with it or $\{\text{NA}\}$ if m_i has no antecedent). Let $\hat{t}_i$ be the highest scoring true antecedent of mention m_i:

$$\hat{t}_i = \operatorname*{argmax}_{t \in \mathcal{T}(m_i)} s_m(t, m_i)$$

Then the loss is given by

$$\sum_{i=1}^{N} \max_{a \in \mathcal{A}(m_i)} \Delta(a, m_i)(1 + s_m(a, m_i) - s_m(\hat{t}_i, m_i))$$

where $\Delta(a, m_i)$ is the mistake-specific cost function

$$\Delta(a, m_i) = \begin{cases} \alpha_{\text{FN}} & \text{if } a = \text{NA} \ \wedge \mathcal{T}(m_i) \neq \{\text{NA}\} \\ \alpha_{\text{FA}} & \text{if } a \neq \text{NA} \ \wedge \mathcal{T}(m_i) = \{\text{NA}\} \\ \alpha_{\text{WL}} & \text{if } a \neq \text{NA} \ \wedge a \notin \mathcal{T}(m_i) \\ 0 & \text{if } a \in \mathcal{T}(m_i) \end{cases}$$

for "false new," "false anaphoric," "wrong link," and correct coreference decisions. The different error penalties allow the system to be tuned for coreference evaluation metrics by biasing it towards making more or fewer coreference links.

Finding Effective Error Penalties. We fix $\alpha_{\text{WL}} = 1.0$ and search for α_{FA} and α_{FN} out of $\{0.1, 0.2, ..., 1.5\}$ with a variant of grid search. Each new trial uses the unexplored set of hyperparameters that has the closest Manhattan distance to the best setting found so far on the dev set. We stopped the search when all immediate neighbors (within 0.1 distance) of the best setting had been explored. We found $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.8, 0.4, 1.0)$ to be best for English and $(\alpha_{\text{FN}}, \alpha_{\text{FA}}, \alpha_{\text{WL}}) = (0.7, 0.4, 1.0)$ to be best for Chinese on the CoNLL 2012 data. We attribute our smaller false new cost from the one used by Wiseman et al. (they set $\alpha_{\text{FN}} = 1.2$) to using more precise mention detection, which results in fewer links to NA.

Training Details. We initialized our word embeddings with 50 dimensional ones produced by `word2vec` (Mikolov et al., 2013) on the Gigaword corpus for English and 64 dimensional ones provided by Polyglot (Al-Rfou et al., 2013) for Chinese. Averaged word embeddings were held fixed during training while the embeddings used for single words were updated. We set our hidden layer sizes to $M_1 = 1000, M_2 = d$ = 500 and minimized the training objective using RMSProp (Hinton and Tieleman, 2012). To regularize the network, we applied L2 regularization to the model weights and dropout (Hinton et al., 2012) with a rate of 0.5 on the word embeddings and the output of each hidden layer.

Pretraining. As in Wiseman et al. (2015), we found that pretraining is crucial for the mention-ranking model's success. We pretrained the network in two stages, minimizing the following objectives from Clark and Manning (2015):

All-Pairs Classification

$$-\sum_{i=1}^{N}[\sum_{t \in \mathcal{T}(m_i)} \log p(t, m_i) + \sum_{f \in \mathcal{F}(m_i)} \log(1 - p(f, m_i))]$$

Top-Pairs Classification

$$-\sum_{i=1}^{N}[\max_{t \in \mathcal{T}(m_i)} \log p(t, m_i) + \min_{f \in \mathcal{F}(m_i)} \log(1 - p(f, m_i))]$$

Where $\mathcal{F}(m_i)$ is the set of false antecedents for m_i and $p(a, m_i) = \textit{sigmoid}(s(a, m_i))$. The top pairs objective is a middle ground between the all-pairs classification and mention ranking objectives: it only processes high-scoring mentions, but is probabilistic rather than max-margin. We first pretrained the network with all-pairs classification for 150 epochs and then with top-pairs classification for 50 epochs. See Section 6.1 for experiments on

the two-stage pretraining.

5 Cluster-Ranking Model

Although a strong coreference system on its own, the mention-ranking model has the disadvantage of only considering local information between pairs of mentions, so it cannot consolidate information at the entity-level. We address this problem by training a cluster-ranking model that scores pairs of clusters instead of pairs of mentions.

Given two clusters of mentions c_i and c_j, the cluster-ranking model produces a score $s_c(c_i, c_j)$ representing their compatibility for coreference. This is produced by applying a single fully connected layer of size one to the representation $\boldsymbol{r}_c(c_i, c_j)$ produced by the cluster-pair encoder:

$$s_c(c_i, c_j) = \boldsymbol{W}_c \boldsymbol{r}_c(c_i, c_j) + b_c$$

where $\boldsymbol{W}_c$ is a $1 \times 2d$ weight matrix. Our cluster-ranking approach also uses a measure of anaphoricity, or how likely it is for a mention m to have an antecedent. This is defined as

$$s_{\text{NA}}(m) = \boldsymbol{W}_{\text{NA}} \boldsymbol{r}_m(\text{NA}, m) + b_{\text{NA}}$$

where $\boldsymbol{W}_{\text{NA}}$ is a $1 \times d$ matrix.

5.1 Cluster-Ranking Policy Network

At test time, the cluster ranker iterates through every mention in the document, merging the current mention's cluster with a preceding one or performing no action. We view this procedure as a sequential decision process where at each step the algorithm observes the current state x and performs some action u.

Specifically, we define a state $x = (C, m)$ to consist of $C = \{c_1, c_2, ...\}$, the set of existing coreference clusters, and m, the current mention being considered. At a start state, each cluster in C contains a single mention. Let $c_m \in C$ be the cluster containing m and $\mathcal{A}(m)$ be a set of candidate antecedents for m: mentions occurring previously in the document. Then the available actions $U(x)$ from x are

- MERGE$[c_m, c]$, where c is a cluster containing a mention in $\mathcal{A}(m)$. This combines c_m and c into a single coreference cluster.
- PASS. This leaves the clustering unchanged.

After determining the new clustering C' based on the existing clustering C and action u, we consider another mention m' to get the next state $x' = (C', m')$.

Using the scoring functions s_c and s_{NA}, we define a policy network π that assigns a probability distribution over $U(x)$ as follows:

$$\pi(\text{MERGE}[c_m, c]|x) \propto e^{s_c(c_m, c)}$$
$$\pi(\text{PASS}|x) \propto e^{s_{\text{NA}}(m)}$$

During inference, π is executed by taking the highest-scoring (most probable) action at each step.

5.2 Easy-First Cluster Ranking

The last detail needed is the ordering in which to consider mentions. Cluster-ranking models in prior work order the mentions according to their positions in the document, processing them left-to-right (Rahman and Ng, 2011; Ma et al., 2014). However, we instead sort the mentions in descending order by their highest scoring candidate coreference link according to the mention-ranking model. This causes inference to occur in an easy-first fashion where hard decisions are delayed until more information is available. Easy-first orderings have been shown to improve the performance of other incremental coreference strategies (Raghunathan et al., 2010; Stoyanov and Eisner, 2012) because they reduce the problem of errors compounding as the algorithm runs.

We also find it beneficial to prune the set of candidate antecedents $\mathcal{A}(m)$ for each mention m. Rather than using all previously occurring mentions as candidate antecedents, we only include high-scoring ones, which greatly reduces the size of the search space. This allows for much faster learning and inference; we are able to remove over 95% of candidate actions with no decrease in the model's performance. For both of these two preprocessing steps, we use $s(a, m) - s(\text{NA}, m)$ as the score of a coreference link between a and m.

5.3 Deep Learning to Search

We face a sequential prediction problem where future observations (visited states) depend on previous actions. This is challenging because it violates the common i.i.d. assumption made in machine learning. Learning-to-search algorithms are effective for this sort of problem, and have been applied successfully to coreference resolution (Daumé III and Marcu, 2005; Clark and Manning, 2015) as

Algorithm 1 Deep Learning to Search

for $i = 1$ **to** num_epochs **do**
  Initialize the current training set $\Gamma = \emptyset$
  for each example $(x, y) \in \mathcal{D}$ **do**
    Run the policy π to completion from start state x to obtain a trajectory of states $\{x_1, x_2, ..., x_n\}$
    for each state x_i in the trajectory **do**
      for each possible action $u \in U(x_i)$ **do**
        Execute u on x_i and then run the reference policy π^{ref} until reaching an end state e
        Assign u a cost by computing the loss on the end state: $l(u) = \mathcal{L}(e, y)$
      end for
      Add the state x_i and associated costs l to Γ
    end for
  end for
  Update π with gradient descent, minimizing $\sum_{(x,l)\in\Gamma} \sum_{u\in U(x)} \pi(u|x)l(u)$.
end for

well as other structured prediction tasks in natural language processing (Daumé III et al., 2014; Chang et al., 2015a).

We train the cluster-ranking model using a learning-to-search algorithm inspired by SEARN (Daumé III et al., 2009), which is described in Algorithm 1. The algorithm takes as input a dataset $\mathcal{D}$ of start states x (in our case documents with each mention in its own singleton coreference cluster) and structured labels y (in our case gold coreference clusters). Its goal is to train the policy π so when it executes from x, reaching a final state e, the resulting loss $\mathcal{L}(e, y)$ is small. We use the negative of the B^3 coreference metric for this loss (Bagga and Baldwin, 1998). Although our system evaluation also includes the MUC (Vilain et al., 1995) and CEAF_{ϕ_4} (Luo, 2005) metrics, we do not incorporate them into the loss because MUC has the flaw of treating all errors equally and CEAF_{ϕ_4} is slow to compute.

For each example $(x, y) \in \mathcal{D}$, the algorithm obtains a trajectory of states $x_1, x_2, ..., x_n$ visited by the current policy by running it to completion (i.e., repeatedly taking the highest scoring action until reaching an end state) from the start state x. This exposes the model to states at train time similar to the ones it will face at test time, allowing it to learn how to cope with mistakes.

Given a state x in a trajectory, the algorithm then assigns a cost $l(u)$ to each action $u \in U(x)$ by executing the action, "rolling out" from the resulting state with a reference policy π^{ref} until reaching an end state e, and computing the resulting loss $\mathcal{L}(e, y)$. This rolling out procedure allows the model to learn how a local action will affect the final score, which cannot be otherwise computed because coreference evaluation metrics do not decompose over cluster merges. The policy network is then trained to minimize the risk associated with taking each action: $\sum_{u\in U(x)} \pi(u|x)l(u)$.

Reference policies typically refer to the gold labels to find actions that are likely to be beneficial. Our reference policy π^{ref} takes the action that increases the B^3 score the most each step, breaking ties randomly. It is generally recommended to use a stochastic mixture of the reference policy and the current learned policy during rollouts when the reference policy is not optimal (Chang et al., 2015b). However, we find only using the reference policy (which is close to optimal) to be much more efficient because it does not require neural network computations and is deterministic, which means the costs of actions can be cached.

Training details. We update π using RMSProp and apply dropout with a rate of 0.5 to the input layer. For most experiments, we initialize the mention-pair encoder component of the cluster-ranking model with the learned weights from the mention-ranking model, which we find to greatly improve performance (see Section 6.2).

Runtime. The full cluster-ranking system runs end-to-end in slightly under 1 second per document on the English test set when using a GPU (including scoring all pairs of mentions with the mention-ranking model for search-space pruning). This means the bottleneck for the overall system is the syntactic parsing required for mention detection (around 4 seconds per document).

Model	English F_1	Chinese F_1
Full Model	65.52	64.41
– MENTION	–1.27	–0.74
– GENRE	–0.25	–2.91
– DISTANCE	–2.42	–2.41
– SPEAKER	–1.26	–0.93
– MATCHING	–2.07	–3.44

Table 1: CoNLL F_1 scores of the mention-ranking model on the dev sets without mention, document genre, distance, speaker, and string matching hand-engineered features.

6 Experiments and Results

Experimental Setup. We run experiments on the English and Chinese portions of the CoNLL 2012 Shared Task data (Pradhan et al., 2012). The models are evaluated using three of the most popular coreference metrics: MUC, B^3, and Entity-based CEAF ($CEAF_{\phi_4}$). We generally report the average F_1 score (CoNLL F_1) of the three, which is common practice in coreference evaluation. We used the most recent version of the CoNLL scorer (version 8.01), which implements the original definitions of the metrics.

Mention Detection. Our experiments were run using system-produced predicted mentions. We used the rule-based mention detection algorithm from Raghunathan et al. (2010), which first extracts pronouns and maximal NP projections as candidate mentions and then filters this set with rules that remove spurious mentions such as numeric entities and pleonastic *it* pronouns.

6.1 Mention-Ranking Model Experiments

Feature Ablations. We performed a feature ablation study to determine the importance of the hand-engineered features included in our model. The results are shown in Table 1. We find the small number of non-embedding features substantially improves model performance, especially the distance and string matching features. This is unsurprising, as the additional features are not easily captured by word embeddings and historically such features have been very important in coreference resolvers (Bengtson and Roth, 2008).

The Importance of Pretraining. We evaluate the benefit of the two-step pretraining for the mention-ranking model and report results in Table 2. Consistent with Wiseman et al. (2015), we find pretraining to greatly improve the model's accuracy. We note in particular that the model benefits from using both pretraining steps from Section 4, which more smoothly transitions the model from a mention-pair classification objective that is easy to optimize to a max-margin objective better suited for a ranking task.

All-Pairs	Top-Pairs	English F_1	Chinese F_1
Yes	Yes	65.52	64.41
Yes	No	–0.36	–0.24
No	Yes	–0.54	–0.33
No	No	–3.58	–5.43

Table 2: CoNLL F_1 scores of the mention-ranking model on the dev sets with different pretraining methods.

Model	English F_1	Chinese F_1
Full Model	66.01	64.86
– PRETRAINING	–5.01	–6.85
– EASY-FIRST	–0.15	–0.12
– L2S	–0.32	–0.25

Table 3: CoNLL F_1 scores of the cluster-ranking model on the dev sets with various ablations. – PRETRAINING: initializing model parameters randomly instead of from the mention-ranking model, – EASY-FIRST: iterating through mentions in order of occurrence instead of according to their highest scoring candidate coreference link, – L2S: training on a fixed trajectory of correct actions instead of using learning to search.

6.2 Cluster-Ranking Model Experiments

We evaluate the importance of three key details of the cluster ranker: initializing it with the mention-ranking model's weights, using an easy-first ordering of mentions, and using learning to search. The results are shown in Table 3.

Pretrained Weights. We compare initializing the cluster-ranking model randomly with initializing it with the weights learned by the mention-ranking model. Using pretrained weights greatly improves performance. We believe the cluster-ranking model has difficulty learning effective weights from scratch due to noise in the signal coming from cluster-level decisions (an overall bad cluster merge may still involve a few cor-

rect pairwise links) and the smaller amount of data used to train the cluster-ranking model (many possible actions are pruned away during preprocessing). We believe the score would be even lower without search-space pruning, which stops the model from considering many bad actions.

Easy-First Cluster Ranking. We compare the effectiveness of easy-first cluster-ranking with the commonly used left-to-right approach. Using a left-to-right strategy simply requires changing the preprocessing step ordering the mentions so mentions are sorted by their position in the document instead of their highest scoring coreference link according to the mention-ranking model. We find the easy-first approach slightly outperforms using a left-to-right ordering of mentions. We believe this is because delaying hard decisions until later reduces the problem of early mistakes causing later decisions to be made incorrectly.

Learning to Search. We also compare learning to search with the simpler approach of training the model on a trajectory of gold coreference decisions (i.e., training on a fixed cost-sensitive classification dataset). Using this approach significantly decreases performance. We attribute this to the model not learning how to deal with mistakes when it only sees correct decisions during training.

6.3 Capturing Semantic Similarity

Using semantic information to improve coreference accuracy has had mixed in results in previous research, and has been called an "uphill battle" in coreference resolution (Durrett and Klein, 2013). However, word embeddings are well known for being effective at capturing semantic relatedness, and we show here that neural network coreference models can take advantage of this.

Perhaps the case where semantic similarity is most important is in linking nominals with no head match (e.g., "the nation" and "the country"). We compare the performance of our neural network model with our earlier statistical system (Clark and Manning, 2015) at classifying mention pairs of this type as being coreferent or not. The neural network shows substantial improvement (18.9 F_1 vs. 10.7 F_1) on this task compared to the more modest improvement it gets at classifying any pair of mentions as coreferent (68.7 F_1 vs. 66.1 F_1). Some example wins are shown in Table 4. These types of coreference links are quite rare in the CoNLL data (about 1.2% of the positive coreference links in the test set), so the improvement does not significantly contribute to the final system's score, but it does suggest progress on this difficult type of coreference problem.

Antecedent	Anaphor
the country's leftist rebels	the guerrillas
the company	the New York firm
the suicide bombing	the attack
the gun	the rifle
the U.S. carrier	the ship

Table 4: Examples of nominal coreferences with no head match that the neural model gets correct, but the system from Clark and Manning (2015) gets incorrect.

6.4 Final System Performance

In Table 5 we compare the results of our system with state-of-the-art approaches for English and Chinese. Our mention-ranking model surpasses all previous systems. We attribute its improvement over the neural mention ranker from Wiseman et al. (2015) to our model using a deeper neural network, pretrained word embeddings, and more sophisticated pretraining.

The cluster-ranking model improves results further across both languages and all evaluation metrics, demonstrating the utility of incorporating entity-level information. The improvement is largest in CEAF_{ϕ_4}, which is encouraging because CEAF_{ϕ_4} is the most recently proposed metric, designed to correct flaws in the other two (Luo, 2005). We believe entity-level information is particularly useful for preventing bad merges between large clusters (see Figure 4 for an example). However, it is worth noting that in practice the much more complicated cluster-ranking model brings only fairly modest gains in performance.

7 Related Work

There has been extensive work on machine learning approaches to coreference resolution (Soon et al., 2001; Ng and Cardie, 2002), with mention-ranking models being particularly popular (Denis and Baldridge, 2007; Durrett and Klein, 2013; Björkelund and Kuhn, 2014).

We train a neural mention-ranking model inspired by Wiseman et al. (2015) as a starting point, but then use it to pretrain a cluster-ranking model that benefits from entity-level information. Wise-

	MUC			B^3			$CEAF_{\phi_4}$			
	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Prec.	Rec.	F_1	Avg. F_1
CoNLL 2012 English Test Data										
Clark and Manning (2015)	76.12	69.38	72.59	65.64	56.01	60.44	59.44	52.98	56.02	63.02
Peng et al. (2015)	–	–	72.22	–	–	60.50	–	–	56.37	63.03
Wiseman et al. (2015)	76.23	69.31	72.60	66.07	55.83	60.52	59.41	54.88	57.05	63.39
Wiseman et al. (2016)	77.49	69.75	73.42	66.83	56.95	61.50	62.14	53.85	57.70	64.21
NN Mention Ranker	79.77	69.10	74.05	69.68	56.37	62.32	63.02	53.59	57.92	64.76
NN Cluster Ranker	78.93	69.75	**74.06**	70.08	56.98	**62.86**	62.48	55.82	**58.96**	**65.29**
CoNLL 2012 Chinese Test Data										
Chen & Ng (2012)	59.92	64.69	62.21	60.26	51.76	55.69	51.61	58.84	54.99	57.63
Björkelund & Kuhn (2014)	69.39	62.57	65.80	61.64	53.87	57.49	59.33	54.65	56.89	60.06
NN Mention Ranker	72.53	65.72	68.96	65.49	56.87	60.88	61.93	57.11	59.42	63.09
NN Cluster Ranker	73.85	65.42	**69.38**	67.53	56.41	**61.47**	62.84	57.62	**60.12**	**63.66**

Table 5: Comparison with the current state-of-the-art approaches on the CoNLL 2012 test sets. NN Mention Ranker and NN Cluster Ranker are contributions of this work.

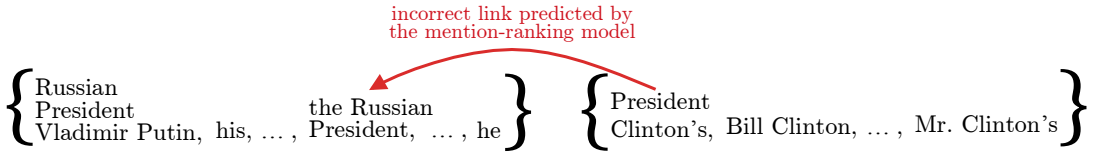


Figure 4: Thanks to entity-level information, the cluster-ranking model correctly declines to merge these two large clusters when running on the test set. However, the mention-ranking model incorrectly links *the Russian President* and *President Clinton's*, which greatly reduces the final precision score.

man et al. (2016) extend their mention-ranking model by incorporating entity-level information produced by a recurrent neural network running over the candidate antecedent-cluster. However, this is an augmentation to a mention-ranking model, and not fundamentally a clustering model as our cluster ranker is.

Entity-level information has also been incorporated in coreference systems using joint inference (McCallum and Wellner, 2003; Poon and Domingos, 2008; Haghighi and Klein, 2010) and systems that build up coreference clusters incrementally (Luo et al., 2004; Yang et al., 2008; Raghunathan et al., 2010). We take the latter approach, and in particular combine the cluster-ranking (Rahman and Ng, 2011; Ma et al., 2014) and easy-first (Stoyanov and Eisner, 2012; Clark and Manning, 2015) clustering strategies. These prior systems all express entity-level information in the form of hand-engineered features and constraints instead of entity-level distributed representations that are learned from data.

We train our system using a learning-to-search algorithm similar to SEARN (Daumé III et al., 2009). Learning-to-search style algorithms have been employed to train coreference resolvers on trajectories of decisions similar to those that would be seen at test-time by Daumé et al. (2005), Ma et al. (2014), and Clark and Manning (2015). Other works use structured perceptron models for the same purpose (Stoyanov and Eisner, 2012; Fernandes et al., 2012; Björkelund and Kuhn, 2014).

8 Conclusion

We have presented a coreference system that captures entity-level information with distributed representations of coreference cluster pairs. These learned, dense, high-dimensional feature vectors provide our cluster-ranking coreference model with a strong ability to distinguish beneficial cluster merges from harmful ones. The model is trained with a learning-to-search algorithm that allows it to learn how local decisions will affect the final coreference score. We evaluate our system on the English and Chinese portions of the CoNLL 2012 Shared Task and report a substantial improvement over the current state-of-the-art.

Acknowledgments

We thank Will Hamilton, Jon Gauthier, and the anonymous reviewers for their thoughtful comments and suggestions. This work was supported by NSF Award IIS-1514268.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. *Conference on Natural Language Learning (CoNLL)*, pages 183–192.

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The First International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference*, pages 563–566.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303.

Anders Björkelund and Jonas Kuhn. 2014. Learning structured perceptrons for coreference resolution with latent antecedents and non-local features. In *Association of Computational Linguistics (ACL)*, pages 47–57.

Kai-Wei Chang, He He, Hal Daumé III, and John Langford. 2015a. Learning to search for dependencies. *arXiv preprint arXiv:1503.05615*.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daumé III, and John Langford. 2015b. Learning to search better than your teacher. In *International Conference on Machine Learning (ICML)*.

Chen Chen and Vincent Ng. 2012. Combining the best of two worlds: A hybrid approach to multilingual coreference resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 56–63.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Association for Computational Linguistics (ACL)*.

Hal Daumé III and Daniel Marcu. 2005. A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 97–104.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325.

Hal Daumé III, John Langford, and Stephane Ross. 2014. Efficient programmable learning to search. *arXiv preprint arXiv:1406.1837*.

Pascal Denis and Jason Baldridge. 2007. A ranking approach to pronoun resolution. In *International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 1588–1593.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1971–1982.

Greg Durrett, David Leo Wright Hall, and Dan Klein. 2013. Decentralized entity-level modeling for coreference resolution. In *Association for Computational Linguistics (ACL)*, pages 114–124.

Eraldo Rezende Fernandes, Cícero Nogueira Dos Santos, and Ruy Luiz Milidiú. 2012. Latent structure perceptron with feature induction for unrestricted coreference resolution. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 41–48.

Aria Haghighi and Dan Klein. 2010. Coreference resolution in a modular, entity-centered model. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*, pages 385–393.

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

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

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

Xiaoqiang Luo, Abe Ittycheriah, Hongyan Jing, Nanda Kambhatla, and Salim Roukos. 2004. A mention-synchronous coreference resolution algorithm based on the Bell tree. In *Association for Computational Linguistics (ACL)*, page 135.

Xiaoqiang Luo. 2005. On coreference resolution performance metrics. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 25–32.

Chao Ma, Janardhan Rao Doppa, J Walker Orr, Prashanth Mannem, Xiaoli Fern, Tom Dietterich, and Prasad Tadepalli. 2014. Prune-and-score: Learning for greedy coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. *Transactions of the Association for Computational Linguistics (TACL)*, 3:405–418.

Andrew McCallum and Ben Wellner. 2003. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI Workshop on Information Integration on the Web*.

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

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *International Conference on Machine Learning (ICML)*, pages 807–814.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Association of Computational Linguistics (ACL)*, pages 104–111.

Haoruo Peng, Kai-Wei Chang, and Dan Roth. 2015. A joint framework for coreference resolution and mention head detection. *Conference on Natural Language Learning (CoNLL)*, 51:12.

Hoifung Poon and Pedro Domingos. 2008. Joint unsupervised coreference resolution with markov logic. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 650–659.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. Conll-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Conference on Computational Natural Language Learning - Shared Task*, pages 1–40.

Karthik Raghunathan, Heeyoung Lee, Sudarshan Rangarajan, Nathanael Chambers, Mihai Surdeanu, Dan Jurafsky, and Christopher Manning. 2010. A multi-pass sieve for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 492–501.

Altaf Rahman and Vincent Ng. 2011. Narrowing the modeling gap: a cluster-ranking approach to coreference resolution. *Journal of Artificial Intelligence Research (JAIR)*, pages 469–521.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Veselin Stoyanov and Jason Eisner. 2012. Easy-first coreference resolution. In *International Conference on Computational Linguistics (COLING)*, pages 2519–2534.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. 1995. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding*, pages 45–52.

Sam Wiseman, Alexander M Rush, Stuart M Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Association of Computational Linguistics (ACL)*, pages 92–100.

Sam Wiseman, Alexander M Rush, Stuart M Shieber, and Jason Weston. 2016. Learning global features for coreference resolution. In *Human Language Technology and North American Association for Computational Linguistics (HLT-NAACL)*.

Xiaofeng Yang, Jian Su, Jun Lang, Chew Lim Tan, Ting Liu, and Sheng Li. 2008. An entity-mention model for coreference resolution with inductive logic programming. In *Association of Computational Linguistics (ACL)*, pages 843–851.

Effects of Creativity and Cluster Tightness on Short Text Clustering Performance

Catherine Finegan-Dollak[1] **Reed Coke**[1] **Rui Zhang**[1]
Xiangyi Ye[2] **Dragomir Radev**[1]
{cfdollak, reedcoke, ryanzh, yexy, radev}@umich.edu
[1]Department of EECS, University of Michigan, Ann Arbor, MI USA
[2]Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI USA

Abstract

Properties of corpora, such as the diversity of vocabulary and how tightly related texts cluster together, impact the best way to cluster short texts. We examine several such properties in a variety of corpora and track their effects on various combinations of similarity metrics and clustering algorithms. We show that semantic similarity metrics outperform traditional n-gram and dependency similarity metrics for k-means clustering of a linguistically creative dataset, but do not help with less creative texts. Yet the choice of similarity metric interacts with the choice of clustering method. We find that graph-based clustering methods perform well on tightly clustered data but poorly on loosely clustered data. Semantic similarity metrics generate loosely clustered output even when applied to a tightly clustered dataset. Thus, the best performing clustering systems could not use semantic metrics.

1 Introduction

Corpora of collective discourse—texts generated by multiple authors in response to the same stimulus—have varying properties depending on the stimulus and goals of the authors. For instance, when multiple puzzle-composers write crossword puzzle clues for the same word, they will try to write creative, unique clues to make the puzzle interesting and challenging; clues for "star" could be "Paparazzi's target" or "Sky light." In contrast, people writing a descriptive caption for a photograph can adopt a less creative style. Corpora may also differ on how similar texts within a particular class are to one another, compared to how similar they are to texts from other classes. For example, entries in a cartoon captioning contest that all relate to the same cartoon may vary widely in subject, while crossword clues for the same word would likely be more tightly clustered.

This paper studies how such text properties affect the best method of clustering short texts. Choosing how to cluster texts involves two major decisions: choosing a similarity metric to determine which texts are alike, and choosing a clustering method to group those texts. We hypothesize that creativity may drive authors to express the same concept in a wide variety of ways, leading to data that can benefit from different similarity metrics than less creative texts. At the same time, we hypothesize that tightly clustered datasets—datasets where each text is much more similar to texts in its cluster than to texts from other clusters—can be clustered by powerful graph-based methods such as Markov Clustering (MCL) and Louvain, which may fail on more loosely clustered data. This paper explores the interaction of these effects.

Recently, distributional semantics has been popular and successful for measuring text similarity (Socher et al., 2011; Cheng and Kartsaklis, 2015; He et al., 2015; Kenter and de Rijke, 2015; Kusner et al., 2015; Ma et al., 2015; Tai et al., 2015; Wang et al., 2015). Word embeddings represent similar words in similar locations in vector space: "cat" is closer to "feline" than to "bird." It would be natural to expect such semantics-based approaches to be useful for clustering, particularly for corpora where authors have tried to express similar ideas in unique ways. And indeed, this paper will show that, depending on the choice of clustering method, semantics-based similarity

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 654–665,
Berlin, Germany, August 7-12, 2016.

measures such as summed word embeddings and deep neural networks can have an advantage over more traditional similarity metrics, such as n-gram counts, n-gram tf-idf vectors, and dependency tree kernels, when applied to creative texts.

However, unlike in most text similarity tasks, in clustering the choice of similarity metric interacts with both the choice of clustering method and the properties of the text. Graph-based clustering techniques can be quite effective in clustering short texts (Rangrej et al., 2011), yet this paper will show that they are sensitive to how tightly clustered the data is. Moreover, the tightness of clusters in a dataset is a property of both the underlying data and the similarity metric. We show that when the underlying data can be clustered tightly enough to use powerful graph-based clustering methods, using semantics-based similarity metrics actually creates a disadvantage compared to methods that rely on the surface form of the text, because semantic metrics reduce tightness.

The remainder of this paper is organized as follows. Section 2 summarizes related work. Section 3 describes four datasets of short texts. In Section 4, we describe the similarity metrics and clustering methods used in our experiments, as well as the evaluation measures. Section 5 shows that semantics-based similarity metrics have some advantage when clustering short texts from the most creative dataset, but ultimately do not perform the best when graph-based clustering is an option. In Section 6, we demonstrate the powerful effect that tightness of clusters has on the best combination of similarity metric and clustering method for a given dataset. Finally, Section 7 draws conclusions.

2 Related Work

The most similar work to the present paper is Shrestha et al. (2012), which acknowledged that the similarity metric and the clustering method could both contribute to clustering results. It compared four similarity methods and also tested four clustering methods. Unlike the present work, it did not consider distributional semantics-based similarity measures or similarity measures that incorporated deep learning. In addition, it reported that the characteristics of the corpora "overshadow[ed] the effect of the similarity measures," making it difficult to conclude that there were any significant differences between the similarity measures.

Several papers address the choice of similarity metric for short text clustering without varying the clustering method. Yan et al. (2012) proposed an alternative term weighting scheme to use in place of tf-idf when clustering using non-negative matrix factorization. King et al. (2013) used the cosine similarity between feature vectors that included context word and part-of-speech features and spelling features and applied Louvain clustering to the resulting graph. Xu et al. (2015) used a convolutional neural network to represent short texts and found that, when used with the k-means clustering algorithm, this deep semantic representation outperformed tf-idf, Laplacian eigenmaps, and average embeddings for clustering.

Other papers focused on choosing the best clustering method for short texts, but kept the similarity metric constant. Rangrej et al. (2011) compared k-means, singular value decomposition, and affinity propagation for tweets, finding affinity propagation the most effective, using tf-idf with cosine similarity or Jaccard for a similarity measure. Errecalde et al. (2010) describe an AntTree-based clustering method. They used the cosine similarity of tf-idf vectors as well. Yin (2013) also use the cosine similarity of tf-idf vectors for a two-stage clustering algorithm for tweets.

One common strategy for short text clustering has been to take advantage of outside sources of knowledge (Banerjee et al., 2007; Wang et al., 2009a; Petersen and Poon, 2011; Rosa et al., 2011; Wang et al., 2014). The present work relies only on the texts themselves, not external information.

3 Datasets

Collective discourse (Qazvinian and Radev, 2011; King et al., 2013) involves multiple writers generating texts in response to the same stimulus. In a corpus of texts relating to several stimuli, it may be desirable to cluster according to which stimulus each text relates to—for instance, grouping all of the news headlines about the same event together. Here, we consider texts triggered by several types of stimuli: photographs that need descriptive captions, cartoons that need humorous captions, and crossword answers that need original clues. Each need shapes the properties of the texts.

Pascal and Flickr Captions. The Pascal Captions dataset (hereinafter PAS) and the 8K ImageFlickr dataset (Rashtchian et al., 2010) are sets of captions solicited from Mechanical Turkers for photographs from Flickr and from the Pat-

tern Analysis, Statistical Modeling, and Computational Learning (PASCAL) Visual Object Classes Challenge (Everingham et al., 2010).

PAS includes twenty categories of images (e.g., dogs, as in Example (1)) and 4998 captions. Each category has fifty images with approximately five captions for each image. We use the category as the gold standard cluster. The 8K ImageFlickr set includes 38,390 captions for 7663 photographs; we treat the image a caption is associated with as the gold standard cluster. To keep dataset sizes comparable, we use a randomly selected subset of 5000 captions (998 clusters) from ImageFlickr (hereinafter FLK).

(1)

"a man walking a small dog on a very wavy beach"
"A person in a large black coats walks a white dog on the beach through rough waves."
"Walking a dog on the edge of the ocean"

This task did not encourage creativity; instructions said to "describe the image in one complete but simple sentence." This could lead to sentences within a cluster being rather similar to each other. However, because photographs may contain overlapping elements—for instance, a photograph in the "bus" category of PAS might also show cars, while a photograph in the "cars" category could also contain a bus—texts in one cluster can also be quite similar to texts from other clusters. Thus, these datasets should not be very tightly clustered.

***New Yorker* Cartoon Captions**. *The New Yorker* magazine has a weekly competition in which readers submit possible captions for a captionless cartoon (Example (2)) (Radev et al., 2015). We use the cartoon each caption is associated with as its gold standard cluster.

The complete dataset includes over 1.9 million captions for 366 cartoons. For this work, we use a total of 5000 captions from 20 randomly selected cartoons as the "TOON" dataset.

(2)

"Objection, Your Honor! Alleged killer whale."
"My client maintains that the penguin had a gun!"
"I demand a change of venue to a maritime court!"

Since caption writers seek to stand out from the crowd, we expect high creativity. This may encourage a more varied vocabulary than the FLK and PAS captions that merely describe the image. We also expect wide variation in the meanings of captions for the same cartoon, due to the different joke senses submitted for each, leading to low intra-cluster similarity. Moreover, some users may submit the same caption for more than one cartoon, so we can expect surprisingly high inter-cluster similarity despite the wide variation in cartoon prompt images. We therefore do not expect TOON to be tightly clustered.

Crossword Clues. A dataset of particularly creative texts is comprised of crossword clues.[1] We use the clues as texts and the answer words as their gold standard cluster; all of the clues in Example (3) belong to the "toe" cluster.

(3) Part of the foot
Little piggy
tic-tac-___
The third O of OOO

The complete crossword clues dataset includes 1.7M different clues corresponding to 174,638 unique answers. The "CLUE" dataset includes 5000 clues corresponding to 20 unique answers selected by randomly choosing answers that have 250 or more unique clues, and then randomly choosing 250 of those clues for each answer.

Since words repeat, crossword authors must be creative to come up with clues that will not bore cruciverbalists. CLUE should thus contain many alternative phrasings for essentially the same idea. At the same time, there is likely to be relatively little overlap between clues for different answers, so CLUE should be tightly clustered.

[1] Collected from `http://crosswordgiant.com/`

4 Method

Here we describe the similarity metrics and clustering methods, as well as evaluation measures.

4.1 Similarity Metrics

We hypothesize that creative texts with wide vocabularies will benefit from similarity metrics based on semantic representation of the text, rather than its surface form. We therefore compare three metrics that rely on surface forms of words—n-gram count vectors, tf-idf vectors, and dependency tree segment counts—to three semantic ones—summed Word2Vec embeddings, LSTM autoencoders, and skip-thought vectors. In each case, we represent texts as vectors and find their cosine similarities; if cosine similarity can be negative, we add one and normalize by two to ensure similarity in the range [0, 1].

N-Gram Counts. First we consider n-gram count vectors. We use three variations: (1) unigrams, (2) unigrams and bigrams, and (3) unigrams, bigrams, and trigrams.

N-Gram tf-idf. We also consider weighting n-grams by tf-idf, as calculated by sklearn (Pedregosa et al., 2011).

Dependency Counts. Grammatical information has been found to be useful in text, particularly short text, similarity. (Liu and Gildea, 2005; Zhang et al., 2005; Wang et al., 2009b; Heilman and Smith, 2010; Tian et al., 2010; Šarić et al., 2012; Tai et al., 2015). To leverage this information, previous work has used dependency kernels (Tian et al., 2010), which measure similarity by the fraction of identical dependency parse segments between two sentences. Here, we accomplish the same effect using a count vector for each sentence, with the dependency parse segments as the vocabulary. We define the set of segments for a dependency parse to consist of, for each word, the word, its parent, and the dependency relation that connects them as shown in Example (4).

(4) Part of shoe
- a. Segment 1: (part, ROOT, nsubj)
- b. Segment 2: (of, part, prep)
- c. Segment 3: (shoe, of, pobj)

Word2Vec. For each word, we obtain, if possible, a vector learned via Word2Vec (Mikolov et al., 2013) from the Google News corpus.[2] We represent a sentence as the normalized sum of its word vectors.

LSTM Autoencoder. We use Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) to build another semantics-based sentence representation. We train an LSTM autoencoder consisting of an encoder network and a decoder network. The encoder reads the input sentence and produces a single vector as the hidden state at the last time step. The decoder takes this hidden state vector as input and attempts to reconstruct the original sentence. The LSTM autoencoder is trained to minimize the reconstruction loss. After training, we extract the hidden state at the last time step of encoder as the vector representation for a sentence. We use 300-dimensional word2vec vectors pretrained on GoogleNews and generate 300-dimensional hidden vectors. LSTM autoencoders are separately trained for each dataset with 20% for validation.

Skip-thoughts (Kiros et al., 2015) trains encoder-decoder Recurrent Neural Networks (RNN) without supervision to predict the next and the previous sentences given the current sentence. The pretrained skip-thought model computes vectors as sentence representations.

4.2 Clustering Methods

We explore five clustering methods: k-means, spectral, affinity propagation, Louvain, and MCL.

K-means is a popular and straightforward clustering algorithm (Berkhin, 2006) that takes a parameter k, the number of clusters, and uses an expectation-maximization approach to find k centroids in the data. In the expectation phase points are assigned to their nearest cluster centroid. In the maximization phase the centroids of are recomputed for each cluster of assigned points. K-means is not a graph-based clustering algorithm, but rather operates in a vector space.

Spectral clustering (Donath and Hoffman, 1973; Shi and Malik, 2000; Ng et al., 2001) is a graph-based clustering approach that finds the graph Laplacian of a similarity matrix, builds a matrix of the first k eigenvectors of the Laplacian, and then applies further clustering to this matrix. The method can be viewed as an approximation of a normalized min-cuts algorithm or of a random walks approach. We use the default implementation provided by sklearn, which applies a Gaussian kernel to determine the graph Laplacian and uses

[2] https://code.google.com/archive/p/word2vec/

k-means for the subsequent clustering step.

Affinity propagation finds exemplars for each cluster and then assigns nodes to a cluster based on these exemplars (Frey and Dueck, 2007). This involves updating two matrices R and A, respectively representing the responsibility and availability of each node. A high value for $R_{(i,k)}$ indicates that node x_i would be a good exemplar for cluster k. A high value for $A_{(i,k)}$ indicates that node x_i is likely to belong to cluster k. We use the default implementation provided by sklearn.

Louvain initializes each node to be its own cluster, then greedily maximizes modularity (Section 6.1) by iteratively merging clusters that are highly interconnected (Blondel et al., 2008).

Markov Cluster Algorithm (MCL) simulates flow on a network via random walk (Van Dongen, 2000). The sequence of nodes is represented via a Markov chain. By applying inflation to the transition matrix, the algorithm can maintain the cluster structure pronounced in the transition matrix of this random walk—a structure that would otherwise disappear over time.[3]

4.3 Evaluation Methods

Adjusted Rand Index We use the sklearn implementation of the Adjusted Rand Index (ARI)[4] (Hubert and Arabie, 1985):

$$\text{ARI} = \frac{RI - \text{Expected RI}}{\max RI - \text{Expected RI}} \quad (1)$$

where RI is the Rand Index,

$$RI = \frac{TP + TN}{TP + FP + FN + TN} \quad (2)$$

TP is the number of true positives, TN is true negatives, and FP and FN are false positives and false negatives, respectively. The Rand Index ranges from 0 to 1. ARI adjusts the Rand Index for chance, so that the score ranges from -1 to 1. Random labeling will achieve an ARI score close to 0; perfect labeling achieves an ARI of 1.

Purity is a score in the $[0, 1]$ range that indicates to what extent sentences in the same predicted cluster actually belong to the same cluster. Given $\Omega = \{\omega_1, \omega_2, ..., \omega_K\}$, the predicted clusters, $C = \{c_1, c_2, ..., c_J\}$, the true clusters, and N, the number of examples, purity is

$$\text{Purity}(\Omega, C) = \frac{1}{N} \sum_{k \in K} \max_{j \in J} |\omega_k \cap c_j| \quad (3)$$

[3]We use the implementation from `http://micans.org/mcl/` with inflation=2.0.

[4]Equivalent to Cohen's Kappa (Warrens, 2008).

Normalized Mutual Information (NMI). We use the sklearn implementation of NMI:

$$NMI(\Omega, C) = \frac{MI(\Omega, C)}{\sqrt{H(C) \cdot H(\Omega)}} \quad (4)$$

The numerator is the mutual information (MI) of predicted cluster labels Ω and true cluster labels C. MI describes how much knowing what the predicted clusters are increases knowledge about what the actual classes are. Using marginal entropy ($H(x)$), NMI normalizes MI so that it ranges from 0 to 1. If C and Ω are identical—that is, if the clusters are perfect—NMI will be 1.

5 Vocabulary Width

5.1 Descriptive Statistics for Vocabulary Width

We predict that creative texts have a wider vocabulary than functional texts. We use two measures to reflect this wide vocabulary: the type/token ratio in the dataset (TTR), and that ratio normalized by the mean length of a text in the dataset.

TTR is an obvious estimate of the width of the vocabulary of a corpus. However, all other things being equal, a corpus of many very short texts triggered by the same stimulus would have more repeated words, proportional to the total number of tokens in the corpus, than would a corpus of a smaller number of longer texts. We might therefore normalize the ratio of types to tokens by dividing by the mean length of a text in the dataset, leading to the normalized type-to-token ratio (NTTR) and TTR values shown in Table 1.

	CLUE	TOON	PAS	FLK
TTR	0.1680	0.1064	0.0625	0.0561
NTTR	0.0377	0.0086	0.0058	0.0047

Table 1: Vocabulary properties of each dataset

FLK, PAS, and CLUE conform to expectations. The creative CLUE has TTR more than double that of the more functional PAS and FLK. The effect is more pronounced using NTTR. Surprisingly, TOON falls closer to the PAS and FLK end of the spectrum, suggesting that vocabulary width does not capture the creativity in the captioning competition; perhaps the creativity of cartoon captions is about expressing different ideas, rather than finding unique ways to express the same idea. For the experiments based on vocabulary width, we therefore compare PAS and CLUE.

5.2 Experiments

We hypothesize that if a dataset uses a wide variety of words to express the same ideas, similarity metrics that rely on the surface form of the sentence will be at a disadvantage compared to similarity metrics based in distributional semantics. Thus, word2vec, LSTM autoencoders, and skip-thoughts ought to perform better than the n-gram-based methods and dependency count method when applied to CLUE, but should enjoy no advantage when applied to PAS.

We begin by comparing the performance of all similarity metrics on PAS and CLUE, using k-means for clustering. We then also examine their performance with MCL.

5.3 Results and Discussion

Table 2 compares the performance of all similarity metrics on PAS and CLUE using k-means and MCL. Using k-means on PAS, the unigram tf-idf similarity metric gives the strongest performance for purity and NMI and came in a close second for ARI. LSTM slightly outperformed the other similarity metrics on ARI, but had middle-of-the-road results on the other evaluations. Overall, the semantics-based similarity metrics gave reasonable but not exceptional ARI and purity results, but were at the low end on NMI. This is consistent with our hypothesis that when authors are not trying to express creativity by using a wider vocabulary, surface-based similarity metrics suffice.

For k-means on CLUE, the picture is quite different: the semantics-based similarity metrics markedly outperformed any other similarity metric on ARI. LSTM also provides the best purity score, followed by skip-thought. The semantics-based metrics do not stand out for NMI, though. Based on these results, we conclude that semantics-based measures provide a significant advantage over traditional similarity metrics when using k-means on the wide-vocabulary, creative CLUE.

When clustering with MCL, however, the semantics-based methods perform exceptionally poorly on both datasets. Interestingly, the n-gram-based similarity metrics performed very well when paired with MCL on CLUE—outperforming the best of the k-means scores—while the same metrics performed terribly with MCL on PAS.

We hypothesize that the semantics-based similarity metrics produce less tightly clustered data than the surface-form-based metrics do, and that this may make clustering difficult for some graph-based clustering methods. The next section describes how we test this hypothesis.

6 Tightness of Clusters

6.1 Descriptive Statistics for Tightness

Two pieces contribute to cluster tightness: the dataset itself and the choice of similarity metric. To illustrate, we represent each text with the vector for its similarity metric—for instance, the sum of its word2vec vectors or the unigram tf-idf vector—and reduce it to two dimensions using linear discriminant analysis. We plot five randomly selected gold standard clusters. Plots for unigram tf-idf and word2vec representations of PAS and CLUE are shown in Figures 1 and 2. These support the intuition that semantics-based similarity metrics are not as tightly clustered as n-gram-based metrics. Note also that the CLUE unigram tf-idf clusters appear tighter than the PAS unigram tf-idf clusters.

To quantify this, we compute modularity (Newman, 2004; Newman, 2006):[5]

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \quad (5)$$

A_{ij} is the edge weight between nodes i and j. $\delta(c_i, c_j)$ indicates whether i and j belong to the same cluster. m is the number of edges. k_i is the degree of vertex i, so $\frac{k_i k_j}{2m}$ is the expected number of edges between i and j in a random graph. Thus, modularity is highest when nodes in a cluster are highly interconnected, but sparsely connected to nodes in different clusters. We use this statistic in an unconventional way, determining the modularity of the golden clusters.

Table 3 shows the modularities for all four datasets using the unigram, trigram, unigram tf-idf, trigram tf-idf, dependency, word2vec, and skipthoughts similarity metrics. As suggested by Figures 1 and 2, the CLUE surface-form-based similarities have the highest modularity by far. The surface-form-based similarities for all datasets have much higher modularity than any of the semantics-based similarities; indeed, the

[5]Newman (2010) notes that modularity for even a perfectly mixed network generally cannot be 1 and describes a normalized modularity formula. We calculated both normalized and non-normalized modularity and found the pattern of results to be the same, so we report only modularity.

	k-Means						MCL					
	PAS			CLUE			PAS			CLUE		
Metric	ARI	Purity	NMI	ARI	Purity	NMI	ARI	Purity	NMI	ARI	Purity	NMI
Unigram	0.0286	0.141	0.110	0.0137	0.173	0.153	1.00E-05	0.058	0.051	0.0620	0.527	0.439
Bigram	0.0230	0.143	0.111	0.0124	0.165	0.142	2.50E-05	0.065	0.070	0.0835	0.585	0.465
Trigram	0.0289	0.139	0.108	0.0148	0.178	0.156	3.60E-05	0.069	0.081	0.1034	0.608	0.478
Uni. tf-idf	0.0445	**0.189**	**0.169**	0.0180	0.202	0.188	2.20E-05	0.061	0.060	**0.1482**	0.643	0.506
Bi. tf-idf	0.0287	0.158	0.135	0.0156	0.205	0.205	3.86E-04	0.104	0.135	0.1327	0.722	0.544
Tri. tf-idf	0.0345	0.176	0.142	0.0134	0.195	**0.213**	6.49E-03	0.212	0.230	0.1280	**0.751**	**0.561**
Dependency	0.0122	0.131	0.104	0.0071	0.169	0.207	**2.07E-02**	**0.280**	**0.264**	0.0832	0.745	0.543
Word2Vec	0.0274	0.142	0.103	0.0527	0.189	0.165	0.000	0.050	0.000	0.0000	0.050	0.000
LSTM	**0.0453**	0.170	0.142	**0.0837**	**0.240**	0.202	0.000	0.050	0.000	0.0000	0.050	0.000
Skipthought	0.0311	0.140	0.106	0.0691	0.215	0.180	0.000	0.050	0.000	0.0000	0.050	0.0009

Table 2: A comparison of all similarity metrics on PAS and CLUE datasets, clustered using k-means and MCL. For all evaluations, higher scores are better.

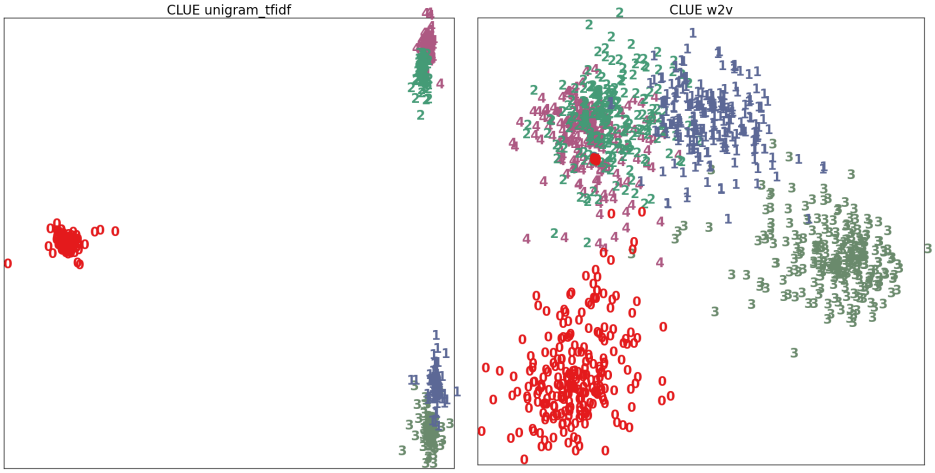


Figure 1: Plots of unigram tf-idf (left) and word2vec (right) vectors representing five randomly selected clusters of CLUE: clues for words "ets," "stay," "yes," "easel," and "aha."

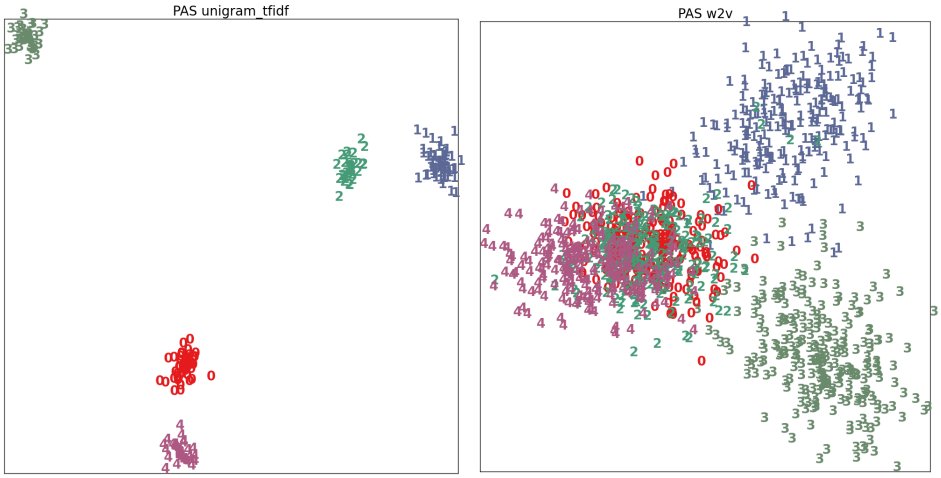


Figure 2: Plots of unigram tf-idf (left) and word2vec (right) vectors representing five randomly selected clusters of PAS: images containing "bus," "boat," "car," "bird," and "motorbike."

semantics-based similarities rarely have modularity much higher than zero. Thus, we conclude both that CLUE is more tightly clustered than the other datasets and that surface-form-based measures yield tighter clusters than semantics-based measures.

CLUE's tight clustering could be due in part to its particularly short texts. Additionally, it might reflect the semantics of the dataset: words that the clues hint at may be less similar to one another than the categories in PAS are to each other. For instance, some images in PAS's "bus" category include cars, and vice-versa.

The difference between semantics-based and surface-form similarity metrics likely arises from the fact that similarity of a word pair is a yes-or-

Metric	PAS	Clues	TOON	FLK
Unigram	0.0254	0.1849	0.0214	0.0065
Bigram	0.0312	0.2216	0.0293	0.0103
Trigram	0.0347	0.2447	0.0352	0.0135
Uni. tf-idf	0.0587	0.3005	0.0519	0.0184
Bi. tf-idf	**0.0877**	0.3875	0.0950	0.0394
Tri. tf-idf	0.0347	0.4339	**0.1311**	**0.0618**
Dependency	0.0799	**0.4729**	0.0451	0.0299
Word2Vec	0.0020	0.0036	0.0008	0.0004
LSTM	0.0072	0.0121	0.0020	0.0009
Skipthought	0.0009	0.0028	0.0006	0.0003

Table 3: Modularity for all datasets

no question to surface-form-based metrics, but a question of degree to semantics-based ones. According to semantics-based methods, "cat" is more similar to "feline" than it is to "dog," but more similar to "dog" than to "motorcycle." This creates some similarity between texts from different clusters, blurring the lines between them. Thus, "The man walks his dog" and "A woman with a cat" are entirely dissimilar according to surface form methods, but not according to the semantics-based measures. Even if the nodes in a cluster are highly interconnected, if the connections between nodes in different clusters are too strong, modularity will be low.

To determine whether cluster tightness influences the best clustering method, we tested all clustering methods on all four datasets using unigram, trigram, unigram tf-idf, trigram tf-idf, word2vec, and skipthought similarity metrics.

6.2 Results and Discussion

As can be seen in Figure 3, the best ARI results by a large margin were those on the tightly clustered CLUE. Louvain, which provides the best ARI for CLUE, and MCL, which provides the second best, both performed most strongly when paired with the surface-form-based similarity metrics (n-gram counts, tf-idf, and dependency count), which had high modularity relative to the semantics-based metrics. Although CLUE also differs from the other datasets in that it has the shortest mean text length, text length by itself cannot explain the observed differences in performance, since the pattern of graph-based clustering methods working best with modular data is consistent within each dataset as well as between datasets.

CLUE is also the only dataset where the semantics-based similarity metrics performed exceptionally well with any of the clustering methods. Recall from Table 1 that CLUE had a markedly wider vocabulary than any other dataset. This further supports our findings in Section 5.3 regarding how creativity affects the usefulness of semantics-based similarity metrics.

FLK, which had the lowest modularity, cannot be clustered by the spectral, Louvain, or MCL algorithms. K-means provides the strongest performance, followed by affinity propagation.

TOON has the worst ARI results. Its best-performing clustering methods are the graph-based Louvain and MCL methods. Both perform well only when paired with the most modular similarity metrics. Louvain seems less sensitive to modularity than MCL does. MCL's best performance by far for TOON is when it is paired with trigram tf-idf, which also had the highest modularity; its performance when paired with the lower-modularity similarity metrics rapidly falls away. In contrast, Louvain fares reasonably well with the lower n-gram tf-idfs, which also had lower modularity than trigram tf-idf.

Louvain and MCL follow a similar pattern on PAS: both perform at their peak on the most modular similarity metric (dependency), but Louvain handles slightly less modular similarity metrics nearly as well as the most modular one, while MCL quickly falters.

K-means' performance is not correlated with modularity. This makes sense, as k-means is the only non-graph-based method. Methods like MCL, which is based on a random walk, may be stymied by too many highly-weighted paths between clusters; the random walk can too easily reach a different neighborhood from where it started. But k-means relies on how close texts are to centroids, not to other texts, and so would be less affected.

The fact that k-means nevertheless performs poorly on TOON suggests that this dataset may be particularly difficult to cluster. An interesting test would be to measure inter-annotator agreement on TOON.

7 Conclusions and Future Work

This work has shown that creativity can influence the best way to cluster text. When using k-means to cluster a dataset where authors tried to be creative, similarity metrics utilizing distri-

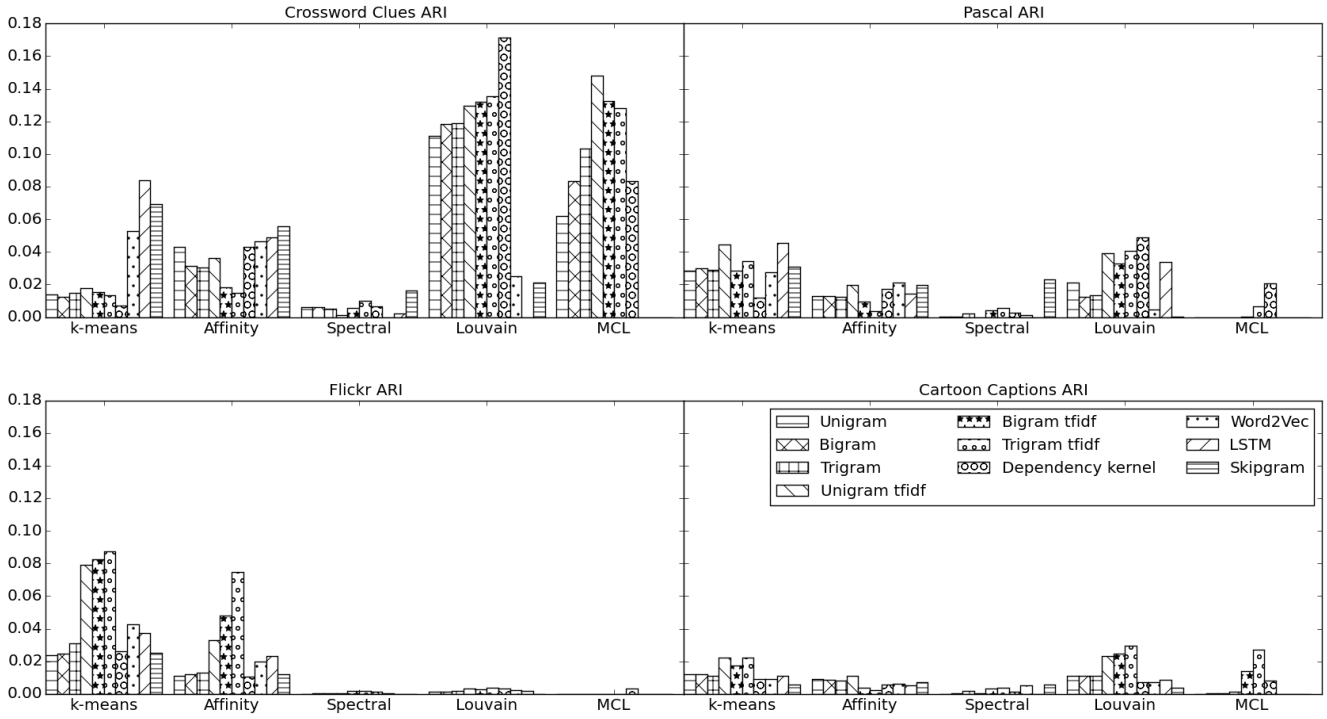


Figure 3: All similarity metrics and all clustering methods for the four datasets.

butional semantics outperformed those that relied on surface forms. We also showed that semantics-based methods do not provide a notable advantage when applying k-means to less creative datasets. Since traditional similarity metrics are often faster to calculate, use of slower semantics-based methods should be limited to creative datasets.

Unlike most work on clustering short texts, we examined how the similarity metric interacts with the clustering method. Even for a creative dataset, if the underlying data is tightly clustered, the use of semantics-based similarity measures can actually hurt performance. Traditional metrics applied to such tightly clustered data generate more modular output that enables the use of sophisticated, graph-based clustering methods such as MCL and Louvain. When either the underlying data or the similarity metrics applied to it produce loose clusters with low modularity, the sophisticated graph clustering algorithms fail, and we must fall back on simpler methods.

Future work can manipulate datasets' text properties to confirm that a specific property is the cause of observed differences in clustering. Such work should alter the datasets TTR and NTTR while holding mean length of texts constant. A pilot effort to use word embeddings to alter the variety of vocabulary in a dataset has so far not succeeded, but future experiments altering vocabulary width or modularity of a dataset and finding that the modified dataset behaved like natural datasets with the same properties could increase confidence in causality. Future work can also explore finer clusters within these datasets, such as clustering CLUE by word sense of the answers and TOON by joke sense.

These results are a first step towards determining the best way to cluster a new dataset based on properties of the text. Future work will explore further how the goals of short text authors translate into measurable properties of the texts they write, and how measuring those properties can help predict which similarity metrics and clustering methods will combine to provide the best performance.

Acknowledgments

The authors are grateful for the help and insights of Jonathan Kummerfeld, Spruce Bondera, Kyle Bouwens, Kurt McEwan, Francisco Rivera Reyes, Clayton Thorrez, Chongruo Wu, Yue Xu, Harry Zhang, and Joel Tetreault. We appreciate the comments of the anonymous reviewers, which helped us improve the paper. Xiangyi Ye was sponsored by the University of Michigan Undergraduate Research Opportunity Program (UROP).

References

Somnath Banerjee, Krishnan Ramanathan, and Ajay Gupta. 2007. Clustering short texts using Wikipedia. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 787–788. ACM.

Pavel Berkhin. 2006. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, pages 25–71. Springer.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008.

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. *arXiv preprint arXiv:1508.02354*.

William E. Donath and Alan J. Hoffman. 1973. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425.

Marcelo Luis Errecalde, Diego Alejandro Ingaramo, and Paolo Rosso. 2010. A new anttree-based algorithm for clustering short-text corpora. *Journal of Computer Science & Technology*, 10.

Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. 2010. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338.

Brendan J. Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *Science*, 315(5814):972–976.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1011–1019, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of Classification*, 2(1):193–218.

Tom Kenter and Maarten de Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1411–1420. ACM.

Ben King, Rahul Jha, Dragomir R Radev, and Robert Mankoff. 2013. Random walk factoid annotation for collective discourse. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*.

Ryan Kiros, Yukun Zhu, Ruslan R. Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 957–966.

Ding Liu and Daniel Gildea. 2005. Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32.

Chenglong Ma, Weiqun Xu, Peijia Li, and Yonghong Yan. 2015. Distributional representations of words for short text classification. In *Proceedings of NAACL-HLT*, pages 33–38.

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

Mark E. J. Newman. 2004. Analysis of weighted networks. *Physical Review E*, 70(5):056131.

Mark E. J. Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

Mark Newman. 2010. *Networks: An Introduction*. Oxford University Press.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856. MIT Press.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Henry Petersen and Josiah Poon. 2011. Enhancing short text clustering with small external repositories. In *Proceedings of the Ninth Australasian Data Mining Conference-Volume 121*, pages 79–90. Australian Computer Society, Inc.

Vahed Qazvinian and Dragomir R. Radev. 2011. Learning from collective human behavior to introduce diversity in lexical choice. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 1098–1108, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dragomir R. Radev, Amanda Stent, Joel R. Tetreault, Aasish Pappu, Aikaterini Iliakopoulou, Agustin Chanfreau, Paloma de Juan, Jordi Vallmitjana, Alejandro Jaimes, Rahul Jha, and Robert Mankoff. 2015. Humor in collective discourse: Unsupervised funniness detection in the new yorker cartoon caption contest. *CoRR*, abs/1506.08126.

Aniket Rangrej, Sayali Kulkarni, and Ashish V. Tendulkar. 2011. Comparative study of clustering techniques for short text documents. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 111–112, New York, NY, USA. ACM.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 139–147. Association for Computational Linguistics.

Kevin Dela Rosa, Rushin Shah, Bo Lin, Anatole Gershman, and Robert Frederking. 2011. Topical clustering of tweets. *Proceedings of the ACM SIGIR: SWSM*.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, pages 441–448. Association for Computational Linguistics.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. Technical report.

Prajol Shrestha, Christine Jacquin, and Béatrice Daille, 2012. *Computational Linguistics and Intelligent Text Processing: 13th International Conference, CICLing 2012, New Delhi, India, March 11-17, 2012, Proceedings, Part II*, chapter Clustering Short Text and Its Evaluation, pages 169–180. Springer Berlin Heidelberg, Berlin, Heidelberg.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Y. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Yun Tian, Haisheng Li, Qiang Cai, and Shouxiang Zhao. 2010. Measuring the similarity of short texts by word similarity and tree kernels. In *Information Computing and Telecommunications (YC-ICT), 2010 IEEE Youth Conference on*, pages 363–366. IEEE.

Stijn Van Dongen. 2000. *Graph Clustering by Flow Simulation*. Ph.D. thesis, University of Utrecht.

Jun Wang, Yiming Zhou, Lin Li, Biyun Hu, and Xia Hu. 2009a. Improving short text clustering performance with keyword expansion. In *The Sixth International Symposium on Neural Networks (ISNN 2009)*, pages 291–298. Springer.

Kai Wang, Zhaoyan Ming, and Tat-Seng Chua. 2009b. A syntactic tree matching approach to finding similar questions in community-based qa services. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 187–194, New York, NY, USA. ACM.

Yu Wang, Lihui Wu, and Hongyu Shao. 2014. Clusters merging method for short texts clustering. *Open Journal of Social Sciences*, 2(09):186.

Peng Wang, Jiaming Xu, Bo Xu, Cheng-Lin Liu, Heng Zhang, Fangyuan Wang, and Hongwei Hao. 2015. Semantic clustering and convolutional neural network for short text categorization. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 2, pages 352–357.

Matthijs J. Warrens. 2008. On the equivalence of cohen's kappa and the hubert-arabie adjusted rand index. *Journal of Classification*, 25(2):177–183.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Short text clustering via convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 62–69.

Xiaohui Yan, Jiafeng Guo, Shenghua Liu, Xue-qi Cheng, and Yanfeng Wang. 2012. Clustering short text using ncut-weighted non-negative matrix factorization. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 2259–2262. ACM.

Jie Yin. 2013. Clustering microtext streams for event identification. In *IJCNLP*, pages 719–725.

Min Zhang, Jian Su, Danmei Wang, Guodong Zhou, and Chew Lim Tan. 2005. Discovering relations between named entities from a large raw corpus using tree similarity-based clustering. In *Natural Language Processing–IJCNLP 2005*, pages 378–389. Springer.

Generative Topic Embedding: a Continuous Representation of Documents

Shaohua Li[1,2] **Tat-Seng Chua**[1] **Jun Zhu**[3] **Chunyan Miao**[2]
shaohua@gmail.com dcscts@nus.edu.sg dcszj@tsinghua.edu.cn ascymiao@ntu.edu.sg
1. School of Computing, National University of Singapore
2. Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY)
3. Department of Computer Science and Technology, Tsinghua University

Abstract

Word embedding maps words into a low-dimensional continuous embedding space by exploiting the local word collocation patterns in a small context window. On the other hand, topic modeling maps documents onto a low-dimensional topic space, by utilizing the global word collocation patterns in the same document. These two types of patterns are complementary. In this paper, we propose a generative topic embedding model to combine the two types of patterns. In our model, topics are represented by embedding vectors, and are shared across documents. The probability of each word is influenced by both its local context and its topic. A variational inference method yields the topic embeddings as well as the topic mixing proportions for each document. Jointly they represent the document in a low-dimensional continuous space. In two document classification tasks, our method performs better than eight existing methods, with fewer features. In addition, we illustrate with an example that our method can generate coherent topics even based on only one document.

1 Introduction

Representing documents as fixed-length feature vectors is important for many document processing algorithms. Traditionally documents are represented as a bag-of-words (BOW) vectors. However, this simple representation suffers from being high-dimensional and highly sparse, and loses semantic relatedness across the vector dimensions.

Word Embedding methods have been demonstrated to be an effective way to represent words as continuous vectors in a low-dimensional embedding space (Bengio et al., 2003; Mikolov et al., 2013; Pennington et al., 2014; Levy et al., 2015). The learned embedding for a word encodes its semantic/syntactic relatedness with other words, by utilizing local word collocation patterns. In each method, one core component is the *embedding link function*, which predicts a word's distribution given its context words, parameterized by their embeddings.

When it comes to documents, we wish to find a method to encode their overall semantics. Given the embeddings of each word in a document, we can imagine the document as a "bag-of-vectors". Related words in the document point in similar directions, forming *semantic clusters*. The centroid of a semantic cluster corresponds to the most representative embedding of this cluster of words, referred to as the *semantic centroids*. We could use these semantic centroids and the number of words around them to represent a document.

In addition, for a set of documents in a particular domain, some semantic clusters may appear in many documents. By learning collocation patterns across the documents, the derived semantic centroids could be more topical and less noisy.

Topic Models, represented by Latent Dirichlet Allocation (LDA) (Blei et al., 2003), are able to group words into topics according to their collocation patterns across documents. When the corpus is large enough, such patterns reflect their semantic relatedness, hence topic models can discover coherent topics. The probability of a word is governed by its latent topic, which is modeled as a categorical distribution in LDA. Typically, only a small number of topics are present in each document, and only a small number of words have high probability in each topic. This intuition motivated Blei et al. (2003) to regularize the topic distributions with Dirichlet priors.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 666–675,
Berlin, Germany, August 7-12, 2016.

Semantic centroids have the same nature as topics in LDA, except that the former exist in the embedding space. This similarity drives us to seek the common semantic centroids with a model similar to LDA. We extend a generative word embedding model PSDVec (Li et al., 2015), by incorporating topics into it. The new model is named TopicVec. In TopicVec, an embedding link function models the word distribution in a topic, in place of the categorical distribution in LDA. The advantage of the link function is that the semantic relatedness is already encoded as the *cosine* distance in the embedding space. Similar to LDA, we regularize the topic distributions with Dirichlet priors. A variational inference algorithm is derived. The learning process derives *topic embeddings* in the same embedding space of words. These topic embeddings aim to approximate the underlying semantic centroids.

To evaluate how well TopicVec represents documents, we performed two document classification tasks against eight existing topic modeling or document representation methods. Two setups of TopicVec outperformed all other methods on two tasks, respectively, with fewer features. In addition, we demonstrate that TopicVec can derive coherent topics based only on *one* document, which is not possible for topic models.

The source code of our implementation is available at https://github.com/askerlee/topicvec.

2 Related Work

Li et al. (2015) proposed a generative word embedding method PSDVec, which is the precursor of TopicVec. PSDVec assumes that the conditional distribution of a word given its context words can be factorized approximately into independent log-bilinear terms. In addition, the word embeddings and regression residuals are regularized by Gaussian priors, reducing their chance of overfitting. The model inference is approached by an efficient Eigendecomposition and blockwise-regression method (Li et al., 2016b). TopicVec differs from PSDVec in that in the conditional distribution of a word, it is not only influenced by its context words, but also by a topic, which is an embedding vector indexed by a latent variable drawn from a Dirichlet-Multinomial distribution.

Hinton and Salakhutdinov (2009) proposed to model topics as a certain number of binary hidden variables, which interact with all words in the document through weighted connections. Larochelle and Lauly (2012) assigned each word a unique topic vector, which is a summarization of the context of the current word.

Huang et al. (2012) proposed to incorporate global (document-level) semantic information to help the learning of word embeddings. The global embedding is simply a weighted average of the embeddings of words in the document.

Le and Mikolov (2014) proposed Paragraph Vector. It assumes each piece of text has a latent paragraph vector, which influences the distributions of all words in this text, in the same way as a latent word. It can be viewed as a special case of TopicVec, with the topic number set to 1. Typically, however, a document consists of multiple semantic centroids, and the limitation of only one topic may lead to underfitting.

Nguyen et al. (2015) proposed Latent Feature Topic Modeling (LFTM), which extends LDA to incorporate word embeddings as latent features. The topic is modeled as a mixture of the conventional categorical distribution and an embedding link function. The coupling between these two components makes the inference difficult. They designed a Gibbs sampler for model inference. Their implementation[1] is slow and infeasible when applied to a large corpous.

Liu et al. (2015) proposed Topical Word Embedding (TWE), which combines word embedding with LDA in a simple and effective way. They train word embeddings and a topic model separately on the same corpus, and then average the embeddings of words in the same topic to get the embedding of this topic. The topic embedding is concatenated with the word embedding to form the topical word embedding of a word. In the end, the topical word embeddings of all words in a document are averaged to be the embedding of the document. This method performs well on our two classification tasks. Weaknesses of TWE include: 1) the way to combine the results of word embedding and LDA lacks statistical foundations; 2) the LDA module requires a large corpus to derive semantically coherent topics.

Das et al. (2015) proposed Gaussian LDA. It uses pre-trained word embeddings. It assumes that words in a topic are random samples from a multivariate Gaussian distribution with the topic embedding as the mean. Hence the probability that a

[1] https://github.com/datquocnguyen/LFTM/

Name	Description
$\boldsymbol{S}$	Vocabulary $\{s_1, \cdots, s_W\}$
$\boldsymbol{V}$	Embedding matrix $(\boldsymbol{v}_{s_1}, \cdots, \boldsymbol{v}_{s_W})$
$\boldsymbol{D}$	Document set $\{d_1, \cdots, d_M\}$
$\boldsymbol{v}_{s_i}$	Embedding of word s_i
$a_{s_i s_j}, \boldsymbol{A}$	Bigram residuals
$\boldsymbol{t}_{ik}, \boldsymbol{T}_i$	Topic embeddings in doc d_i
$r_{ik}, \boldsymbol{r}_i$	Topic residuals in doc d_i
z_{ij}	Topic assignment of the j-th word j in doc d_i
$\boldsymbol{\phi}_i$	Mixing proportions of topics in doc d_i

Table 1: Table of notations

word belongs to a topic is determined by the Euclidean distance between the word embedding and the topic embedding. This assumption might be improper as the Euclidean distance is not an optimal measure of semantic relatedness between two embeddings[2].

3 Notations and Definitions

Throughout this paper, we use uppercase bold letters such as $\boldsymbol{S}, \boldsymbol{V}$ to denote a matrix or set, lowercase bold letters such as $\boldsymbol{v}_{w_i}$ to denote a vector, a normal uppercase letter such as N, W to denote a scalar constant, and a normal lowercase letter as s_i, w_i to denote a scalar variable.

Table 1 lists the notations in this paper.

In a document, a sequence of words is referred to as a *text window*, denoted by $w_i, \cdots, w_{i+l}$, or $w_i{:}w_{i+l}$. A text window of chosen size c before a word w_i defines the *context* of w_i as $w_{i-c}, \cdots, w_{i-1}$. Here w_i is referred to as the *focus word*. Each context word w_{i-j} and the focus word w_i comprise a bigram w_{i-j}, w_i.

We assume each word in a document is semantically similar to a *topic embedding*. Topic embeddings reside in the same N-dimensional space as word embeddings. When it is clear from context, topic embeddings are often referred to as *topic*s. Each document has K candidate topics, arranged in the matrix form $\boldsymbol{T}_i = (\boldsymbol{t}_{i1} \cdots \boldsymbol{t}_{iK})$, referred to as the *topic matrix*. Specifically, we fix $\boldsymbol{t}_{i1} = \boldsymbol{0}$, referring to it as the *null topic*.

In a document d_i, each word w_{ij} is assigned to a topic indexed by $z_{ij} \in \{1, \cdots, K\}$. Geometrically this means the embedding $\boldsymbol{v}_{w_{ij}}$ tends to align with the direction of $\boldsymbol{t}_{i,z_{ij}}$. Each topic $\boldsymbol{t}_{ik}$ has a document-specific prior probability to be assigned to a word, denoted as $\phi_{ik} = P(k|d_i)$. The vector $\boldsymbol{\phi}_i = (\phi_{i1}, \cdots, \phi_{iK})$ is referred to as the *mixing proportions* of these topics in document d_i.

[2]Almost all modern word embedding methods adopt the exponentiated cosine similarity as the link function, hence the cosine similarity may be assumed to be a better estimate of the semantic relatedness between embeddings derived from these methods.

4 Link Function of Topic Embedding

In this section, we formulate the distribution of a word given its context words and topic, in the form of a link function.

The core of most word embedding methods is a *link function* that connects the embeddings of a focus word and its context words, to define the distribution of the focus word. Li et al. (2015) proposed the following link function:

$$\begin{aligned} &P(w_c \mid w_0 : w_{c-1}) \\ \approx &P(w_c) \exp\Big\{ \boldsymbol{v}_{w_c}^\top \sum_{l=0}^{c-1} \boldsymbol{v}_{w_l} + \sum_{l=0}^{c-1} a_{w_l w_c} \Big\}. \quad (1) \end{aligned}$$

Here $a_{w_l w_c}$ is referred as the bigram residual, indicating the non-linear part not captured by $\boldsymbol{v}_{w_c}^\top \boldsymbol{v}_{w_l}$. It is essentially the logarithm of the normalizing constant of a softmax term. Some literature, e.g. (Pennington et al., 2014), refers to such a term as a bias term.

(1) is based on the assumption that the conditional distribution $P(w_c \mid w_0 : w_{c-1})$ can be factorized approximately into independent log-bilinear terms, each corresponding to a context word. This approximation leads to an efficient and effective word embedding algorithm *PSDVec* (Li et al., 2015). We follow this assumption, and propose to incorporate the topic of w_c in a way like a latent word. In particular, in addition to the context words, the corresponding embedding $\boldsymbol{t}_{ik}$ is included as a new log-bilinear term that influences the distribution of w_c. Hence we obtain the following extended link function:

$$\begin{aligned} &P(w_c \mid w_0{:}w_{c-1}, z_c, d_i) \approx P(w_c) \cdot \\ &\exp\Big\{ \boldsymbol{v}_{w_c}^\top \Big(\sum_{l=0}^{c-1} \boldsymbol{v}_{w_l} + \boldsymbol{t}_{z_c} \Big) + \sum_{l=0}^{c-1} a_{w_l w_c} + r_{z_c} \Big\}, \quad (2) \end{aligned}$$

where d_i is the current document, and r_{z_c} is the logarithm of the normalizing constant, named the *topic residual*. Note that the topic embeddings $\boldsymbol{t}_{z_c}$ may be specific to d_i. For simplicity of notation, we drop the document index in $\boldsymbol{t}_{z_c}$. To restrict the impact of topics and avoid overfitting, we constrain the magnitudes of all topic embeddings, so that they are always within a hyperball of radius γ.

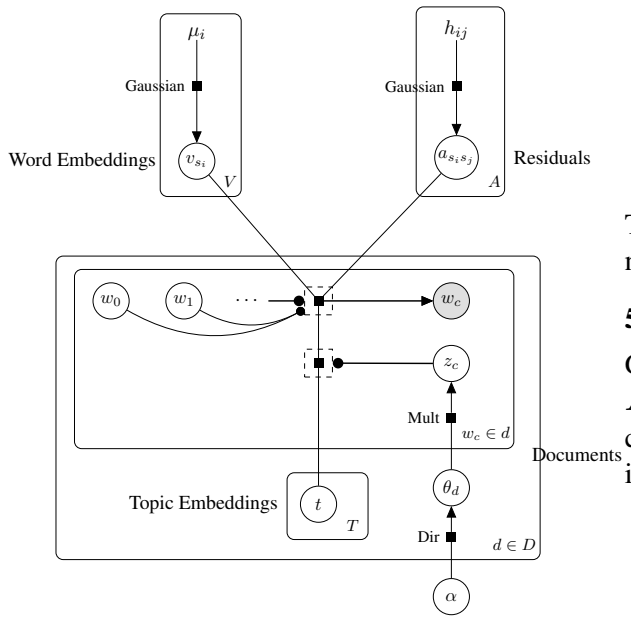


Figure 1: Graphical representation of TopicVec.

It is infeasible to compute the exact value of the topic residual r_k. We approximate it by the context size $c = 0$. Then (2) becomes:

$$P(w_c \mid k, d_i) = P(w_c) \exp\left\{\boldsymbol{v}_{w_c}^\top \boldsymbol{t}_k + r_k\right\}. \quad (3)$$

It is required that $\sum_{w_c \in \boldsymbol{S}} P(w_c \mid k) = 1$ to make (3) a distribution. It follows that

$$r_k = -\log\Big(\sum_{s_j \in \boldsymbol{S}} P(s_j) \exp\{\boldsymbol{v}_{s_j}^\top \boldsymbol{t}_k\}\Big). \quad (4)$$

(4) can be expressed in the matrix form:

$$\boldsymbol{r} = -\log(\boldsymbol{u} \exp\{\boldsymbol{V}^\top \boldsymbol{T}\}), \quad (5)$$

where $\boldsymbol{u}$ is the row vector of unigram probabilities.

5 Generative Process and Likelihood

The generative process of words in documents can be regarded as a hybrid of LDA and PSDVec. Analogous to PSDVec, the word embedding $\boldsymbol{v}_{s_i}$ and residual $a_{s_i s_j}$ are drawn from respective Gaussians. For the sake of clarity, we ignore their generation steps, and focus on the topic embeddings. The remaining generative process is as follows:

1. For the k-th topic, draw a topic embedding uniformly from a hyperball of radius γ, i.e. $\boldsymbol{t}_k \sim \text{Unif}(B_\gamma)$;
2. For each document d_i:
   (a) Draw the mixing proportions $\boldsymbol{\phi}_i$ from the Dirichlet prior $\text{Dir}(\boldsymbol{\alpha})$;
   (b) For the j-th word:
      i. Draw topic assignment z_{ij} from the categorical distribution $\text{Cat}(\boldsymbol{\phi}_i)$;
      ii. Draw word w_{ij} from $\boldsymbol{S}$ according to $P(w_{ij} \mid w_{i,j-c}{:}w_{i,j-1}, z_{ij}, d_i)$.

The above generative process is presented in plate notation in Figure (1).

5.1 Likelihood Function

Given the embeddings $\boldsymbol{V}$, the bigram residuals $\boldsymbol{A}$, the topics $\boldsymbol{T}_i$ and the hyperparameter $\boldsymbol{\alpha}$, the complete-data likelihood of a single document d_i is:

$$\begin{aligned}
&p(d_i, \boldsymbol{Z}_i, \boldsymbol{\phi}_i | \boldsymbol{\alpha}, \boldsymbol{V}, \boldsymbol{A}, \boldsymbol{T}_i) \\
=&p(\boldsymbol{\phi}_i|\boldsymbol{\alpha}) p(\boldsymbol{Z}_i|\boldsymbol{\phi}_i) p(d_i|\boldsymbol{V}, \boldsymbol{A}, \boldsymbol{T}_i, \boldsymbol{Z}_i) \\
=&\frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{j=1}^K \phi_{ij}^{\alpha_j - 1} \cdot \prod_{j=1}^{L_i} \Bigg(\phi_{i,z_{ij}} P(w_{ij}) \\
&\cdot \exp\Bigg\{ \boldsymbol{v}_{w_{ij}}^\top \Big(\sum_{l=j-c}^{j-1} \boldsymbol{v}_{w_{il}} + \boldsymbol{t}_{z_{ij}} \Big) \\
&+ \sum_{l=j-c}^{j-1} a_{w_{il} w_{ij}} + r_{i,z_{ij}} \Bigg\} \Bigg),
\end{aligned} \quad (6)$$

where $\boldsymbol{Z}_i = (z_{i1}, \cdots, z_{iL_i})$, and $\Gamma(\cdot)$ is the Gamma function.

Let $\boldsymbol{Z}, \boldsymbol{T}, \boldsymbol{\phi}$ denote the collection of all the document-specific $\{\boldsymbol{Z}_i\}_{i=1}^M, \{\boldsymbol{T}_i\}_{i=1}^M, \{\boldsymbol{\phi}_i\}_{i=1}^M$, respectively. Then the complete-data likelihood of the whole corpus is:

$$\begin{aligned}
&p(\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{V}, \boldsymbol{Z}, \boldsymbol{T}, \boldsymbol{\phi} | \boldsymbol{\alpha}, \gamma, \boldsymbol{\mu}) \\
=&\prod_{i=1}^W P(\boldsymbol{v}_{s_i}; \mu_i) \prod_{i,j=1}^{W,W} P(a_{s_i s_j}; f(h_{ij})) \prod_k^K \text{Unif}(B_\gamma) \\
&\cdot \prod_{i=1}^M \{p(\boldsymbol{\phi}_i|\boldsymbol{\alpha}) p(\boldsymbol{Z}_i|\boldsymbol{\phi}_i) p(d_i|\boldsymbol{V}, \boldsymbol{A}, \boldsymbol{T}_i, \boldsymbol{Z}_i)\} \\
=&\frac{1}{\mathcal{Z}(\boldsymbol{H}, \boldsymbol{\mu}) U_\gamma^K} \exp\{-\sum_{i,j=1}^{W,W} f(h_{i,j}) a_{s_i s_j}^2 - \sum_{i=1}^W \mu_i \|\boldsymbol{v}_{s_i}\|^2\} \\
&\cdot \prod_{i=1}^M \Bigg\{ \frac{\Gamma(\sum_{k=1}^K \alpha_k)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{j=1}^K \phi_{ij}^{\alpha_j - 1} \cdot \prod_{j=1}^{L_i} \Bigg(\phi_{i,z_{ij}} P(w_{ij}) \\
&\cdot \exp\Big\{ \boldsymbol{v}_{w_{ij}}^\top \Big(\sum_{l=j-c}^{j-1} \boldsymbol{v}_{w_{il}} + \boldsymbol{t}_{z_{ij}} \Big) + \sum_{l=j-c}^{j-1} a_{w_{il} w_{ij}} + r_{i,z_{ij}} \Big\} \Bigg) \Bigg\},
\end{aligned} \quad (7)$$

where $P(\boldsymbol{v}_{s_i}; \mu_i)$ and $P(a_{s_i s_j}; f(h_{ij}))$ are the two Gaussian priors as defined in (Li et al., 2015).

Following the convention in (Li et al., 2015), $h_{ij}, \boldsymbol{H}$ are empirical bigram probabilities, $\boldsymbol{\mu}$ are the embedding magnitude penalty coefficients, and $\mathcal{Z}(\boldsymbol{H}, \boldsymbol{\mu})$ is the normalizing constant for word embeddings. U_γ is the volume of the hyperball of radius γ.

Taking the logarithm of both sides, we obtain

$$\begin{aligned}
&\log p(\boldsymbol{D}, \boldsymbol{A}, \boldsymbol{V}, \boldsymbol{Z}, \boldsymbol{T}, \boldsymbol{\phi} | \boldsymbol{\alpha}, \gamma, \boldsymbol{\mu}) \\
=&C_0 - \log \mathcal{Z}(\boldsymbol{H}, \boldsymbol{\mu}) - \|\boldsymbol{A}\|^2_{f(\boldsymbol{H})} - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 \\
+&\sum_{i=1}^{M} \Bigg\{ \sum_{k=1}^{K} \log \phi_{ik} (m_{ik} + \alpha_k - 1) + \sum_{j=1}^{L_i} \Bigg(r_{i,z_{ij}} \\
+&\boldsymbol{v}_{w_{ij}}^\top \Big(\sum_{l=j-c}^{j-1} \boldsymbol{v}_{w_{il}} + \boldsymbol{t}_{z_{ij}} \Big) + \sum_{l=j-c}^{j-1} a_{w_{il} w_{ij}} \Bigg) \Bigg\}, \quad (8)
\end{aligned}$$

where $m_{ik} = \sum_{j=1}^{L_i} \delta(z_{ij} = k)$ counts the number of words assigned with the k-th topic in d_i, $C_0 = M \log \frac{\Gamma(\sum_{k=1}^{K} \alpha_k)}{\prod_{k=1}^{K} \Gamma(\alpha_k)} + \sum_{i,j=1}^{M,L_i} \log P(w_{ij}) - K \log U_\gamma$ is constant given the hyperparameters.

6 Variational Inference Algorithm

6.1 Learning Objective and Process

Given the hyperparameters $\boldsymbol{\alpha}, \gamma, \boldsymbol{\mu}$, the learning objective is to find the embeddings $\boldsymbol{V}$, the topics $\boldsymbol{T}$, and the word-topic and document-topic distributions $p(\boldsymbol{Z}_i, \boldsymbol{\phi}_i | d_i, \boldsymbol{A}, \boldsymbol{V}, \boldsymbol{T})$. Here the hyperparameters $\boldsymbol{\alpha}, \gamma, \boldsymbol{\mu}$ are kept constant, and we make them implicit in the distribution notations.

However, the coupling between $\boldsymbol{A}, \boldsymbol{V}$ and $\boldsymbol{T}, \boldsymbol{Z}, \boldsymbol{\phi}$ makes it inefficient to optimize them simultaneously. To get around this difficulty, we learn word embeddings and topic embeddings separately. Specifically, the learning process is divided into two stages:

1. In the first stage, considering that the topics have a relatively small impact to word distributions and the impact might be "averaged out" across different documents, we simplify the model by ignoring topics temporarily. Then the model falls back to the original PSDVec. The optimal solution $\boldsymbol{V}^*, \boldsymbol{A}^*$ is obtained accordingly;
2. In the second stage, we treat $\boldsymbol{V}^*, \boldsymbol{A}^*$ as constant, plug it into the likelihood function, and find the corresponding optimal $\boldsymbol{T}^*, p(\boldsymbol{Z}, \boldsymbol{\phi} | \boldsymbol{D}, \boldsymbol{A}^*, \boldsymbol{V}^*, \boldsymbol{T}^*)$ of the full model. As in LDA, this posterior is analytically intractable, and we use a simpler variational distribution $q(\boldsymbol{Z}, \boldsymbol{\phi})$ to approximate it.

6.2 Mean-Field Approximation and Variational GEM Algorithm

In this stage, we fix $\boldsymbol{V} = \boldsymbol{V}^*, \boldsymbol{A} = \boldsymbol{A}^*$, and seek the optimal $\boldsymbol{T}^*, p(\boldsymbol{Z}, \boldsymbol{\phi} | \boldsymbol{D}, \boldsymbol{A}^*, \boldsymbol{V}^*, \boldsymbol{T}^*)$. As $\boldsymbol{V}^*, \boldsymbol{A}^*$ are constant, we also make them implicit in the following expressions.

For an arbitrary variational distribution $q(\boldsymbol{Z}, \boldsymbol{\phi})$, the following equalities hold

$$\begin{aligned}
&E_q \log \left[\frac{p(\boldsymbol{D}, \boldsymbol{Z}, \boldsymbol{\phi} | \boldsymbol{T})}{q(\boldsymbol{Z}, \boldsymbol{\phi})} \right] \\
=&E_q \left[\log p(\boldsymbol{D}, \boldsymbol{Z}, \boldsymbol{\phi} | \boldsymbol{T}) \right] + \mathcal{H}(q) \\
=&\log p(\boldsymbol{D} | \boldsymbol{T}) - \mathrm{KL}(q || p), \quad (9)
\end{aligned}$$

where $p = p(\boldsymbol{Z}, \boldsymbol{\phi} | \boldsymbol{D}, \boldsymbol{T})$, $\mathcal{H}(q)$ is the entropy of q. This implies

$$\begin{aligned}
&\mathrm{KL}(q || p) \\
=&\log p(\boldsymbol{D} | \boldsymbol{T}) - \Big(E_q \left[\log p(\boldsymbol{D}, \boldsymbol{Z}, \boldsymbol{\phi} | \boldsymbol{T}) \right] + \mathcal{H}(q) \Big) \\
=&\log p(\boldsymbol{D} | \boldsymbol{T}) - \mathcal{L}(q, \boldsymbol{T}). \quad (10)
\end{aligned}$$

In (10), $E_q \left[\log p(\boldsymbol{D}, \boldsymbol{Z}, \boldsymbol{\phi} | \boldsymbol{T}) \right] + \mathcal{H}(q)$ is usually referred to as the *variational free energy* $\mathcal{L}(q, \boldsymbol{T})$, which is a lower bound of $\log p(\boldsymbol{D} | \boldsymbol{T})$. Directly maximizing $\log p(\boldsymbol{D} | \boldsymbol{T})$ w.r.t. $\boldsymbol{T}$ is intractable due to the hidden variables $\boldsymbol{Z}, \boldsymbol{\phi}$, so we maximize its lower bound $\mathcal{L}(q, \boldsymbol{T})$ instead. We adopt a mean-field approximation of the true posterior as the variational distribution, and use a variational algorithm to find $q^*, \boldsymbol{T}^*$ maximizing $\mathcal{L}(q, \boldsymbol{T})$.

The following variational distribution is used:

$$\begin{aligned}
&q(\boldsymbol{Z}, \boldsymbol{\phi}; \boldsymbol{\pi}, \boldsymbol{\theta}) = q(\boldsymbol{\phi}; \boldsymbol{\theta}) q(\boldsymbol{Z}; \boldsymbol{\pi}) \\
=&\prod_{i=1}^{M} \left\{ \mathrm{Dir}(\boldsymbol{\phi}_i; \boldsymbol{\theta}_i) \prod_{j=1}^{L_i} \mathrm{Cat}(z_{ij}; \boldsymbol{\pi}_{ij}) \right\}. \quad (11)
\end{aligned}$$

We can obtain (Li et al., 2016a)

$$\begin{aligned}
&\mathcal{L}(q, \boldsymbol{T}) \\
=&\sum_{i=1}^{M} \Bigg\{ \sum_{k=1}^{K} \Big(\sum_{j=1}^{L_i} \pi_{ij}^k + \alpha_k - 1 \Big) \Big(\psi(\theta_{ik}) - \psi(\theta_{i0}) \Big) \\
&+ \mathrm{Tr}(\boldsymbol{T}_i^\top \sum_{j=1}^{L_i} \boldsymbol{v}_{w_{ij}} \boldsymbol{\pi}_{ij}^\top) + \boldsymbol{r}_i^\top \sum_{j=1}^{L_i} \boldsymbol{\pi}_{ij} \Bigg\} \\
&+ \mathcal{H}(q) + C_1, \quad (12)
\end{aligned}$$

where $\boldsymbol{T}_i$ is the topic matrix of the i-th document, and $\boldsymbol{r}_i$ is the vector constructed by concatenating all the topic residuals r_{ik}. $C_1 = C_0 - \log \mathcal{Z}(\boldsymbol{H}, \boldsymbol{\mu}) - \|\boldsymbol{A}\|^2_{f(\boldsymbol{H})} - \sum_{i=1}^{W} \mu_i \|\boldsymbol{v}_{s_i}\|^2 + \sum_{i,j=1}^{M,L_i} \Big(\boldsymbol{v}_{w_{ij}}^\top \sum_{k=j-c}^{j-1} \boldsymbol{v}_{w_{ik}} + \sum_{k=j-c}^{j-1} a_{w_{ik} w_{ij}} \Big)$ is constant.

We proceed to optimize (12) with a Generalized Expectation-Maximization (GEM) algorithm w.r.t. q and $\boldsymbol{T}$ as follows:

1. Initialize all the topics $\boldsymbol{T}_i = \boldsymbol{0}$, and correspondingly their residuals $\boldsymbol{r}_i = \boldsymbol{0}$;
2. Iterate over the following two steps until convergence. In the l-th step:
   (a) Let the topics and residuals be $\boldsymbol{T} = \boldsymbol{T}^{(l-1)}, \boldsymbol{r} = \boldsymbol{r}^{(l-1)}$, find $q^{(l)}(\boldsymbol{Z}, \boldsymbol{\phi})$ that maximizes $\mathcal{L}(q, \boldsymbol{T}^{(l-1)})$. This is the Expectation step (E-step). In this step, $\log p(\boldsymbol{D}|\boldsymbol{T})$ is constant. Then the q that maximizes $\mathcal{L}(q, \boldsymbol{T}^{(l)})$ will minimize $\mathrm{KL}(q||p)$, i.e. such a q is the closest variational distribution to p measured by KL-divergence;
   (b) Given the variational distribution $q^{(l)}(\boldsymbol{Z}, \boldsymbol{\phi})$, find $\boldsymbol{T}^{(l)}, \boldsymbol{r}^{(l)}$ that improve $\mathcal{L}(q^{(l)}, \boldsymbol{T})$, using Gradient descent method. This is the generalized Maximization step (M-step). In this step, $\boldsymbol{\pi}, \boldsymbol{\theta}, \mathcal{H}(q)$ are constant.

6.2.1 Update Equations of $\boldsymbol{\pi}, \boldsymbol{\theta}$ in E-Step

In the E-step, $\boldsymbol{T} = \boldsymbol{T}^{(l-1)}, \boldsymbol{r} = \boldsymbol{r}^{(l-1)}$ are constant. Taking the derivative of $\mathcal{L}(q, \boldsymbol{T}^{(l-1)})$ w.r.t. π_{ij}^k and θ_{ik}, respectively, we can obtain the optimal solutions (Li et al., 2016a) at:

$$\pi_{ij}^k \propto \exp\{\psi(\theta_{ik}) + \boldsymbol{v}_{w_{ij}}^\top \boldsymbol{t}_{ik} + r_{ik}\}. \tag{13}$$

$$\theta_{ik} = \sum_{j=1}^{L_i} \pi_{ij}^k + \alpha_k. \tag{14}$$

6.2.2 Update Equation of $\boldsymbol{T}_i$ in M-Step

In the Generalized M-step, $\boldsymbol{\pi} = \boldsymbol{\pi}^{(l)}, \boldsymbol{\theta} = \boldsymbol{\theta}^{(l)}$ are constant. For notational simplicity, we drop their superscripts (l).

To update $\boldsymbol{T}_i$, we first take the derivative of (12) w.r.t. $\boldsymbol{T}_i$, and then take the Gradient Descent method.

The derivative is obtained as (Li et al., 2016a):

$$\frac{\partial \mathcal{L}(q^{(l)}, \boldsymbol{T})}{\partial \boldsymbol{T}_i} = \sum_{j=1}^{L_i} \boldsymbol{v}_{w_{ij}} \boldsymbol{\pi}_{ij}^\top + \sum_{k=1}^{K} \bar{m}_{ik} \frac{\partial r_{ik}}{\partial \boldsymbol{T}_i}, \tag{15}$$

where $\bar{m}_{ik} = \sum_{j=1}^{L_i} \pi_{ij}^k = E[m_{ik}]$, the sum of the variational probabilities of each word being assigned to the k-th topic in the i-th document. $\frac{\partial r_{ik}}{\partial \boldsymbol{T}_i}$ is a gradient matrix, whose j-th column is $\frac{\partial r_{ik}}{\partial \boldsymbol{t}_{ij}}$.

Remind that $r_{ik} = -\log\Big(E_{P(s)}[\exp\{\boldsymbol{v}_s^\top \boldsymbol{t}_{ik}\}]\Big)$. When $j \neq k$, it is easy to verify that $\frac{\partial r_{ik}}{\partial \boldsymbol{t}_{ij}} = \boldsymbol{0}$. When $j = k$, we have

$$\begin{aligned} \frac{\partial r_{ik}}{\partial \boldsymbol{t}_{ik}} &= e^{-r_{ik}} \cdot E_{P(s)}[\exp\{\boldsymbol{v}_s^\top \boldsymbol{t}_{ik}\} \boldsymbol{v}_s] \\ &= e^{-r_{ik}} \cdot \sum_{s \in W} \exp\{\boldsymbol{v}_s^\top \boldsymbol{t}_{ik}\} P(s) \boldsymbol{v}_s \\ &= e^{-r_{ik}} \cdot \exp\{\boldsymbol{t}_{ik}^\top \boldsymbol{V}\} (\boldsymbol{u} \circ \boldsymbol{V}), \end{aligned} \tag{16}$$

where $\boldsymbol{u} \circ \boldsymbol{V}$ is to multiply each column of $\boldsymbol{V}$ with $\boldsymbol{u}$ element-by-element.

Therefore $\frac{\partial r_{ik}}{\partial \boldsymbol{T}_i} = (\boldsymbol{0}, \cdots \frac{\partial r_{ik}}{\partial \boldsymbol{t}_{ik}}, \cdots, \boldsymbol{0})$. Plugging it into (15), we obtain

$$\frac{\partial \mathcal{L}(q^{(l)}, \boldsymbol{T})}{\partial \boldsymbol{T}_i} = \sum_{j=1}^{L_i} \boldsymbol{v}_{w_{ij}} \boldsymbol{\pi}_{ij}^\top + (\bar{m}_{i1} \frac{\partial r_{i1}}{\partial \boldsymbol{t}_{i1}}, \cdots, \bar{m}_{iK} \frac{\partial r_{iK}}{\partial \boldsymbol{t}_{iK}}).$$

We proceed to optimize $\boldsymbol{T}_i$ with a gradient descent method:

$$\boldsymbol{T}_i^{(l)} = \boldsymbol{T}^{(l-1)} + \lambda(l, L_i) \frac{\partial \mathcal{L}(q^{(l)}, \boldsymbol{T})}{\partial \boldsymbol{T}_i},$$

where $\lambda(l, L_i) = \frac{L_0 \lambda_0}{l \cdot \max\{L_i, L_0\}}$ is the learning rate function, L_0 is a pre-specified document length threshold, and λ_0 is the initial learning rate. As the magnitude of $\frac{\partial \mathcal{L}(q^{(l)}, \boldsymbol{T})}{\partial \boldsymbol{T}_i}$ is approximately proportional to the document length L_i, to avoid the step size becoming too big a on a long document, if $L_i > L_0$, we normalize it by L_i.

To satisfy the constraint that $\|\boldsymbol{t}_{ik}^{(l)}\| \leq \gamma$, when $\boldsymbol{t}_{ik}^{(l)} > \gamma$, we normalize it by $\gamma / \|\boldsymbol{t}_{ik}^{(l)}\|$.

After we obtain the new $\boldsymbol{T}$, we update $\boldsymbol{r}_i^{(m)}$ using (5).

Sometimes, especially in the initial few iterations, due to the excessively big step size of the gradient descent, $\mathcal{L}(q, \boldsymbol{T})$ may decrease after the update of $\boldsymbol{T}$. Nonetheless the general direction of $\mathcal{L}(q, \boldsymbol{T})$ is increasing.

6.3 Sharing of Topics across Documents

In principle we could use one set of topics across the whole corpus, or choose different topics for different subsets of documents. One could choose a way to best utilize cross-document information.

For instance, when the document category information is available, we could make the documents in each category share their respective set

of topics, so that M categories correspond to M sets of topics. In the learning algorithm, only the update of π_{ij}^k needs to be changed to cater for this situation: when the k-th topic is relevant to the document i, we update π_{ij}^k using (13); otherwise $\pi_{ij}^k = 0$.

An identifiability problem may arise when we split topic embeddings according to document subsets. In different topic groups, some highly similar redundant topics may be learned. If we project documents into the topic space, portions of documents in the same topic in different documents may be projected onto different dimensions of the topic space, and similar documents may eventually be projected into very different topic proportion vectors. In this situation, directly using the projected topic proportion vectors could cause problems in unsupervised tasks such as clustering. A simple solution to this problem would be to compute the pairwise similarities between topic embeddings, and consider these similarities when computing the similarity between two projected topic proportion vectors. Two similar documents will then still receive a high similarity score.

7 Experimental Results

To investigate the quality of document representation of our TopicVec model, we compared its performance against eight topic modeling or document representation methods in two document classification tasks. Moreover, to show the topic coherence of TopicVec on a single document, we present the top words in top topics learned on a news article.

7.1 Document Classification Evaluation

7.1.1 Experimental Setup

Compared Methods Two setups of TopicVec were evaluated:

- **TopicVec**: the topic proportions learned by TopicVec;
- **TV+WV**: the topic proportions, concatenated with the mean word embedding of the document (same as the MeanWV below).

We compare the performance of our methods against eight methods, including three topic modeling methods, three continuous document representation methods, and the conventional bag-of-words (**BOW**) method. The count vector of BOW is unweighted.

The topic modeling methods include:

- **LDA**: the vanilla LDA (Blei et al., 2003) in the gensim library[3];
- **sLDA**: Supervised Topic Model[4] (McAuliffe and Blei, 2008), which improves the predictive performance of LDA by modeling class labels;
- **LFTM**: Latent Feature Topic Modeling[5] (Nguyen et al., 2015).

The document-topic proportions of topic modeling methods were used as their document representation.

The document representation methods are:

- **Doc2Vec**: Paragraph Vector (Le and Mikolov, 2014) in the gensim library[6].
- **TWE**: Topical Word Embedding[7] (Liu et al., 2015), which represents a document by concatenating average topic embedding and average word embedding, similar to our TV+WV;
- **GaussianLDA**: Gaussian LDA[8] (Das et al., 2015), which assumes that words in a topic are random samples from a multivariate Gaussian distribution with the mean as the topic embedding. Similar to TopicVec, we derived the posterior topic proportions as the features of each document;
- **MeanWV**: The mean word embedding of the document.

Datasets We used two standard document classification corpora: the 20 Newsgroups[9] and the ApteMod version of the Reuters-21578 corpus[10]. The two corpora are referred to as the **20News** and **Reuters** in the following.

20News contains about 20,000 newsgroup documents evenly partitioned into 20 different categories. Reuters contains 10,788 documents, where each document is assigned to one or more categories. For the evaluation of document classification, documents appearing in two or more categories were removed. The numbers of documents in the categories of Reuters are highly imbalanced, and we only selected the largest 10 categories, leaving us with 8,025 documents in total.

[3] https://radimrehurek.com/gensim/models/ldamodel.html
[4] http://www.cs.cmu.edu/~chongw/slda/
[5] https://github.com/datquocnguyen/LFTM/
[6] https://radimrehurek.com/gensim/models/doc2vec.html
[7] https://github.com/largelymfs/topical_word_embeddings/
[8] https://github.com/rajarshd/Gaussian_LDA
[9] http://qwone.com/~jason/20Newsgroups/
[10] http://www.nltk.org/book/ch02.html

The same preprocessing steps were applied to all methods: words were lowercased; stop words and words out of the word embedding vocabulary (which means that they are extremely rare) were removed.

Experimental Settings TopicVec used the word embeddings trained using PSDVec on a March 2015 Wikipedia snapshot. It contains the most frequent 180,000 words. The dimensionality of word embeddings and topic embeddings was 500. The hyperparameters were $\boldsymbol{\alpha} = (0.1, \cdots, 0.1), \gamma = 5$. For 20news and Reuters, we specified 15 and 12 topics in each category on the training set, respectively. The first topic in each category was always set to null. The learned topic embeddings were combined to form the whole topic set, where redundant null topics in different categories were removed, leaving us with 281 topics for 20News and 111 topics for Reuters. The initial learning rate was set to 0.1. After 100 GEM iterations on each dataset, the topic embeddings were obtained. Then the posterior document-topic distributions of the test sets were derived by performing one E-step given the topic embeddings trained on the training set.

LFTM includes two models: LF-LDA and LF-DMM. We chose the better performing LF-LDA to evaluate. TWE includes three models, and we chose the best performing TWE-1 to compare.

LDA, sLDA, LFTM and TWE used the specified 50 topics on Reuters, as this is the optimal topic number according to (Lu et al., 2011). On the larger 20news dataset, they used the specified 100 topics. Other hyperparameters of all compared methods were left at their default values.

GaussianLDA was specified 100 topics on 20news and 70 topics on Reuters. As each sampling iteration took over 2 hours, we only had time for 100 sampling iterations.

For each method, after obtaining the document representations of the training and test sets, we trained an ℓ-1 regularized linear SVM one-vs-all classifier on the training set using the scikit-learn library[11]. We then evaluated its predictive performance on the test set.

Evaluation metrics Considering that the largest few categories dominate Reuters, we adopted macro-averaged precision, recall and F1 measures as the evaluation metrics, to avoid the average results being dominated by the performance of the top categories.

[11]http://scikit-learn.org/stable/modules/svm.html

	20News			Reuters		
	Prec	Rec	F1	Prec	Rec	F1
BOW	69.1	68.5	68.6	92.5	90.3	91.1
LDA	61.9	61.4	60.3	76.1	74.3	74.8
sLDA	61.4	60.9	60.9	88.3	83.3	85.1
LFTM	63.5	64.8	63.7	84.6	86.3	84.9
MeanWV	70.4	70.3	70.1	**92.0**	89.6	90.5
Doc2Vec	56.3	56.6	55.4	84.4	50.0	58.5
TWE	69.5	69.3	68.8	91.0	89.1	89.9
GaussianLDA	30.9	26.5	22.7	46.2	31.5	35.3
TopicVec	71.4	71.3	71.2	91.8	**92.0**	**91.7**
TV+WV[1]	**72.1**	**71.9**	**71.8**	91.4	91.9	91.5

[1]Combined features of TopicVec topic proportions and MeanWV.

Table 2: Performance on multi-class text classification. Best score is in boldface.

Avg. Features	BOW	MeanWV	TWE	TopicVec	TV+WV
20News	50381	500	800	281	781
Reuters	17989	500	800	111	611

Table 3: Number of features of the five best performing methods.

Evaluation Results Table 2 presents the performance of the different methods on the two classification tasks. The highest scores were highlighted with boldface. It can be seen that TV+WV and TopicVec obtained the best performance on the two tasks, respectively. With only topic proportions as features, TopicVec performed slightly better than BOW, MeanWV and TWE, and significantly outperformed four other methods. The number of features it used was much lower than BOW, MeanWV and TWE (Table 3).

GaussianLDA performed considerably inferior to all other methods. After checking the generated topic embeddings manually, we found that the embeddings for different topics are highly similar to each other. Hence the posterior topic proportions were almost uniform and non-discriminative. In addition, on the two datasets, even the fastest Alias sampling in (Das et al., 2015) took over 2 hours for one iteration and 10 days for the whole 100 iterations. In contrast, our method finished the 100 EM iterations in 2 hours.

Figure 2: Topic Cloud of the pharmaceutical company acquisition news.

7.2 Qualitative Assessment of Topics Derived from a Single Document

Topic models need a large set of documents to extract coherent topics. Hence, methods depending on topic models, such as TWE, are subject to this limitation. In contrast, TopicVec can extract coherent topics and obtain document representations even when only one document is provided as input.

To illustrate this feature, we ran TopicVec on a New York Times news article about a pharmaceutical company acquisition[12], and obtained 20 topics.

Figure 2 presents the most relevant words in the top-6 topics as a *topic cloud*. We first calculated the relevance between a word and a topic as the frequency-weighted cosine similarity of their embeddings. Then the most relevant words were selected to represent each topic. The sizes of the topic slices are proportional to the topic proportions, and the font sizes of individual words are proportional to their relevance to the topics. Among these top-6 topics, the largest and smallest topic proportions are 26.7% and 9.9%, respectively.

As shown in Figure 2, words in obtained topics were generally coherent, although the topics were only derived from a single document. The reason is that TopicVec takes advantage of the rich semantic information encoded in word embeddings, which were pretrained on a large corpus.

The topic coherence suggests that the derived topic embeddings were approximately the semantic centroids of the document. This capacity may aid applications such as document retrieval, where a "compressed representation" of the query document is helpful.

[12]http://www.nytimes.com/2015/09/21/business/a-huge-overnight-increase-in-a-drugs-price-raises-protests.html

8 Conclusions and Future Work

In this paper, we proposed TopicVec, a generative model combining word embedding and LDA, with the aim of exploiting the word collocation patterns both at the level of the local context and the global document. Experiments show that TopicVec can learn high-quality document representations, even given only one document.

In our classification tasks we only explored the use of topic proportions of a document as its representation. However, jointly representing a document by topic proportions and topic embeddings would be more accurate. Efficient algorithms for this task have been proposed (Kusner et al., 2015).

Our method has potential applications in various scenarios, such as document retrieval, classification, clustering and summarization.

Acknowlegement

We thank Xiuchao Sui and Linmei Hu for their help and support. We thank the anonymous mentor provided by ACL for the careful proofreading. This research is funded by the National Research Foundation, Prime Minister's Office, Singapore under its IDM Futures Funding Initiative and IRC@SG Funding Initiative administered by IDMPO.

References

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

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Rajarshi Das, Manzil Zaheer, and Chris Dyer. 2015. Gaussian LDA for topic models with word embeddings. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*

and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 795–804, Beijing, China, July. Association for Computational Linguistics.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pages 1607–1614.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q. Weinberger. 2015. From word embeddings to document distances. In David Blei and Francis Bach, editors, *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 957–966. JMLR Workshop and Conference Proceedings.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In *Advances in Neural Information Processing Systems*, pages 2708–2716.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Shaohua Li, Jun Zhu, and Chunyan Miao. 2015. A generative word embedding model and its low rank positive semidefinite solution. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1599–1609, Lisbon, Portugal, September. Association for Computational Linguistics.

Shaohua Li, Tat-Seng Chua, Jun Zhu, and Chunyan Miao. 2016a. Generative topic embedding: a continuous representation of documents (extended version with proofs). Technical report. https://github.com/askerlee/topicvec/blob/master/topicvec-ext.pdf.

Shaohua Li, Jun Zhu, and Chunyan Miao. 2016b. PSDVec: a toolbox for incremental and scalable word embedding. *To appear in Neurocomputing*.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*, pages 2418–2424.

Yue Lu, Qiaozhu Mei, and ChengXiang Zhai. 2011. Investigating task performance of probabilistic topic models: an empirical study of PLSA and LDA. *Information Retrieval*, 14(2):178–203.

Jon D McAuliffe and David M Blei. 2008. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS 2013*, pages 3111–3119.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *Transactions of the Association for Computational Linguistics*, 3:299–313.

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

Detecting Common Discussion Topics Across Culture From News Reader Comments

Bei Shi[1], Wai Lam[1], Lidong Bing[2] and Yinqing Xu[1]
[1]Department of Systems Engineering and Engineering Management
The Chinese University of Hong Kong, Hong Kong
[2]Machine Learning Department
Carnegie Mellon University, Pittsburgh, PA 15213
{bshi,wlam,yqxu}@se.cuhk.edu.hk
lbing@cs.cmu.edu

Abstract

News reader comments found in many on-line news websites are typically massive in amount. We investigate the task of **C**ultural-common **T**opic **D**etection (CTD), which is aimed at discovering common discussion topics from news reader comments written in different languages. We propose a new probabilistic graphical model called MCTA which can cope with the language gap and capture the common semantics in different languages. We also develop a partially collapsed Gibbs sampler which effectively incorporates the term translation relationship into the detection of cultural-common topics for model parameter learning. Experimental results show improvements over the state-of-the-art model.

1 Introduction

Nowadays the rapid development of information and communication technology enables more and more people around the world to engage in the movement of globalization. One effect of globalization is to facilitate greater connections between people bringing cultures closer than before. This also contributes to the convergence of some elements of different cultures (Melluish, 2014). For example, there is a growing tendency of people watching the same movie, listening to the same music, and reading the news about the same event. This kind of cultural homogenization brings the emergence of commonality of some aspects of different cultures worldwide. It would be beneficial to identify such common aspects among cultures. For example, it can provide some insights for global market and international business (Cavusgil et al., 2014).

Many news websites from different regions in the world report significant events which are of interests to people from different continents. These websites also allow readers around the world to give their comments in their own languages. The volume of comments is often enormous especially for popular events. In a news website, readers from a particular culture background tend to write comments in their own preferred languages. For some important or global events, we observe that readers from different cultures, via different languages, express common discussion topics. For instance, on March 8 2014, Malaysia Airlines Flight MH370, carrying 227 passengers and 12 crew members, disappeared. Upon the happening of this event, many news articles around the world reported it and many readers from different continents commented on this event. Through analyzing the reader comments manually, we observe that both English-speaking and Chinese-speaking readers expressed in their corresponding languages their desire for praying for the MH370 flight. This is an example of a **cultural-common** discussion topic. Identifying such cultural-common topics automatically can facilitate better understanding and organization of the common concerns or interests of readers with different language background. Such technology can be deployed for developing various applications. One application is to build a reader comment digest system that can organize comments by cultural-common discussion topics and rank the topics by popularity. This provides a functionality of analyzing the common focus of readers from different cultures on a particular event. An example of such application is shown in Figure 3. Under each event, reader comments are grouped by cultural-common topics.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 676–685,
Berlin, Germany, August 7-12, 2016.

In this paper, we investigate the task of **C**ultural-common **T**opic **D**etection (CTD) on multilingual news reader comments. Reader comments about a global event, written in different languages, from different news websites around the world exist in massive amount. The main goal of this task is to discover cultural-common discussion topics from raw multilingual news reader comments for a news event. One challenge is that the discussion topics are unknown. Another challenge is related to the language gap issue. Precisely, the words of reader comments in different languages are composed of different terms in their corresponding languages. Such language gap issue poses a great deal of challenge for identifying cultural-common discussion topics in multilingual news comments settings.

One recent work done by Prasojo et al. (2015) is to organize news reader comments around entities and aspects discussed by readers. Such organization of reader comments cannot handle the identification of common discussion topics. On the other hand, the Muto model proposed by Boyd-Graber and Blei (2009) can extract common topics from multilingual documents. This model merely outputs cross-lingual topics of matching word pairs. One example of such kind of topic contains key terms of word pairs such as "plane:飞机 ocean:海洋 ...". The assumption of one-to-one mapping of words has some drawbacks. One drawback is that the correspondence of identified common topics is restricted to the vocabulary level. Another drawback is that the one-to-one mapping of words cannot fit the original word occurrences well. For example, the English term "plane" appears in the English documents frequently while the Chinese translation "飞机" appears less. It is not reasonable that "plane" and "飞机" share the same probability mass in common topics. Another closely related existing work is the PCLSA model proposed by Zhang et al. (2010). PCLSA employs a mixture of English words and Chinese words to represent common topics. It incorporates bilingual constraints into the Probabilistic Latent Semantic Analysis (PLSA) model (Hofmann, 2001) and assumes that word pairs in the dictionary share similar probability in a common topic. However, similar to one-to-one mapping of words, such bilingual constraints cannot handle well the original word co-occurrence in each language resulting in a degradation of the coherence and interpretability of common topics.

We propose a new probabilistic graphical model which is able to detect cultural-common topics from multilingual news reader comments in an unsupervised manner. In principle, no labeled data is needed. In this paper, we focus on dealing with two languages, namely, English and Chinese news reader comments. Different from prior works, we design a technique based on auxiliary distributions which incorporates word distributions from the other language and can capture the common semantics on the topic level. We develop a partially collapsed Gibbs sampler which decouples the inference of topic distribution and word distribution. We also incorporate the term translation relationship, derived from a bilingual dictionary, into the detection of cultural-common topics for model parameter learning.

We have prepared a data set by collecting English and Chinese reader comments from different regions reflecting different culture. Our experimental results are encouraging showing improvements over the state-of-the-art model.

2 Related Work

Prasojo et al. (2015) and Biyani et al. (2015) organized news reader comments via identified entities or aspects. Such kind of organization via entities or aspects cannot capture common topics discussed by readers. Digesting merely based on entities fails to work in multilingual settings due to the fact that the common entities have distinct mentions in different languages.

Zhai et al. (2004) discovered common topics from comparable texts via a PLSA based mixture model. Paul and Girju (2009) proposed a Mixed-Collection Topic Model for finding common topics from different collections. Despite the fact that the above models can find a kind of common topic, they only deal with a single language setting without considering the language gap.

Some works discover common latent topics from multilingual corpora. For aligned corpora, they assume that the topic distribution in each document is the same (Vulić et al., 2011; Vulić and Moens, 2014; Erosheva et al., 2004; Fukumasu et al., 2012; Mimno et al., 2009; Ni et al., 2009; Zhang et al., 2013; Peng et al., 2014). However, aligned corpora are often unavailable for most domains. For unaligned corpora, cross-lingual topic models use some language resources, such

as a bilingual dictionary or a bilingual knowledge base to bridge the language gap (Boyd-Graber and Blei, 2009; Zhang et al., 2010; Jagarlamudi and Daumé III, 2010). As mentioned above, the goals of Boyd-Graber and Blei (2009) as well as Jagarlamud and Daumé (2010) focus on mining the correspondence of topics at the vocabulary level, which are different from that of Zhang et al. (2010) and ours. The model in Zhang et al. (2010) adds the constraints of word translation pairs into PLSA. These constraints cannot handle the original word co-occurrences well. In contrast, we consider the language gap by incorporating word distributions from the other language, capturing the common semantics on the topic level. Moreover, we use a fully Bayesian paradigm with a prior distribution.

Some existing topic methods conduct cross-lingual sentiment analysis (Lu et al., 2011; Guo et al., 2010; Lin et al., 2014; Boyd-Graber and Resnik, 2010). These models are not suitable for our CTD task because they mainly detect common elements related to product aspects. Moreover some works focus more on detecting sentiments.

3 Our Proposed Model

3.1 Model Description

The problem definition of the CTD task is described as follows. For a particular event, both English and Chinese news reader comments are collected from different regions reflecting different culture. The set of English comments is denoted by $\mathcal{E}$ and the set of Chinese comments is denoted by $\mathcal{C}$. The goal of the CTD task is to extract cultural-common topics $k \in \{1, 2, \ldots, K\}$ from $\mathcal{E}$ and $\mathcal{C}$. The set of multilingual news reader comments of each event are processed within the same event.

Our proposed model is called **M**ultilingual **C**ultural-common **T**opic **A**nalysis (MCTA) which is based on graphical model paradigm as depicted in Figure 1. The plate on the right represents cultural-common topics. Each cultural-common topic k is represented by an English word distribution φ_k^e over English vocabulary Λ^e and a Chinese word distribution φ_k^c over Chinese vocabulary Λ^c. We make use of a bilingual dictionary, which is composed of many-to-many word translations among English and Chinese words. To capture common semantics of multilingual news reader comments, we design two auxiliary distributions η_e, with dimension Λ^e, and η_c, with dimension Λ^c, to help the generation of φ_k^e and φ_k^c. Precisely, we generate η_e and η_c from the Dirichlet prior distributions $Dir(\beta \cdot \mathbf{1}_{|\Lambda^e|})$ and $Dir(\beta \cdot \mathbf{1}_{|\Lambda^c|})$ respectively, where $\mathbf{1}_D$ denotes a D-dimensional vector whose components are 1. Then we draw φ_k^e from the mixture of η_k^e and the translation of η_k^c. It is formulated as:

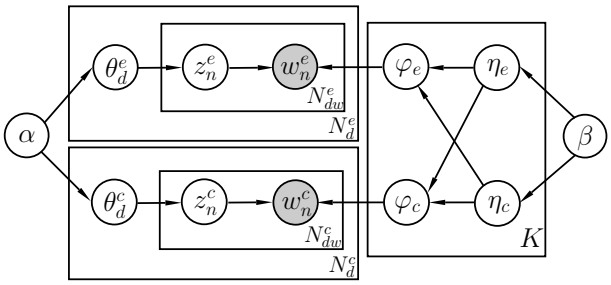


Figure 1: Our proposed graphical model

$$\varphi_k^e \propto \lambda(\eta_k^c)^T \mathbf{M}^{c \to e} + (1-\lambda)\eta_k^e \quad \text{where} \quad \eta^e, \eta^c \sim Dir(\beta) \tag{1}$$

where $\lambda \in (0, 1)$ is a parameter which balances the nature of original topics and transferred information from the other language. $\mathbf{M}^{c \to e}$ is a mapping $|\Lambda^c| \times |\Lambda^e|$ matrix from Λ^c to Λ^e. Each element $M_{ij}^{c \to e}$ is the mapping occurrence probability of the English term w_j^e given the Chinese term w_i^c in the set of news reader comments. This probability is calculated as:

$$M_{ij}^{c \to e} = \frac{C(w_j^e) + 1}{|T(w_i^c)| + \sum_{w^e \in T(w_i^c)} C(w^e)} \tag{2}$$

where $C(w_j^e)$ is the count of w_j^e in all news reader comments and $T(w_i^c)$ is the set of English translations of w_i^c found in the bilingual dictionary. The "add-one" smoothing is adopted. Note that the sum of each row is equal to 1. Using the same principle, we can derive φ_k^c which can be formulated as:

$$\varphi_k^c \propto \lambda(\eta_k^e)^T M^{e \to c} + (1-\lambda)\eta_k^c \quad \text{where} \quad \eta^e, \eta^c \sim Dir(\beta) \tag{3}$$

As a result, the incorporation of η_k^e and η_k^c on the topic level encourages the word distribution φ_k^e and φ_k^c to share common semantic components of reader comments in different languages.

The upper left plate in Figure 1 represents English reader comments. N_d^e denotes the number of English reader comments and N_{dw}^e denotes

the number of words in the English comment d^e. Each English reader comment d^e is characterized by a K-dimensional topic membership vector θ_d^e, which is assumed to be generated by the prior $Dir(\alpha \cdot \mathbf{1}_K)$. For each word w_n^e in an English comment d^e, we generate the topic z_e^n from θ_d^e. We generate the word w_n^e from the corresponding distribution φ_k^e.

The bottom left plate in Figure 1 represents Chinese reader comments. Similarly, we generate the topic distribution θ_d^c from the prior $Dir(\alpha \cdot \mathbf{1}_K)$. The topic z_c^n of each word w_n^c in a Chinese comment d^c is generated from θ_d^c. We generate word w_n^c from the corresponding distribution φ_k^c.

The generative process is formally depicted as:

- For each topic $k \in \mathcal{K}$
  - choose auxiliary distributions $\eta_k^e \sim Dir(\beta \cdot \mathbf{1}_{|\Lambda^e|})$ and $\eta_k^c \sim Dir(\beta \cdot \mathbf{1}_{|\Lambda^c|})$
  - choose English word distribution φ_k^e and φ_k^c using Eq. 1 and Eq. 3 respectively.
- For each English comment $d^e \in \mathcal{E}$, choose $\theta_d^e \sim Dir(\alpha \cdot \mathbf{1}_K)$
  - For each position n in d^e
    - draw $z_n^e \sim$ Multi(θ_d^e)
    - draw $w_n^e \sim$ Multi(φ_k^e)
- For each Chinese comment $d^c \in \mathcal{C}$, choose $\theta_d^c \sim Dir(\alpha \cdot \mathbf{1}_K)$
  - For each position n in d^c
    - draw $z_n^c \sim$ Multi(θ_d^c)
    - draw $w_n^c \sim$ Multi(φ_k^c)

Note that for simplicity, we present our model on the bilingual setting of Chinese and English. It can be extended to multilingual setting via introducing auxiliary distributions for each language. Each topic word distribution for each language is generated by the convex combination of all the auxiliary distributions.

3.2 Posterior Inference

In order to decouple the inference of z_n and φ_k for each language, we develop a partially collapsed Gibbs method which just discards θ_d^e and θ_d^c. Given φ_k^e, we sample the new assignments of the topic z_{di}^e in English news reader comments d^e with the following conditional probability:

$$P(z_{di}^e = k|\mathbf{z}^{e,\neg i}, W^e, \alpha, \varphi_k^e) \propto (N_{dk}^{e,\neg i} + \alpha_k) \times \varphi_k^e \tag{4}$$

where $\mathbf{z}^{e,\neg i}$ denotes the topic assignments except the assignment of the ith word. N_{dk}^e is the number of words in English document d^e whose topics are assigned to k. Similarly, we sample z_{di}^c with the following equation:

$$P(z_{di}^c = k|\mathbf{z}^{c,\neg i}, W^c, \alpha, \varphi_k^c) \propto (N_{dk}^{c,\neg i} + \alpha_k) \times \varphi_k^c \tag{5}$$

Given the topic assignments, the probability of the entire comment set can be:

$$p(W|\mathbf{z}, \varphi_k^e, \varphi_k^c) = \prod_{w \in \Lambda^e} (\varphi_{kw}^e)^{N_{kw}^e} \times \prod_{w \in \Lambda^c} (\varphi_{kw}^c)^{N_{kw}^c} \tag{6}$$

where N_{kw}^e is the number of words w in English news reader comments assigned to the topic k and N_{kw}^c is the number of words w in Chinese news reader comments assigned to the topic k.

Using Eq. 6, we can obtain the posterior likelihood related to η_k^e and η_k^c:

$$\begin{aligned}\mathcal{L}_{MAP} = & \\ & \sum_{w_i \in \Lambda^e} N_{kw_i}^e \log(\lambda \sum_{w_j \in \Lambda^c} M_{ji}^{c \to e} \eta_{kw_j}^c + (1-\lambda)\eta_{kw_i}^e) \\ & + \sum_{w_i \in \Lambda^c} N_{kw_i}^c \log(\lambda \sum_{w_j \in \Lambda^e} M_{ji}^{e \to c} \eta_{kw_j}^e + (1-\lambda)\eta_{kw_i}^c) \\ & + \sum_{w_i \in \Lambda^e} (\beta - 1) \log \eta_{kw_i}^e + \sum_{w_i \in \Lambda^c} (\beta - 1) \log \eta_{kw_i}^c\end{aligned} \tag{7}$$

We optimize Eq. 7 under the constraints of $\sum_{w_i \in \Lambda^e} \eta_{kw_i}^e = 1$ and $\sum_{w_i \in \Lambda^c} \eta_{kw_i}^c = 1$. Using the fixed-point method, we obtain the update

Algorithm 1 Partially Collapsed Gibbs Sampling for MCTA

```
Initialize z, φ_k^e, φ_k^c, η_k^e, η_k^c
for iter = 1 to Max_iter do
    for each English comment d in E do
        for each word w_n^e in d do
            draw z_n^e using Eq. 4
        end for
    end for
    for each Chinese comment d in C do
        for each word w_n^c in d do
            draw z_n^c using Eq. 5.
        end for
    end for
    Update η_k^e, η_k^c by Eq. 8 and Eq. 9
    Update φ_k^e, φ_k^c according to Eq. 1 and Eq. 3
end for
Output θ_dk by Eq. 10
```

equations of $\eta^e_{kw_t}$ and $\eta^c_{kw_t}$ shown in Eq. 8 and Eq. 9.

$$\eta^e_{kw_t} \propto [\frac{(1-\lambda)N^e_{kw_t}}{\lambda \sum_{w_j \in \Lambda^c} M^{c\to e}_{jt} \eta^c_{kw_j} + (1-\lambda)\eta^e_{kw_t}} + \sum_{w_i \in \Lambda^c} \frac{\lambda N^c_{kw_i} M^{e\to c}_{ti}}{\lambda \sum_{w_j \in \Lambda^e} M^{e\to c}_{ji} \eta^e_{kw_j} + (1-\lambda)\eta^c_{kw_i}}]\eta^e_{kw_t} + \beta \tag{8}$$

$$\eta^c_{kw_t} \propto [\frac{(1-\lambda)N^c_{kw_t}}{\lambda \sum_{w_j \in \Lambda^e} M^{e\to c}_{jt} \eta^e_{kw_j} + (1-\lambda)\eta^c_{kw_t}} + \sum_{w_i \in \Lambda^e} \frac{\lambda N^e_{kw_i} M^{c\to e}_{ti}}{\lambda \sum_{w_j \in \Lambda^c} M^{c\to e}_{ji} \eta^c_{kw_j} + (1-\lambda)\eta^e_{kw_i}}]\eta^c_{kw_t} + \beta \tag{9}$$

Moreover, the posterior estimates for the topic distribution θ_d can be computed as follows.

$$\theta_{dk} = \frac{N_{dk} + \alpha}{\sum_{k \in \mathcal{K}} N_{dk} + K\alpha} \tag{10}$$

The whole detailed algorithm is depicted in Algorithm 1. When $\lambda = 0$, the updated equations of η^e_k and η^c_k can be simplified as:

$$\begin{aligned} \eta^e_{kw_t} &\propto N^e_{kw_t} + \beta \\ \eta^c_{kw_t} &\propto N^c_{kw_t} + \beta \end{aligned} \tag{11}$$

Then we have:

$$\begin{aligned} \varphi^e_k &\sim Dir(N^e_{kw_1} + \beta, N^e_{kw_2} + \beta, \dots) \\ \varphi^c_k &\sim Dir(N^c_{kw_1} + \beta, N^c_{kw_2} + \beta, \dots) \end{aligned} \tag{12}$$

Therefore, the algorithm degrades to a Gibbs sampler of LDA.

4 Experiments

4.1 Data Set and Preprocessing

We have prepared a data set by collecting English and Chinese comments from different regions reflecting different culture for some significant events as depicted in Table 1. The English reader comments are collected from Yahoo[1] and the Chinese reader comments are collected from Sina News[2]. We first remove news reader comments whose length is less than 5 words. We remove the punctuations and the stop words. For English comments, we also stem each word to its root form using Porter Stemmer (Porter, 1980). For the Chinese reader comments, we use the Jieba package[3] to segment and remove Chinese stop words. We utilize an English-Chinese dictionary from MDBG[4].

Event	Title	#English comments	#Chinese comments
1	MH370 flight accident	8608	5223
2	ISIS in Iraq	6341	3263
3	Ebola occurs	2974	1622
4	Taiwan Crashed Plane	6780	2648
5	iphone6 publish	5837	4352
6	Shooting of Michael Brown	17547	3693
7	Charlie Hebdo shooting	1845	551
8	Shanghai stampede	3824	3175
9	Lee Kuan Yew death	2418	1534
10	AIIB foundation	7221	3198

Table 1: The statistics for the data set

4.2 Comparative Methods

The PCLSA model proposed by Zhang et al. (2010) can be regarded as the state-of-the-art model for detecting latent common topics from multilingual text documents. We implemented PCLSA as one of the comparative methods in our experiments.

Another comparative model used in the experiment is LDA (Blei et al., 2003), which can generate K English topics and K Chinese topics from English and Chinese reader comments respectively. Then we translate Chinese topics into English topics and use symmetric KL divergence to align translated Chinese topics with original English topics. Each aligned topic pair is regarded as a cultural-common topic.

4.3 Experiment Settings

For each event, we partitioned the comments into a subset of 90% for the graphical model parameter estimation. The remaining 10% is used as a holdout data for the evaluation of the *CCP* metric as discussed in Section 4.4.1. We repeated the runs five times. For each run, we randomly split the comments to obtain the holdout data. As a result, we have five runs for our method as well as comparative methods. We make use of the holdout data of one event, namely the event "MH370

[1] http://news.yahoo.com

[2] http://news.sina.com.cn/world/

[3] https://github.com/fxsjy/jieba

[4] http://www.mdbg.net/chindict/chindict.php?page=cc-cedict

Flight Accident", to estimate the number of topics K for all models and λ in Eq. 1 for our model. The setting of K is described in Section 4.4.3. We set $\lambda = 0.5$ after tuning. For hyper-parameters, we set α to 0.5 and β to 0.01. When performing our Gibbs algorithm, we set the maximum iteration number as 1000, and the burn-in sweeps as 100.

4.4 Cultural-common Topic Evaluation

We conduct quantitative experiments to evaluate how well our MCTA model can discover cultural-common topics.

4.4.1 Evaluation Metrics

We use two metrics to evaluate the topic quality. The first metric is the "cross-collection perplexity" measure denoted as *CCP* which is similar to the one used in Zhang et al. (2010). The *CCP* of high quality cultural-common topics should be lower than those topics which are not shared by the English and Chinese reader comments. The calculation of *CCP* consists of two steps: *1)* For each $k \in \mathcal{K}$, we translate φ_k^e into Chinese word distribution $T(\varphi_k^e)$ and translate φ_k^c English word distribution $T(\varphi_k^c)$. To translate φ_k^e and φ_k^c, we look up the bilingual dictionary and conduct word-to-word translation. If one word has several translations, we distribute its probability mass equally to each English translation. *2)* We use $T(\varphi_k^e)$ to fit the holdout Chinese comments $\mathcal{C}$ and $T(\varphi_k^c)$ to fit the holdout English comments $\mathcal{E}$ using Eq. 13 (Blei et al., 2003). Eq. 13 depicts the calculation of *CCP*. The lower the *CCP* value is, the better the performance is.

$$
\begin{aligned}
&CCP = \\
&\frac{1}{2}\exp\{-\frac{\sum_{d\in\mathcal{E}}\sum_{w\in d}\sum_{k\in\mathcal{K}}\log p(k|\theta_d)p(w|T(\varphi_k^c))}{\sum_{d\in\mathcal{E}} N_d^e}\}+ \\
&\frac{1}{2}\exp\{-\frac{\sum_{d\in\mathcal{C}}\sum_{w\in d}\sum_{k\in\mathcal{K}}\log p(k|\theta_d)p(w|T(\varphi_k^e))}{\sum_{d\in\mathcal{C}} N_d^c}\}
\end{aligned}
\tag{13}
$$

For each detected common topic, we wish to evaluate the degree of commonality. We design another metric called "topic commonality distance" denoted by *TCD*. We first evaluate the KL-divergence between the English topic and translated Chinese topic. We also evaluate the KL-divergence between the Chinese topic and translated English topic. Then *TCD* is computed as the average sum of the two KL-divergences. The lower the *TCD* measure is, the better the topic is.

Event	LDA	PCLSA	MCTA
1	1963.57	1842.24	1784.05
2	1940.03	1831.55	1756.92
3	1958.09	1905.43	1808.01
4	1916.49	1847.16	1775.32
5	1901.44	1797.92	1744.07
6	1916.70	1853.66	1786.77
7	1945.22	1897.15	1824.10
8	1942.29	1862.14	1749.43
9	1943.53	1856.70	1739.66
10	1866.23	1815.44	1749.49
avg.	1929.36	1850.94	**1771.78**

Table 2: Topic quality evaluation as measured by *CCP*

The topic detected by PCLSA is a mixture of English and Chinese words. We obtain English representation and Chinese representation of the topic by the conditional probabilities as given in Eq. 14.

$$
\begin{aligned}
p(w^e|\varphi_k^e) &= \frac{p(w^e|\varphi_k)}{\sum_{w\in\Lambda^e} p(w|\varphi_k)} \\
p(w^c|\varphi_k^c) &= \frac{p(w^c|\varphi_k)}{\sum_{w\in\Lambda^c} p(w|\varphi_k)}
\end{aligned}
\tag{14}
$$

4.4.2 Experimental Results

The average *CCP* values of the three models are shown in Table 2. Our MCTA model achieves the best performance compared with PCLSA and LDA. Both MCTA and PCLSA achieve a better *CCP* than LDA because they can bridge the language gap in the multilingual news reader comments to some extent. Compared with PCLSA, our MCTA model demonstrates a 4.2% improvement. Our MCTA model provides a better characterization of the collections. One reason is that our MCTA model learns the word distribution of cultural-common topics using an effective topic modeling with a prior Dirichlet distribution. It is similar to the advantage of LDA over PLSA. Moreover, the bilingual constraints in PCLSA cannot handle the original natural word co-occurrence well in each language. In contrast, MCTA represents cultural-common topics as a mixture of the original topics and the translated topics, which capture the comment semantics more effectively.

The average *TCD* of three models are shown in Table 3. Our MCTA outperforms the two comparative methods. The cultural-common topics iden-

Event	LDA	PCLSA	MCTA
1	0.029	0.0075	0.0042
2	0.029	0.0072	0.0043
3	0.033	0.0076	0.0046
4	0.031	0.0075	0.0046
5	0.033	0.0086	0.0069
6	0.029	0.0066	0.0058
7	0.036	0.0080	0.0044
8	0.033	0.0079	0.0034
9	0.034	0.0088	0.0036
10	0.029	0.0067	0.0036
avg.	0.032	0.0076	**0.0045**

Table 3: Topic quality evaluation as measured by *TCD*

Event	LDA	PCLSA	MCTA
1	0.128	0.117	0.138
2	0.144	0.126	0.158
3	0.122	0.117	0.120
4	0.138	0.138	0.169
5	0.128	0.109	0.152
6	0.134	0.138	0.152
7	0.103	0.108	0.111
8	0.110	0.099	0.124
9	0.080	0.085	0.096
10	0.138	0.133	0.154
avg.	0.122	0.117	**0.137**

Table 4: Topic coherence evaluation

tified by MCTA have better topic commonality because our MCTA model can capture the common semantics between news reader comments in different languages.

4.4.3 Determining Number of Topics

As mentioned in Section 4.3, we use the hold-out data of one event to determine K. For each $\lambda \in \{0.2, 0.5, 0.8\}$, we vary K in the range of $[5, 200]$. Figure 2 depicts the effect of K on the cross-collection perplexity as measured by *CCP*. We can see that *CCP* decreases with the increase of the number of topics. Moreover, through manual inspection we observed that when K is 30 or more, even though *CCP* decreases, the topics will be repeated. Similar observations for the number of topics can be found in Paul and Girju (2009). Therefore, we set $K = 30$. We can also see that our model is not very sensitive to the balance parameter λ.

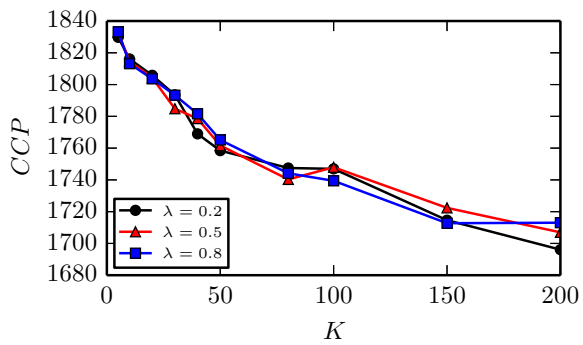


Figure 2: The effect of K

4.5 Topic Coherence Evaluation

We also evaluate the coherence of topics generated by PCLSA and MCTA, which indicates the interpretability of topics. Following Newman et al. (2010), we use a pointwise mutual information (PMI) score to measure the topic coherence. We compute the average PMI score of top 20 topic word pairs using Eq. 15. Newman et al. (2010) observed that it is important to use an external data set to evaluate PMI. Therefore, we use a 20-word sliding window in Wikipedia (Shaoul, 2010) to identify the co-occurrence of word pairs.

$$PMI(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i)P(w_j)} \qquad (15)$$

The experimental results are shown in Table 4. We can see that our MCTA model generally improves the coherence of the learned topics compared with PCLSA. The word-to-word bilingual constraints in PCLSA are not as effective. On the other hand, our MTCA model incorporates the bilingual translations using auxiliary distributions which incorporate word distributions from the other language on the topic level and can capture common semantics of multilingual reader comments.

5 Application and Case Study

We present an application for news comment digest and show some examples of detected cultural-common discussion topics in Figure 3. Under each event, the system can group reader comments into cultural-common discussion topics which can capture common concerns of readers in different languages. For each common topic, it shows top ranked words and corresponding reader comments

Event: MH370 Flight Accident	
Topic Terms	Reader Comments
family hope love dead people victim passenger sad sorry life 家庭(family) 家属(family) 家人(family) 亲人(family) 失望(disappoint) 希望(hope) 心酸(sad) 愿(wish) 痛(pain) 心痛(sad)	I feel sorry for the families of the victims of this flight - this aircraft piece being found probably brings that terrible day back
	I feel so sorry for the relatives of the missing passengers who are doomed to spend the rest of their lives getting their hopes continuously...
	The family members should now begin to have a closure as the plane's flaperon has been identified. The Australians have been proven correct as...
	时间真快，一年半的时间过去了，不知那些失去亲人的朋友们走出悲痛了没有？唯愿逝者在天堂安息，生者在人间安康！
	家属朋友们，失去亲人的痛苦的，但生活是美好的，一定要好好生活，让逝者安心！
	每一次的提起370 就会让那些失去亲人的家人心痛折磨一次！
	⋮
ocean island search India mile area locate Australia drift west 洋流(ocean currents) 印度洋(Indian ocean) 区域(Region) 海里(mile) 搜索(search) 搜寻(search) 搜救(rescure) 海底(sea floor) 澳洲(Australia) 海域(sea area)	They were looking in the West Australian current. That would have brought the part to the north of Australia. If it got into I equatorial current...
	They need to start their sonar scans about 1000 miles south of the tip of India seeing how the currents in that ocean work, and how long it took for that piece to float to the island so far out. It's pretty simple to estimate seeing how Fukushima fishing boats travelled a set distance over a set time, given a set current...
	look at current maps. well off the western coast of Aus is the S. Equitorial Current in the Indian Ocean which flows in a circular counter clockwise pattern. It most certainly could have come from a plane that crashed off the AUS coast.
	这么多阴谋论者说这是美国搞的鬼，我只能呵呵了，美国的调查结论说是在南印度洋，在澳大利亚那边，现在发现的位置是不是和美国的调查结果一致？洋流的运动方向和推测地点、残骸地点是否符合？为什么一定要把空难说成某国的阴谋才甘心？
	南印度洋的洋流是自东向西，这个残骸落在这里，那么飞机应该坠毁在东方的海面上。
	即使这片残骸属于MH370客机，在留尼汪被发现也并不意味着飞机的失事地点就在留尼汪。假设飞机在澳大利亚海域坠毁，其残骸很有可能被洋流带到印度洋，一年以后被海浪冲上安德烈海滩.
	⋮
⋮	

Event:ISIS in Iraq	
Topic Terms	Reader Comments
muslim islam religion world christian god people believe jew human 信仰(belief) 宗教(religion) 世界(world) 相信(believe) 全世界(world) 伊斯兰(Islam) 穆斯林(muslim) 人(people) 犹太(jew) 人类(human)	1 don't understand Muslims, Islam or the Holy Qur'an! The aim of Islam is not to instil Sharia over the entire world, Islam preaches that you believe in God worship Him alone and do right good by your belief.....
	Oh, I get it. It's about the badness of Muslims being humbled and humiliated in prison by Americans. But IS rapes and mutilates and pilliages...
	If there was no Muslim religion in Iraq, there would be no ISIS because there would have been no necessity for a thug like Saddam to control...
	ISIS是个宗教极端组织没错，但是如果ISIS没有下这个令，而是被故意栽赃，其用意显然不在针对ISIS本身
	宗教不是祸首，真正的魁首是打着宗教旗号的极端分子，都说国人没信仰但是一样有右翼激进分子，激进的民族主义，然后是民粹，最后是种族主义，纳粹不也是这么一步一步上台的么，历史总是似曾相识。
	可以看出一切邪恶都是出自宗教！宗教欺骗人类的另一面！以后别拿我们汉族没有信仰来说事了！看看你们信仰的后果吧！
	⋮
⋮	

Event:AIIB Foundation	
Topic Terms	Reader Comments
bank aiib world imf asian develop investment institution infrastructure member 银行(bank) 金融(finance) 世界银行(The World Bank) 世行(The World Bank) 金融机构(Finance institution) 亚洲开发银行(Asian Development Bank) 成员(member) 国际货币基金组织(IMF) 国际(International) 世界(world)	Looks like all the rats are jumping off the sinking world bank and IMF ship. America has pushed their bulling ways long enough and people are...
	The Federal Reserve, the World Bank, The IMF, and the BIS are failed, self-serving institutions. One can only hope that China will stimulate world growth and the suppression by the west will finally come to an end. The US dollar no longer deserves to be the world's reserve currency.
	Bank shopping !!! No more stranglehold by the IMF and World Bank. If Ukraine had only waited another year. Too bad.
	把米国和日本排除在亚投行之外，让他们自己单独经营亚洲开发银行和世界银行！[哈哈]
	欧洲国家被美国坑惨了，世界银行、国际货币基金的钱都在为美国服务，反过来美国又利用乌克兰危机打压欧元，如今欧洲国家看明白了，还是中国靠普。
	世界银行和亚洲开发银行都对亚投行表示欢迎，明显有些言不由衷。信他才怪
	⋮

Figure 3: Some sample common discussion topics of some events

according to θ^e_{dk} and θ^c_{dk}. Considering the event "MH370 Flight accident", it shows two of the detected cultural-common topics. The first one indicates that readers pray for the family in the accident and the second one is related to the search of the crashed plane. For the common topic about praying for the family, we can see that the topics contain both English words and Chinese words which are very relevant and share common semantics of "family" and "hope". Moreover, the corresponding English and Chinese reader comments, both of which mention the family in the accident, illustrate a high coherent common discussion topic. Similarly for the second common topic, there are common semantics between English and Chinese top ranked words about the search of the crashed plane. Some of the English comments and Chinese comments mention the query of the position of the crashed plane. Interesting common topics are also generated for other events, such as the common topic of religion for the event "ISIS in Iraq" and the topic of economic organization for the event "AIIB foundation".

6 Conclusions

We investigate the task of cultural-common discussion topic detection from multilingual news reader comments. To tackle the task, we develop a new model called MCTA which can cope with the language gap and extract coherent cultural-common topics from multilingual news reader comments. We also develop a partially collapsed Gibbs sampler which incorporates the term translation relationship into the detection of cultural-common topics effectively for model parameter learning.

Acknowledgments

The work described in this paper is substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Code: 14203414) and the Direct Grant of the Faculty of Engineering, CUHK (Project Code: 4055034). This work is also affiliated with the CUHK MoE-Microsoft Key Laboratory of Human-centric Computing and Interface Technologies. We also thank the anonymous reviewers for their insightful comments.

References

Prakhar Biyani, Cornelia Caragea, and Narayan Bhamidipati. 2015. Entity-specific sentiment classification of yahoo news comments. *arXiv preprint arXiv:1506.03775*.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David M Blei. 2009. Multilingual topic models for unaligned text. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 75–82.

Jordan Boyd-Graber and Philip Resnik. 2010. Holistic sentiment analysis across languages: Multilingual supervised latent dirichlet allocation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 45–55.

S Tamer Cavusgil, Gary Knight, John R Riesenberger, Hussain G Rammal, and Elizabeth L Rose. 2014. *International business: strategy, management and the new realities*. Pearson Australia.

Elena Erosheva, Stephen Fienberg, and John Lafferty. 2004. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5220–5227.

Kosuke Fukumasu, Koji Eguchi, and Eric P Xing. 2012. Symmetric correspondence topic models for multilingual text analysis. In *Advances in Neural Information Processing Systems*, pages 1286–1294.

Honglei Guo, Huijia Zhu, Zhili Guo, Xiaoxun Zhang, and Zhong Su. 2010. Opinionit: a text mining system for cross-lingual opinion analysis. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1199–1208.

Thomas Hofmann. 2001. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196.

Jagadeesh Jagarlamudi and Hal Daumé III. 2010. Extracting multilingual topics from unaligned comparable corpora. In *Proceedings of the 32nd European Conference on IR Research*, pages 444–456. Springer.

Zheng Lin, Xiaolong Jin, Xueke Xu, Weiping Wang, Xueqi Cheng, and Yuanzhuo Wang. 2014. A cross-lingual joint aspect/sentiment model for sentiment analysis. In *Proceedings of the 23rd ACM International Conference on Information and Knowledge Management*, pages 1089–1098.

Bin Lu, Chenhao Tan, Claire Cardie, and Benjamin K Tsou. 2011. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 320–330.

Steve Melluish. 2014. Globalization, culture and psychology. *International Review of Psychiatry*, 26(5):538–543.

David Mimno, Hanna M Wallach, Jason Naradowsky, David A Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 880–889.

David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. 2010. Automatic evaluation of topic coherence. In *Human Language Technologies: the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining multilingual topics from wikipedia. In *Proceedings of the 18th International Conference on World Wide Web*, pages 1155–1156.

Michael Paul and Roxana Girju. 2009. Cross-cultural analysis of blogs and forums with mixed-collection topic models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1408–1417.

Nanyun Peng, Yiming Wang, and Mark Dredze. 2014. Learning polylingual topic models from code-switched social media documents. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 674–679.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Radityo Eko Prasojo, Mouna Kacimi, and Werner Nutt. 2015. Entity and aspect extraction for organizing news comments. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 233–242.

Cyrus Shaoul. 2010. The westbury lab wikipedia corpus. *Edmonton, AB: University of Alberta*.

Ivan Vulić and Marie-Francine Moens. 2014. Probabilistic models of cross-lingual semantic similarity in context based on latent cross-lingual concepts induced from comparable data. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 349–362.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying word translations from comparable corpora using latent topic models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 479–484.

ChengXiang Zhai, Atulya Velivelli, and Bei Yu. 2004. A cross-collection mixture model for comparative text mining. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 743–748.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-lingual latent topic extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1128–1137.

Tao Zhang, Kang Liu, and Jun Zhao. 2013. Cross lingual entity linking with bilingual topic model. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 2218–2224.

A Discriminative Topic Model using Document Network Structure

Weiwei Yang
Computer Science
University of Maryland
College Park, MD
wwyang@cs.umd.edu

Jordan Boyd-Graber
Computer Science
University of Colorado
Boulder, CO
Jordan.Boyd.Graber@colorado.edu

Philip Resnik
Linguistics and UMIACS
University of Maryland
College Park, MD
resnik@umd.edu

Abstract

Document collections often have links between documents—citations, hyperlinks, or revisions—and which links are added is often based on topical similarity. To model these intuitions, we introduce a new topic model for documents situated within a network structure, integrating latent blocks of documents with a max-margin learning criterion for link prediction using topic- and word-level features. Experiments on a scientific paper dataset and collection of webpages show that, by more robustly exploiting the rich link structure within a document network, our model improves link prediction, topic quality, and block distributions.

1 Introduction

Documents often appear within a network structure: social media mentions, retweets, and follower relationships; Web pages by hyperlinks; scientific papers by citations. Network structure interacts with the topics in the text, in that documents linked in a network are more likely to have similar topic distributions. For instance, a citation link between two papers suggests that they are about a similar field, and a mentioning link between two social media users often indicates common interests. Conversely, documents' similar topic distributions can suggest links between them. For example, topic model (Blei et al., 2003, LDA) and block detection papers (Holland et al., 1983) are relevant to our research, so we cite them. Similarly, if a social media user A finds another user B with shared interests, then A is more likely to follow B.

Our approach is part of a natural progression of network modeling in which models integrate more information in more sophisticated ways. Some past methods only consider the network itself (Kim and Leskovec, 2012; Liben-Nowell and Kleinberg, 2007), which loses the rich information in text. In other cases, methods take both links and text into account (Chaturvedi et al., 2012), but they are modeled separately, not jointly, limiting the model's ability to capture interactions between the two. The *relational topic model* (Chang and Blei, 2010, RTM) goes further, jointly modeling topics and links, but it considers only pairwise document relationships, failing to capture network structure at the level of groups or *blocks* of documents.

We propose a new joint model that makes fuller use of the rich link structure within a document network. Specifically, our model embeds the *weighted stochastic block model* (Aicher et al., 2014, WSBM) to identify blocks in which documents are densely connected. WSBM basically categorizes each item in a network probabilistically as belonging to one of L blocks, by reviewing its connections with each block. Our model can be viewed as a principled probabilistic extension of Yang et al. (2015), who identify blocks in a document network deterministically as *strongly connected components* (SCC). Like them, we assign a distinct Dirichlet prior to each block to capture its topical commonalities. Jointly, a linear regression model with a discriminative, max-margin objective function (Zhu et al., 2012; Zhu et al., 2014) is trained to reconstruct the links, taking into account the features of documents' topic and word distributions (Nguyen et al., 2013), block assignments, and inter-block link rates.

We validate our approach on a scientific paper abstract dataset and collection of webpages, with citation links and hyperlinks respectively, to predict links among previously unseen documents and from those new documents to training documents. Embedding the WSBM in a network/topic

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 686–696,
Berlin, Germany, August 7-12, 2016.

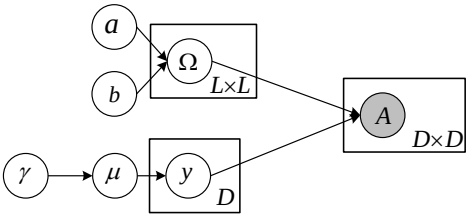


Figure 1: Weighted Stochastic Block Model

model leads to substantial improvements in link prediction over previous models; it also improves block detection and topic interpretability. The key advantage in embedding WSBM is its flexibility and robustness in the face of noisy links. Our results also lend additional support for using max-margin learning for a "downstream" supervised topic model (McAuliffe and Blei, 2008), and that predictions from lexical as well as topic features improves performance (Nguyen et al., 2013).

The rest of this paper is organized as follows. Section 2 introduces two previous link-modeling methods, WSBM and RTM. Section 3 presents our methods to incorporate block priors in topic modeling and include various features in link prediction, as well as the aggregated discriminative topic model whose posterior inference is introduced in Section 4. In Section 5 we show how our model can improve link prediction and (often) improve topic coherence.

2 Dealing with Links

2.1 Weighted Stochastic Block Model

WSBM (Aicher et al., 2014) is a generalized *stochastic block model* (Holland et al., 1983; Wang and Wong, 1987, SBM) and predicts non-negative integer-weight links, instead of binary-weight links. A block is a collection of documents which are densely connected with each other but sparsely connected with documents in other blocks. WSBM assumes that a document belongs to exactly one block. A link connecting two documents in blocks l and l' has a weight generated from a Poisson distribution with parameters $\Omega_{l,l'}$ which has a Gamma prior with parameters a and b, as Figure 1 shows.

The whole generative process is:

1. For each pair of blocks $(l, l') \in \{1, \dots, L\}^2$
   (a) Draw inter-block link rate $\Omega_{l,l'} \sim \text{Gamma}(a, b)$
2. Draw block distribution $\boldsymbol{\mu} \sim \text{Dir}(\gamma)$
3. For each document $d \in \{1, \dots, D\}$
   (a) Draw block assignment $y_d \sim \text{Mult}(\boldsymbol{\mu})$
4. For each link $(d, d') \in \{1, \dots, D\}^2$
   (a) Draw link weight $A_{d,d'} \sim \text{Poisson}(\Omega_{y_d, y_{d'}})$

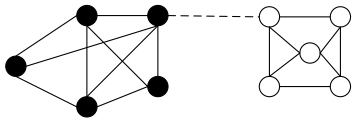


Figure 2: SCC can be distracted by spurious links connecting two groups, while WSBM maintains the distinction.

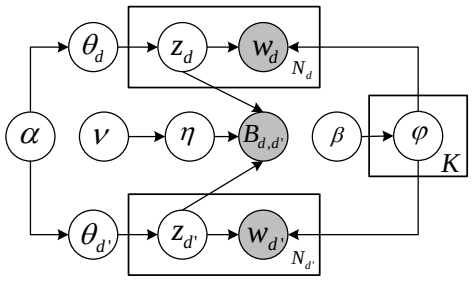


Figure 3: A Two-document Segment of RTM

WSBM is a probabilistic block detection algorithm and more robust than some deterministic algorithms like SCC, which is vulnerable to noisy links. For instance, we would intuitively say Figure 2 has two blocks—as denoted by coloring—whether or not the dashed link exists. If the dashed link does not exist, both WSBM and SCC can identify two blocks. However, if the dashed link does exist, SCC will return only one big block that contains all nodes, while WSBM still keeps the nodes in two reasonable blocks.

2.2 Relational Topic Model

RTM (Chang and Blei, 2010) is a downstream model that generates documents and links simultaneously (Figure 3). Its generative process is:

1. For each topic $k \in \{1, \dots, K\}$
   (a) Draw word distribution $\boldsymbol{\phi_k} \sim \text{Dir}(\beta)$
   (b) Draw topic regression parameter $\eta_k \sim \mathcal{N}(0, \nu^2)$
2. For each document $d \in \{1, \dots, D\}$
   (a) Draw topic distribution $\boldsymbol{\theta_d} \sim \text{Dir}(\alpha)$
   (b) For each token $t_{d,n}$ in document d
      i. Draw topic assignment $z_{d,n} \sim \text{Mult}(\boldsymbol{\theta_d})$
      ii. Draw word $w_{d,n} \sim \text{Mult}(\boldsymbol{\phi}_{\boldsymbol{z_{d,n}}})$
3. For each explicit link (d, d')
   (a) Draw link weight $B_{d,d'} \sim \Psi(\cdot \mid \boldsymbol{z_d}, \boldsymbol{z_{d'}}, \boldsymbol{\eta})$

In the inference process, the updating of topic assignments is guided by links so that linked documents are more likely to have similar topic distributions. Meanwhile, the linear regression (whose

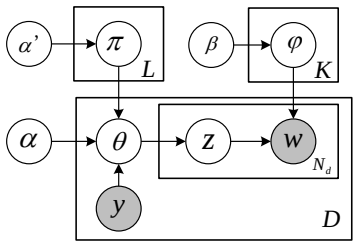


Figure 4: Graphical Model of BP-LDA

output is fed into link probability function Ψ) is updated to maximize the network likelihood using current topic assignments.

3 Discriminative Topic Model with Block Prior and Various Features

Our model is able to identify blocks from the network with an embedded WSBM, extract topic patterns of each block as prior knowledge, and use all this information to reconstruct the links.

3.1 LDA with Block Priors (BP-LDA)

As argued in the introduction, linked documents are likely to have similar topic distributions, which can be generalized to documents in the same block. Inspired by this intuition and the block assignment we obtain in the previous section, we want to extract some prior knowledge from these blocks. Thus we propose an LDA with block priors, hence BP-LDA, as shown in Figure 4, which has the following generative process:

1. For each topic $k \in \{1, \dots, K\}$
   (a) Draw word distribution $\boldsymbol{\phi_k} \sim \text{Dir}(\beta)$
2. For each block $l \in \{1, \dots, L\}$
   (a) Draw topic distribution $\boldsymbol{\pi_l} \sim \text{Dir}(\alpha')$
3. For each document $d \in \{1, \dots, D\}$
   (a) Draw topic distribution $\boldsymbol{\theta_d} \sim \text{Dir}(\alpha \boldsymbol{\pi_{y_d}})$
   (b) For each token $t_{d,n}$ in document d
      i. Draw topic assignment $z_{d,n} \sim \text{Mult}(\boldsymbol{\theta_d})$
      ii. Draw word $w_{d,n} \sim \text{Mult}(\boldsymbol{\phi_{z_{d,n}}})$

Unlike conventional LDA, which uses an uninformative topic prior, BP-LDA puts a Dirichlet prior $\boldsymbol{\pi}$ on each block to capture the block's topic distribution and use it as an informative prior when drawing each document's topic distribution. In other words, a document's topic distribution—i.e., what the document is about—is not just informed by the words present in the document but the broader context of its network neighborhood.

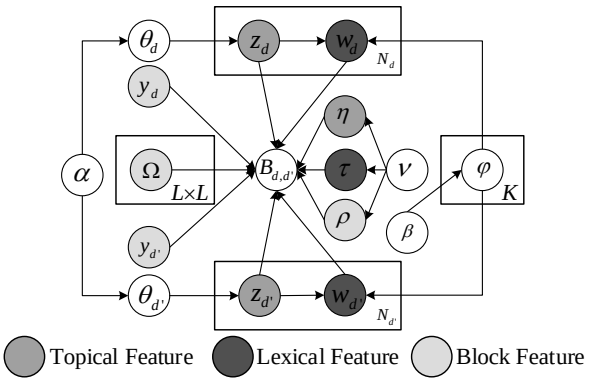


Figure 5: A two-document segment of VF-RTM. Various features are denoted by grayscale. $B_{d,d'}$ is observed, but we keep it in white background to avoid confusion.

3.2 RTM with Various Features (VF-RTM)

Building on Chang and Blei (2010), we want to generate the links between documents based on various features, hence VF-RTM. In addition to topic distributions, VF-RTM also includes documents' word distributions (Nguyen et al., 2013) and the link rate of two documents' assigned blocks, with the intent that these additional features improve link generation. VF-RTM involves the relationship between a pair of documents, so it is difficult to show the whole model; therefore Figure 5 illustrates with a two-document segment. The generative process is:

1. For each pair of blocks $(l, l') \in \{1, \dots, L\}^2$
   (a) Draw block regression parameter $\rho_{l,l'} \sim \mathcal{N}(0, \nu^2)$
2. For each topic $k \in \{1, \dots, K\}$
   (a) Draw word distribution $\boldsymbol{\phi_k} \sim \text{Dir}(\beta)$
   (b) Draw topic regression parameter $\eta_k \sim \mathcal{N}(0, \nu^2)$
3. For each word $v \in \{1, \dots, V\}$
   (a) Draw lexical regression parameter $\tau_v \sim \mathcal{N}(0, \nu^2)$
4. For each document $d \in \{1, \dots, D\}$
   (a) Draw topic distribution $\boldsymbol{\theta_d} \sim \text{Dir}(\alpha)$
   (b) For each token $t_{d,n}$ in document d
      i. Draw topic assignment $z_{d,n} \sim \text{Mult}(\boldsymbol{\theta_d})$
      ii. Draw word $w_{d,n} \sim \text{Mult}(\boldsymbol{\phi_{z_{d,n}}})$
5. For each explicit link (d, d')
   (a) Draw link weight $B_{d,d'} \sim \Psi(\cdot \mid y_d, y_{d'}, \boldsymbol{\Omega}, \boldsymbol{z_d}, \boldsymbol{z_{d'}}, \boldsymbol{w_d}, \boldsymbol{w_{d'}}, \boldsymbol{\eta}, \boldsymbol{\tau}, \boldsymbol{\rho})$

Links are generated by a link probability function Ψ which takes the regression value $R_{d,d'}$ of documents d and d' as an argument. Assuming documents d and d' belong to blocks l and l' respectively, $R_{d,d'}$ is

$$R_{d,d'} = \boldsymbol{\eta}^{\mathrm{T}}(\overline{\boldsymbol{z}}_d \circ \overline{\boldsymbol{z}}_{d'}) + \boldsymbol{\tau}^{\mathrm{T}}(\overline{\boldsymbol{w}}_d \circ \overline{\boldsymbol{w}}_{d'}) + \rho_{l,l'}\Omega_{l,l'}, \quad (1)$$

where $\overline{\boldsymbol{z}}_d$ is a K-length vector with each element $\overline{z}_{d,k} = \frac{1}{N_d}\sum_n \mathbb{1}(z_{d,n} = k)$; $\overline{\boldsymbol{w}}_d$ is a V-length vector with each element $\overline{w}_{d,v} = \frac{1}{N_d}\sum_n \mathbb{1}(w_{d,n} = v)$; $\circ$ denotes the Hadamard (element-wise) product;[1] $\boldsymbol{\eta}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$ are the weight vectors and matrix for topic-based, lexical-based and rate-based predictions, respectively.

A common choice of Ψ is a sigmoid (Chang and Blei, 2010). However, we instead use hinge loss so that VF-RTM can use the max-margin principle, making more effective use of side information when inferring topic assignments (Zhu et al., 2012). Using hinge loss, the probability that documents d and d' are linked is

$$\Pr(B_{d,d'}) = \exp(-2\max(0, \zeta_{d,d'})), \quad (2)$$

where $\zeta_{d,d'} = 1 - B_{d,d'}R_{d,d'}$. Positive and negative link weights are denoted by 1 and -1, respectively, in contrast to sigmoid loss.

3.3 Aggregated Model

Finally, we put all the pieces together and propose LBH-RTM: RTM with lexical weights (L), block priors (B), and hinge loss (H). Its graphical model is given in Figure 6.

1. For each pair of blocks $(l, l') \in \{1, \dots, L\}^2$
   (a) Draw inter-block link rate $\Omega_{l,l'} \sim \text{Gamma}(a, b)$
   (b) Draw block regression parameter $\rho_{l,l'} \sim \mathcal{N}(0, \nu^2)$
2. Draw block distribution $\boldsymbol{\mu} \sim \text{Dir}(\gamma)$
3. For each block $l \in \{1, \dots, L\}$
   (a) Draw topic distribution $\boldsymbol{\pi_l} \sim \text{Dir}(\alpha')$
4. For each topic $k \in \{1, \dots, K\}$
   (a) Draw word distribution $\boldsymbol{\phi_k} \sim \text{Dir}(\beta)$
   (b) Draw topic regression parameter $\eta_k \sim \mathcal{N}(0, \nu^2)$
5. For each word $v \in \{1, \dots, V\}$
   (a) Draw lexical regression parameter $\tau_v \sim \mathcal{N}(0, \nu^2)$
6. For each document $d \in \{1, \dots, D\}$
   (a) Draw block assignment $y_d \sim \text{Mult}(\boldsymbol{\mu})$
   (b) Draw topic distribution $\boldsymbol{\theta_d} \sim \text{Dir}(\alpha \boldsymbol{\pi}_{\boldsymbol{y_d}})$
   (c) For each token $t_{d,n}$ in document d
      i. Draw topic assignment $z_{d,n} \sim \text{Mult}(\boldsymbol{\theta_d})$
      ii. Draw word $w_{d,n} \sim \text{Mult}(\boldsymbol{\phi}_{\boldsymbol{z_{d,n}}})$
7. For each link $(d, d') \in \{1, \dots, D\}^2$
   (a) Draw link weight $A_{d,d'} \sim \text{Poisson}(\Omega_{y_d, y_{d'}})$
8. For each explicit link (d, d')
   (a) Draw link weight $B_{d,d'} \sim \Psi(\cdot \mid y_d, y_{d'}, \boldsymbol{\Omega}, \boldsymbol{z_d}, \boldsymbol{z_{d'}}, \boldsymbol{w_d}, \boldsymbol{w_{d'}}, \boldsymbol{\eta}, \boldsymbol{\tau}, \boldsymbol{\rho})$

$\boldsymbol{A}$ and $\boldsymbol{B}$ are assumed independent in the model, but they can be derived from the same set of links in practice.

[1] As Chang and Blei (2010) point out, the Hadamard product is able to capture similarity between hidden topic representations of two documents.

Algorithm 1 Sampling Process

```
1:  Set λ = 1 and initialize topic assignments
2:  for m = 1 to M do
3:      Optimize η, τ, and ρ using L-BFGS
4:      for d = 1 to D do
5:          Draw block assignment y_d
6:          for each token n do
7:              Draw a topic assignment z_{d,n}
8:          end for
9:          for each explicit link (d, d') do
10:             Draw λ^{-1}_{d,d'} (and then λ_{d,d'})
11:         end for
12:     end for
13: end for
```

Link set $\boldsymbol{A}$ is primarily used to find blocks, so it treats all links *deterministically*. In other words, the links observed in the input are considered explicit positive links, while the unobserved links are considered explicit negative links, in contrast to the implicit links in $\boldsymbol{B}$.

In terms of link set $\boldsymbol{B}$, while it adopts all explicit positive links from the input, it *does not deny* the existence of unobserved links, or implicit negative links. Thus $\boldsymbol{B}$ consists of only explicit positive links. However, to avoid overfitting, we sample some implicit links and add them to $\boldsymbol{B}$ as explicit negative links.

4 Posterior Inference

Posterior inference (Algorithm 1) consists of the sampling of topic and block assignments and the optimization of weight vectors and matrix.[2] We add an auxiliary variable $\boldsymbol{\lambda}$ for hinge loss (see Section 4.2), so the updating of $\boldsymbol{\lambda}$ is not necessary when using sigmoid loss.

The sampling procedure is an iterative process after initialization (Line 1). In each iteration, we first optimize the weight vectors and matrix (Line 3) before updating documents' block assignments (Line 5) and topic assignments (Line 7). When using hinge loss, the auxiliary variable $\boldsymbol{\lambda}$ for every explicit link needs to be updated (Line 10).

4.1 Sampling Block Assignments

Block assignment sampling is done by Gibbs sampling, using the block assignments and links in $\boldsymbol{A}$

[2] More details about sampling procedures and equations in this section (including the sampling and optimization equations using sigmoid loss) are available in the supplementary material.

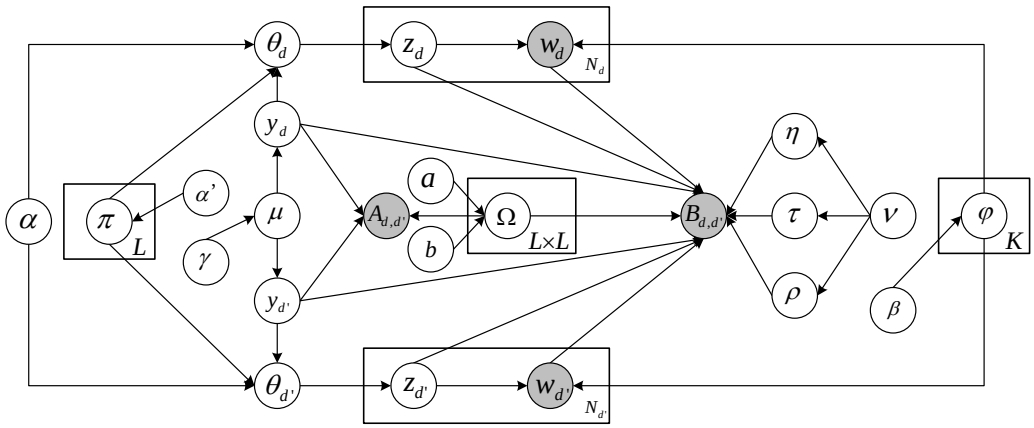


Figure 6: The graphical model of LBH-RTM for two documents, in which a weighted stochastic block model is embedded (γ, μ, y, a, b, $\boldsymbol{\Omega}$, and $\boldsymbol{A}$). Each document's topic distribution has an informative prior $\boldsymbol{\pi}$. The model predicts links between documents ($\boldsymbol{B}$) based on topics ($\boldsymbol{z}$), words ($\boldsymbol{w}$), and inter-block link rates ($\boldsymbol{\Omega}$), using a max-margin objective.

excluding document d and its related links.[3] The probability that d is assigned to block l is

$$\Pr(y_d = l \mid \boldsymbol{A}_{-d}, \boldsymbol{y}_{-d}, a, b, \gamma) \propto \left(N_l^{-d} + \gamma\right) \times \prod_{l'} \frac{\left(S_e^{-d}(l,l') + b\right)^{S_w^{-d}(l,l')+a}}{\left(S_e^{-d}(l,l') + b + S_e(d,l')\right)^{S_w^{-d}(l,l')+a+S_w(d,l')}} \prod_{i=0}^{S_w(d,l')-1} \left(S_w^{-d}(l,l') + a + i\right), \quad (3)$$

where N_l is the number of documents assigned to block l; $^{-d}$ denotes that the count excludes document d; $S_w(d,l)$ and $S_w(l,l')$ are the sums of link weights from document d to block l and from block l to block l', respectively:

$$S_w(d,l) = \sum_{d':y_{d'}=l} A_{d,d'} \quad (4)$$

$$S_w(l,l') = \sum_{d:y_d=l} S_w(d,l'). \quad (5)$$

$S_e(d,l)$ is the number of possible links from document d to l (i.e., assuming document d connects to every document in block l), which equals N_l. The number of possible links from block l to l' is $S_e(l,l')$ (i.e., assuming every document in block l connects to every document in block l'):

$$S_e(l,l') = \begin{cases} N_l \times N_{l'} & l \neq l' \\ \frac{1}{2} N_l(N_l - 1) & l = l'. \end{cases} \quad (6)$$

If we rearrange the terms of Equation 3 and put the terms which have $S_w(d,l')$ together, the value of $S_w(d,l')$ increases (i.e., document d is more densely connected with documents in block l'), the probability of assigning d to block l decreases exponentially. Thus if d is more densely connected with block l and sparsely connected with other blocks, it is more likely to be assigned to block l.

4.2 Sampling Topic Assignments

Following Polson and Scott (2011), by introducing an auxiliary variable $\lambda_{d,d'}$, the conditional probability of assigning $t_{d,n}$, the n-th token in document d, to topic k is

$$\Pr(z_{d,n} = k \mid \boldsymbol{z}_{-d,n}, \boldsymbol{w}_{-d,n}, w_{d,n} = v, y_d = l) \propto \left(N_{d,k}^{-d,n} + \alpha \pi_{l,k}^{-d,n}\right) \frac{N_{k,v}^{-d,n} + \beta}{N_{k,\cdot}^{-d,n} + V\beta} \prod_{d'} \exp\left(-\frac{(\zeta_{d,d'} + \lambda_{d,d'})^2}{2\lambda_{d,d'}}\right), \quad (7)$$

where $N_{d,k}$ is the number of tokens in document d that are assigned to topic k; $N_{k,v}$ denotes the count of word v assigned to topic k; Marginal counts are denoted by $\cdot$; $^{-d,n}$ denotes that the count excludes $t_{d,n}$; d' denotes all documents that have explicit links with document d. The block topic prior $\pi_{l,k}^{-d,n}$ is estimated based on the maximal path assumption (Cowans, 2006; Wallach, 2008):

$$\pi_{l,k}^{-d,n} = \frac{\sum_{d':y_{d'}=l} N_{d',k}^{-d,n} + \alpha'}{\sum_{d':y_{d'}=l} N_{d',\cdot}^{-d,n} + K\alpha'}. \quad (8)$$

the link prediction argument $\zeta_{d,d'}$ is

$$\zeta_{d,d'} = 1 - B_{d,d'}\left(\frac{\eta_k}{N_{d,\cdot}} \frac{N_{d',k}}{N_{d',\cdot}} + R_{d,d'}^{-d,n}\right). \quad (9)$$

[3] These equations deal with undirected edges, but they can be adapted for directed edges. See supplementary material.

where

$$R_{d,d'}^{-d,n} = \sum_{k=1}^{K} \eta_k \frac{N_{d,k}^{-d,n}}{N_{d,\cdot}} \frac{N_{d',k}}{N_{d',\cdot}} + \sum_{v=1}^{V} \tau_v \frac{N_{d,v}}{N_{d,\cdot}} \frac{N_{d',v}}{N_{d',\cdot}} + \rho_{y_d, y_{d'}} \Omega_{y_d, y_{d'}}. \quad (10)$$

Looking at the first term of Equation 7, the probability of assigning $t_{d,n}$ to topic k depends not only on its own topic distribution, but also the topic distribution of the block it belongs to. The links also matter: Equation 9 gives us the intuition that a topic which could increase the likelihood of links is more likely to be selected, which forms an interaction between topics and the link graph—the links are guiding the topic sampling while updating topic assignments is maximizing the likelihood of the link graph.

4.3 Parameter Optimization

While topic assignments are updated iteratively, the weight vectors and matrix $\boldsymbol{\eta}$, $\boldsymbol{\tau}$, and $\boldsymbol{\rho}$ are optimized in each global iteration over the whole corpus using L-BFGS (Liu and Nocedal, 1989). It takes the likelihood of generating $\boldsymbol{B}$ using $\boldsymbol{\eta}$, $\boldsymbol{\tau}$, $\boldsymbol{\rho}$, and current topic and block assignments as the objective function, and optimizes it using the partial derivatives with respect to every weight vector/matrix element.

The log likelihood of $\boldsymbol{B}$ using hinge loss is

$$\mathcal{L}(\boldsymbol{B}) \propto -\sum_{d,d'} \frac{R_{d,d'}^2 - 2(1+\lambda_{d,d'}) B_{d,d'} R_{d,d'}}{2\lambda_{d,d'}} - \sum_{k=1}^{K} \frac{\eta_k^2}{2\nu^2} - \sum_{v=1}^{V} \frac{\tau_v^2}{2\nu^2} - \sum_{l=1}^{L} \sum_{l'=1}^{L} \frac{\rho_{l,l'}^2}{2\nu^2}. \quad (11)$$

We also need to update the auxiliary variable $\lambda_{d,d'}$. Since the likelihood of $\lambda_{d,d'}$ follows a generalized inverse Gaussian distribution $\mathrm{GIG}\left(\lambda_{d,d'}; \frac{1}{2}, 1, \zeta_{d,d'}^2\right)$, we sample its reciprocal $\lambda_{d,d'}^{-1}$ from an inverse Gaussian distribution as

$$\Pr\left(\lambda_{d,d'}^{-1} \mid \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\eta}, \boldsymbol{\tau}, \boldsymbol{\rho}\right) = \mathrm{IG}\left(\lambda_{d,d'}^{-1}; \frac{1}{|\zeta_{d,d'}|}, 1\right). \quad (12)$$

5 Experimental Results

We evaluate using the two datasets. The first one is CORA dataset (McCallum et al., 2000). After removing stopwords and words that appear in fewer than ten documents, as well as documents with no words or links, our vocabulary has 1,240 unique words. The corpus has 2,362 computer science paper abstracts with 4,231 citation links.

Model	PLR	
	CORA	WEBKB
RTM (Chang and Blei, 2010)	419.33	141.65
LCH-RTM (Yang et al., 2015)	459.55	150.32
BS-RTM	391.88	127.25
LBS-RTM	383.25	125.41
LBH-RTM	**360.38**	**111.79**

Table 1: Predictive Link Rank Results

The second dataset is WEBKB. It is already preprocessed and has 1,703 unique words in vocabulary. The corpus has 877 web pages with 1,608 hyperlinks.

We treat all links as undirected. Both datasets are split into 5 folds, each further split into a development and test set with approximately the same size when used for evaluation.

5.1 Link Prediction Results

In this section, we evaluate LBH-RTM and its variations on link prediction tasks using *predictive link rank* (PLR). A document's PLR is the average rank of the documents to which it has explicit positive links, among all documents, so lower PLR is better.

Following the experiment setup in Chang and Blei (2010), we train the models on the training set and predict citation links within held-out documents as well as from held-out documents to training documents. We tune two important parameters—α and negative edge ratio (the ratio of the number of sampled negative links to the number of explicit positive links)—on the development set and apply the trained model which performs the best on the development set to the test set.[4] The cross validation results are given in Table 1, where models are differently equipped with lexical weights (L), WSBM prior (B), SCC prior (C), hinge loss (H), and sigmoid loss (S).[5] Link prediction generally improves with incremental application of prior knowledge and more sophisticated learning techniques.

The embedded WSBM brings around 6.5% and 10.2% improvement over RTM in PLR on the

[4] We also tune the number of blocks for embedded WSBM and set it to 35 (CORA) and 20 (WEBKB). The block topic priors are not applied on unseen documents, since we don't have available links.

[5] The values of RTM are different from the result reported by Chang and Blei (2010), because we re-preprocessed the CORA dataset and used different parameters.

CORA and WEBKB datasets, respectively. This indicates that the blocks identified by WSBM are reasonable and consistent with reality. The lexical weights also help link prediction (LBS-RTM), though less for BS-RTM. This is understandable since word distributions are much sparser and do not make as significant a contribution as topic distributions. Finally, hinge loss improves PLR substantially (LBH-RTM), about 14.1% and 21.1% improvement over RTM on the CORA and WEBKB datasets respectively, demonstrating the effectiveness of max-margin learning.

The only difference between LCH-RTM and LBH-RTM is the block detection algorithm. However, their link prediction performance is poles apart—LCH-RTM even fails to outperform RTM. This implies that the quality of blocks identified by SCC is not as good as WSBM, which we also illustrate in Section 5.4.

5.2 Illustrative Example

We illustrate our model's behavior qualitatively by looking at two abstracts, Koplon and Sontag (1997) and Albertini and Sontag (1992) from the CORA dataset, designated K and A for short.

Paper K studies the application of Fourier-type activation functions in fully recurrent neural networks. Paper A shows that if two neural networks have equal behaviors as "black boxes", they must have the same number of neurons and the same weights (except sign reversals).

From the titles and abstracts, we can easily find that both of them are about *neural networks* (NN). They both contain words like *neural*, *neuron*, *network*, *recurrent*, *activation*, and *nonlinear*, which corresponds to the topic with words *neural*, *network*, *train*, *learn*, *function*, *recurrent*, etc. There is a citation between K and A. The ranking of this link improves as the model gets more sophisticated (Table 2), except LCH-RTM, which is consistent with our PLR results.

In Figure 7, we also show the proportions of topics that dominate the two documents according to the various models. There are multiple topics dominating K and A according to RTM (Figure 7(a)). As the model gets more sophisticated, the NN topic proportion gets higher. Finally, only the NN topic dominates the two documents when LBH-RTM is applied (Figure 7(e)).

LCH-RTM gives the highest proportion to the NN topic (Figure 7(b)). However, the NN topic splits into two topics and the proportions are not assigned to the same topic, which greatly brings down the link prediction performance. The splitting of the NN topic also happens in other models (Figure 7(a) and 7(d)), but they assign proportions to the same topic(s). Further comparing with LBH-RTM, the blocks detected by SCC are not improving the modeling of topics and links—some documents that should be in two different blocks are assigned to the same one, as we will show in Section 5.4.

Model	Rank of the Link
RTM	1,265
LCH-RTM	1,385
BS-RTM	635
LBS-RTM	132
LBH-RTM	**106**

Table 2: PLR of the citation link between example documents K and A (described in Section 5.2)

Model	FET		LLR	
	CORA	WEBKB	CORA	WEBKB
RTM	0.1330	0.1312	3.001	6.055
LCH-RTM	0.1418	0.1678	3.071	6.577
BS-RTM	0.1415	0.1950	3.033	6.418
LBS-RTM	0.1342	0.1963	2.984	6.212
LBH-RTM	**0.1453**	**0.2628**	**3.105**	**6.669**

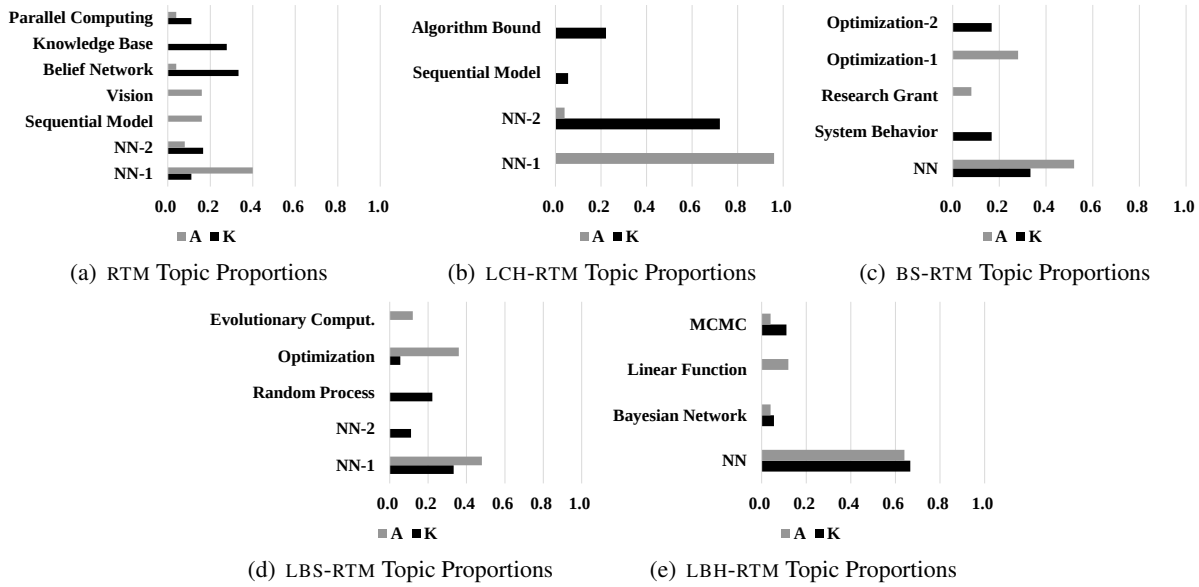
Table 3: Average Association Scores of Topics

5.3 Topic Quality Results

We use an automatic coherence detection method (Lau et al., 2014) to evaluate topic quality. Specifically, for each topic, we pick out the top n words and compute the average association score of each pair of words, based on the held-out documents in development and test sets.

We choose $n = 25$ and use *Fisher's exact test* (Upton, 1992, FET) and *log likelihood ratio* (Moore, 2004, LLR) as the association measures (Table 3). The main advantage of these measures is that they are robust even when the reference corpus is not large.

Coherence improves with WSBM and max-margin learning, but drops a little when adding lexical weights except the FET score on the WEBKB dataset, because lexical weights are intended to improve link prediction performance, not topic quality. Topic quality of LBH-RTM is also better than that of LCH-RTM, suggesting that WSBM benefits topic quality more than SCC.



(a) RTM Topic Proportions (b) LCH-RTM Topic Proportions (c) BS-RTM Topic Proportions

(d) LBS-RTM Topic Proportions (e) LBH-RTM Topic Proportions

Figure 7: Topic proportions given by various models on our two illustrative documents (K and A, described in described in Section 5.2). As the model gets more sophisticated, the NN topic proportion gets higher and finally dominates the two documents when LBH-RTM is applied. Though LCH-RTM gives the highest proportion to the NN topic, it splits the NN topic into two and does not assign the proportions to the same one.

Block	1	2
#Nodes	42	84
#Links in the Block	55	142
#Links across Blocks	2	

Table 4: Statistics of Blocks 1 (learning theory) and 2 (Bayes nets), which are merged in SCC.

5.4 Block Analysis

In this section, we illustrate the effectiveness of the embedded WSBM over SCC.[6] As we have argued, WSBM is able to separate two internally densely-connected blocks even if there are few links connecting them, while SCC tends to merge them in this case. As an example, we focus on two blocks in the CORA dataset identified by WSBM, designated Blocks 1 and 2. Some statistics are given in Table 4. The two blocks are very sparsely connected, but comparatively quite densely connected inside either block. The two blocks' topic distributions also reveal their differences: abstracts in Block 1 mainly focus on learning theory (*learn*, *algorithm*, *bound*, *result*, etc.) and MCMC (*markov*, *chain*, *distribution*, *converge*, etc.). Abstracts in Block 2, however, have higher weights on Bayesian networks (*network*, *model*, *learn*, *bayesian*, etc.) and Bayesian estimation (*estimate*, *bayesian*, *parameter*, *analysis*, etc.), which differs from Block 1's emphasis. Because of the two inter-block links, SCC merges the two blocks into one, which makes the block topic distribution unclear and misleads the sampler. WSBM, on the other hand, keeps the two blocks separate, which generates a high-quality prior for the sampler.

[6] We omit the comparison of WSBM with other models, because this has been done by Aicher et al. (2014). In addition, WSBM is a probabilistic method while SCC is deterministic. They are not comparable quantitatively, so we compare them qualitatively.

6 Related Work

Topic models are widely used in information retrieval (Wei and Croft, 2006), word sense disambiguation (Boyd-Graber et al., 2007), dialogue segmentation (Purver et al., 2006), and collaborative filtering (Marlin, 2003).

Topic models can be extended in either *upstream* or *downstream* way. *Upstream* models generate topics conditioned on supervisory information (Daumé III, 2009; Mimno and McCallum, 2012; Li and Perona, 2005). *Downstream* models, on the contrary, generates topics and supervisory data simultaneously, which turns unsupervised topic models to (semi-)supervised ones. Supervisory data, like labels of documents and links between documents, can be generated from either a maximum likelihood estimation approach (McAuliffe and Blei, 2008; Chang and

Blei, 2010; Boyd-Graber and Resnik, 2010) or a maximum entropy discrimination approach (Zhu et al., 2012; Yang et al., 2015).

In block detection literature, stochastic block model (Holland et al., 1983; Wang and Wong, 1987, SBM) is one of the most basic generative models dealing with binary-weighted edges. SBM assumes that each node belongs to only one block and each link exists with a probability that depends on the block assignments of its connecting nodes. It has been generalized for degree-correction (Karrer and Newman, 2011), bipartite structure (Larremore et al., 2014), and categorial values (Guimerà and Sales-Pardo, 2013), as well as nonnegative integer-weight network (Aicher et al., 2014, WSBM).

Our model combines both topic model and block detection in a unified framework. It takes text, links, and the interaction between text and links into account simultaneously, contrast to the methods that only consider graph structure (Kim and Leskovec, 2012; Liben-Nowell and Kleinberg, 2007) or separate text and links (Chaturvedi et al., 2012).

7 Conclusions and Future Work

We introduce LBH-RTM, a discriminative topic model that jointly models topics and document links, detecting blocks in the document network probabilistically by embedding the weighted stochastic block model, rather via connected-components as in previous models. A separate Dirichlet prior for each block captures its topic preferences, serving as an informed prior when inferring documents' topic distributions. Max-margin learning learns to predict links from documents' topic and word distributions and block assignments.

Our model better captures the connections and content of paper abstracts, as measured by predictive link rank and topic quality. LBH-RTM yields topics with better coherence, though not all techniques contribute to the improvement. We support our quantitative results with qualitative analysis looking at a pair of example documents and at a pair of blocks, highlighting the robustness of embedded WSBM over blocks defined as SCC.

As next steps, we plan to explore model variations to support a wider range of use cases. For example, although we have presented a version of the model defined using undirected binary weight edges in the experiment, it would be straightforward to adapt to model both directed/undirected and binary/nonnegative real weight edges. We are also interested in modeling changing topics and vocabularies (Blei and Lafferty, 2006; Zhai and Boyd-Graber, 2013). In the spirit of treating links probabilistically, we plan to explore application of the model in suggesting links that do not exist but should, for example in discovering missed citations, marking social dynamics (Nguyen et al., 2014), and identifying topically related content in multilingual networks of documents (Hu et al., 2014).

Acknowledgment

This research has been supported in part, under subcontract to Raytheon BBN Technologies, by DARPA award HR0011-15-C-0113. Boyd-Graber is also supported by NSF grants IIS/1320538, IIS/1409287, and NCSE/1422492. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Christopher Aicher, Abigail Z. Jacobs, and Aaron Clauset. 2014. Learning latent block structure in weighted networks. *Journal of Complex Networks*.

Francesca Albertini and Eduardo D. Sontag. 1992. For neural networks, function determines form. In *Proceedings of IEEE Conference on Decision and Control*.

David M. Blei and John D. Lafferty. 2006. Dynamic topic models. In *Proceedings of the International Conference of Machine Learning*.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*.

Jordan Boyd-Graber and Philip Resnik. 2010. Holistic sentiment analysis across languages: Multilingual supervised latent Dirichlet allocation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jordan Boyd-Graber, David M. Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of Empirical Methods in Natural Language Processing*.

Jonathan Chang and David M. Blei. 2010. Hierarchical relational models for document networks. *The Annals of Applied Statistics*.

Snigdha Chaturvedi, Hal Daumé III, Taesun Moon, and Shashank Srivastava. 2012. A topical graph kernel for link prediction in labeled graphs. In *Proceedings of the International Conference of Machine Learning*.

Philip J. Cowans. 2006. *Probabilistic Document Modelling*. Ph.D. thesis, University of Cambridge.

Hal Daumé III. 2009. Markov random topic fields. In *Proceedings of the Association for Computational Linguistics*.

Roger Guimerà and Marta Sales-Pardo. 2013. A network inference method for large-scale unsupervised identification of novel drug-drug interactions. *PLoS Computational Biology*.

Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social Networks*.

Yuening Hu, Ke Zhai, Vlad Eidelman, and Jordan Boyd-Graber. 2014. Polylingual tree-based topic models for translation domain adaptation. In *Proceedings of the Association for Computational Linguistics*.

Brian Karrer and Mark EJ Newman. 2011. Stochastic blockmodels and community structure in networks. *Physical Review E*.

Myunghwan Kim and Jure Leskovec. 2012. Latent multi-group membership graph model. In *Proceedings of the International Conference of Machine Learning*.

Renée Koplon and Eduardo D. Sontag. 1997. Using Fourier-neural recurrent networks to fit sequential input/output data. *Neurocomputing*.

Daniel B. Larremore, Aaron Clauset, and Abigail Z. Jacobs. 2014. Efficiently inferring community structure in bipartite networks. *Physical Review E*.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the Association for Computational Linguistics*.

Fei-Fei Li and Pietro Perona. 2005. A Bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition*.

David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*.

Dong C. Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*.

Benjamin Marlin. 2003. Modeling user rating profiles for collaborative filtering. In *Proceedings of Advances in Neural Information Processing Systems*.

Jon D. McAuliffe and David M. Blei. 2008. Supervised topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of Internet portals with machine learning. *Information Retrieval*.

David Mimno and Andrew McCallum. 2012. Topic models conditioned on arbitrary features with Dirichlet-multinomial regression. In *Proceedings of Uncertainty in Artificial Intelligence*.

Robert Moore. 2004. On log-likelihood-ratios and the significance of rare events. In *Proceedings of Empirical Methods in Natural Language Processing*.

Viet-An Nguyen, Jordan Boyd-Graber, and Philip Resnik. 2013. Lexical and hierarchical topic regression. In *Proceedings of Advances in Neural Information Processing Systems*.

Viet-An Nguyen, Jordan Boyd-Graber, Philip Resnik, Deborah Cai, Jennifer Midberry, and Yuanxin Wang. 2014. Modeling topic control to detect influence in conversations using nonparametric topic models. *Machine Learning*.

Nicholas G. Polson and Steven L. Scott. 2011. Data augmentation for support vector machines. *Bayesian Analysis*.

Matthew Purver, Thomas L. Griffiths, Konrad P. Körding, and Joshua B. Tenenbaum. 2006. Unsupervised topic modelling for multi-party spoken discourse. In *Proceedings of the Association for Computational Linguistics*.

Graham JG Upton. 1992. Fisher's exact test. *Journal of the Royal Statistical Society*.

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

Yuchung J. Wang and George Y. Wong. 1987. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*.

Xing Wei and W. Bruce Croft. 2006. LDA-based document models for ad-hoc retrieval. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Weiwei Yang, Jordan Boyd-Graber, and Philip Resnik. 2015. Birds of a feather linked together: A discriminative topic model using link-based priors. In *Proceedings of Empirical Methods in Natural Language Processing*.

Ke Zhai and Jordan Boyd-Graber. 2013. Online latent Dirichlet allocation with infinite vocabulary. In *Proceedings of the International Conference of Machine Learning*.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2012. MedLDA: Maximum margin supervised topic models. *Journal of Machine Learning Research*.

Jun Zhu, Ning Chen, Hugh Perkins, and Bo Zhang. 2014. Gibbs max-margin topic models with data augmentation. *Journal of Machine Learning Research*.

AraSenTi: Large-Scale Twitter-Specific Arabic Sentiment Lexicons

Nora Al-Twairesh[1,2], Hend Al-Khalifa[2], AbdulMalik Al-Salman[1]

Computer Science Department[1], Information Technology Department[2]
College of Computer and Information Sciences
King Saud University
{twairesh,hendk,salman@ksu.edu.sa}

Abstract

Sentiment Analysis (SA) is an active research area nowadays due to the tremendous interest in aggregating and evaluating opinions being disseminated by users on the Web. SA of English has been thoroughly researched; however research on SA of Arabic has just flourished. Twitter is considered a powerful tool for disseminating information and a rich resource for opinionated text containing views on many different topics. In this paper we attempt to bridge a gap in Arabic SA of Twitter which is the lack of sentiment lexicons that are tailored for the informal language of Twitter. We generate two lexicons extracted from a large dataset of tweets using two approaches and evaluate their use in a simple lexicon based method. The evaluation is performed on internal and external datasets. The performance of these automatically generated lexicons was very promising, albeit the simple method used for classification. The best F-score obtained was 89.58% on the internal dataset and 63.1-64.7% on the external datasets.

1 Introduction

The past decade has witnessed the proliferation of social media websites which has led to the production of vast amounts of unstructured text on the Web. This text can be characterized as objective, i.e. containing facts, or subjective i.e. containing opinions and sentiments about entities. Sentiment Analysis (SA) is the research field that is concerned with identifying opinions in text and classifying them as positive, negative or neutral. SA of English has been thoroughly researched; however research on SA of Arabic has just flourished.

Arabic is ranked fourth among languages on the web although it is the fastest growing language on the web among other languages (Internet World Stats, 2015). Arabic is a morphologically rich language where one lemma can have hundreds of surface forms; this complicates the tasks of SA. Moreover, the Arabic language has many variants. The formal language is called Modern Standard Arabic (MSA) and the spoken language differs in different Arabic countries producing numerous Arabic dialects sometimes called informal Arabic or colloquial Arabic. The language used in social media is known to be highly dialectal (Darwish and Magdy, 2014). Dialects differ from MSA phonologically, morphologically and syntactically and they do not have standard orthographies (Habash, 2010). Consequently, resources built for MSA cannot be adapted to dialects very well.

The informal language used in social media and in Twitter in particular makes the SA of tweets a challenging task. The language on social media is known to contain slang, nonstandard spellings and evolves by time. As such sentiment lexicons that are built from standard dictionaries cannot adequately capture the informal language in social media text. Therefore, in this paper we propose to generate Arabic sentiment lexicons that are tweet-specific i.e. generated from tweets. We present two approaches to generating Arabic sentiment lexicons from a large dataset of 2.2 million tweets. The lexicons are evaluated on three datasets, one internal dataset extracted from the larger dataset of tweets and two external datasets from the literature on Arabic SA. Moreover, the lexicons are compared to an external Arabic lexicon generated also from tweets. A simple lexicon-based method is used to evaluate the lexicons.

This paper is organized as follows: Section 2 reviews the related work on sentiment lexicon generation. Section 3 describes the details of the datasets used to generate the lexicons and how they were collected. Section 4 presents the approaches used to generate the lexicons. Section 5 details the experimental setup while Section 6 presents and analyzes the results. Finally, we

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 697–705,
Berlin, Germany, August 7-12, 2016.

conclude the paper and present potential future work in Section 7.

2 Related Work

Words that convey positive or negative sentiment are fundamental for sentiment analysis. Compiling a list of these words is what is referred to as **sentiment lexicon generation**. There are three approaches to generate a sentiment lexicon (Liu, 2012): *manual approach, dictionary-based approach, and corpus-based approach*. The **manual approach** is usually not done alone since it is time consuming and labor intensive. It is used however, in conjunction with automated approaches to check the correction of the resulting lexicons from these approaches. In this section we review popular English and Arabic sentiment lexicons in the literature.

2.1 English Sentiment Lexicons

In the **dictionary based approach** as the name implies a dictionary is used by utilizing the synonym and antonym lists that are associated with dictionary words. The technique starts with a small set of sentiment words as seeds with known positive or negative orientations. The seed words are looked up in the dictionary then their synonyms and antonyms are added to the seed set and a new iteration starts. The process ends when no new words are found. A manual inspection is usually done after the process ends to correct errors. A majority of studies under this approach used the WordNet with different approaches for expanding the list such as distance-based measures (Kamps, 2004; Williams and Anand, 2009) and graph-based methods (Blair-Goldensohn et al., 2008; Rao and Ravichandran, 2009). Pioneering work in this approach is the construction of SentiWordNet by (Esuli and Sebastiani, 2005). Initially, they started with a set of positive seeds and a set of negative seeds then expanded the sets using the synonym and antonym relations in WordNet. This formed a training set which they used in a supervised learning classifier and applied it to all the glosses in WordNet, the process is run iteratively. Then in a following attempt (Esuli and Sebastiani, 2006), a committee of classifiers based on the previous method were used to build SentiWordNet which contains terms that are associated with three scores for objectivity, positivity and negativity, where the sum of the scores is 1. The latest version is SentiWordNet 3.0 (Baccianella et al., 2010).

As for **corpus-based approaches**, the words of the lexicon are extracted from the corpus using a seed list of known sentiment words and different approaches to find words of similar or opposite polarity. One of the earliest work in this approach was that of (Hatzivassiloglou and McKeown, 1997), where they utilized connectives e.g. and, but, etc. between adjectives in a corpus to learn new sentiment words not in the seed list. Turney, (2002); Turney and Littman, (2002) used the once popular AltaVista search engine to find the sentiment of a certain word through calculating the association strength between the word and a set of positive words minus the association strength between the word and a set of negative words. The association strength was measured using Pointwise-Mutual Information (PMI). The result is the sentiment score of the word, if it is positive this means the word is strongly associated with positive polarity and as such its polarity will be positive and if it is negative the word's polarity will be negative. The magnitude indicates the sentiment intensity of the word. We used PMI to generate one of the lexicons in this paper.

After the emergence of sentiment analysis as an evolving research field, several lexicons were constructed according to the approaches mentioned above. In the Bing Liu's lexicon (Hu and Liu, 2004), which falls under the dictionary-based method, the WordNet was exploited to infer the semantic orientation of adjectives extracted from customer reviews. The lexicon only provides the prior polarity of words: positive or negative, the sentiment intensity of the words was not calculated. Another popular sentiment lexicon is the MPQA subjectivity lexicon (Wilson et al., 2005) which was constructed by manually annotating the subjective expressions in the MPQA corpus. The words were annotated with four tags: positive, negative, both and neutral then further classified as strong or weak to denote intensity. We use these two lexicons in the generation of the other lexicon in this paper.

With the proliferation of social media websites, the need for lexicons that can capture the peculiarities of social medial language emerges. As such, many solutions for sentiment analysis of social media and Twitter in particular initiate by developing sentiment lexicons that are extracted from Twitter (Tang et al., 2014; Kiritchenko et al., 2014).

2.2 Arabic Sentiment Lexicons

Generating sentiment lexicons for Arabic has gained the interest of the research community lately. Consequently, we found several efforts for generating these lexicons. A recent effort to build a large scale multi-genre multi dialect Arabic sentiment lexicon was proposed by (Abdul-Mageed and Diab, 2014). However, it covers only two dialects: Egyptian and Levantine and is not yet fully applied to SSA tasks. Badaro et al., (2014) constructed ArSenL, a large scale Arabic sentiment lexicon. They relied on four resources to create ArSenL: English WordNet (EWN), Arabic WordNet (AWN), English SentiWordNet (ESWN), and SAMA (Standard Arabic Morphological Analyzer). Two approaches were followed producing two different lexicons: the first approach used AWN, by mapping AWN entries into ESWN using existing offsets thus producing ArSenL-AWN. The second approach utilizes SAMA's English glosses by finding the highest overlapping synsets between these glosses and ESWN thus producing ArSenL-Eng. Hence ArSenL is the union of these two lexicons. Although this lexicon can be considered as the largest Arabic sentiment lexicon developed to date, it is unfortunate that it only has MSA entries and no dialect words and is not developed from a social media context which could affect the accuracy when applied on social media text.

Following the example of ArSenL, the lexicon SLSA (Sentiment Lexicon for Standard Arabic) (Eskander and Rambow, 2015) was constructed by linking the lexicon of an Arabic morphological analyzer Aramorph with SentiWordNet. Although the approach is very similar to ArSenL, since both use SentiWordNet to obtain the scores of words, the linking algorithm used to link the glosses in Aramorph with those in SentiWordNet is different. SLSA starts by linking every entry in Aramorph with SentiWordNet if the one-gloss word and POS match. Intrinsic and extrinsic evaluations were performed by comparing SLSA and ArSenL which demonstrated the superiority of SLSA. Nevertheless, SLSA like ArSenL does not include dialect words and cannot accurately analyze social media text.

Mohammad et al., (2015), generated three Arabic lexicons from Twitter. Three datasets were collected from Twitter: the first was tweets that contained the emoticons:":)" and ":(", the second was tweets that contained a seed list of positive and negative Arabic words as hashtags and the third was also from tweets that contained Arabic positive and negative words as hashtags but these were dialectal words. Then using PMI three lexicons were generated from these datasets: Arabic Emoticon Lexicon, Arabic Hashtag Lexicon and Dialectal Arabic Hashtag Lexicon. Our approach in generating one of the lexicons is very similar and thus we use one of their lexicons in the experiments to compare with our lexicons. The best performing lexicon was the Dialectal Arabic Hashtag Lexicon therefore we use it in this paper to compare and evaluate our lexicons.

3 Dataset Collection

We followed the approaches in previous work on SA of English Twitter to collect the datasets. As in (Go et al., 2009; Pak and Paroubek, 2010) we utilized emoticons as noisy labels to construct the first dataset EMO-TWEET. Tweets containing the emoticons: ":)" and ":(" and the rule "lang:ar" (to retrieve Arabic tweets only) were collected during November and December 2015. The total number of Tweets collected is shown in Table 1.

Davidov et al., (2010) and Kiritchenko et al., (2014) used hashtags of sentiment words such as #good and #bad to create corpora of positive and negative tweets, we adopted a similar method to theirs. Initially, we tried collecting tweets that contain Arabic sentiment words with hashtags but the search results were too low. We designated this result to a cultural difference in using hashtags between the western and eastern societies. Arabs do not use hashtags in this way. Accordingly we opted to use the sentiment words as keywords without the hashtag sign and the number of search results was substantial. Tweets containing 10 Arabic words having positive polarity and 10 Arabic words having negative polarity were collected during January 2016. The keywords are in Table 2 and the number of tweets collected in Table1. These results constitute our second dataset KEY-TWEET.

Retweets, tweets containing URLs or media and tweets containing non-Arabic words were all excluded from the dataset. The reason for excluding tweets with URLs and media is that we found that most of the tweets that contain URLS and media were spam. We also noticed that although we had specified in the search query that the fetched tweets should be in Arabic "lang:ar" some of the tweets were in English and other languages. So we had to add a filter to eliminate tweets with non-Arabic characters.

In total, the number of collected tweets was around 6.3 million Arabic tweets in a time span of three months. After filtration and cleaning of

the tweets, the remaining were 2.2 million tweets.

	EMO-TWEET		*KEY-TWEET*	
	Positive Emoticon :)	*Negative Emoticon :(*	*Positive keywords*	*Negative keywords*
Total number of tweets collected	2,245,054	1,272,352	1,823,517	1,000,212
After cleaning and filtering	1,033,393	407,828	447,170	337,535
Number of Tokens	12,739,308	5,082,070	9,058,412	7,135,331

Table 1: Number of collected tweets, number of tweets in datasets after cleaning and filtering and number of tokens in each dataset.

Positive Keywords	*English Translation*	*Negative Keywords*	*English Translation*
سعادة sEAdp	Happiness	محزن mHzn	Sad
خير xyr	Good	مؤسف m&sf	Regrettable
تفاؤل tfA&l	Optimism	للأسف ll>sf	Unfortunately
أعجبني >Ejbny	I like it	فاشل fA$l	Failing, unsuccessful
نجاح njAH	Success	تشاؤم t$A&m	Pessimism
فرح frH	Joy	سيء sy'	Bad
إيجابي <yjAby	Positive	سلبي slby	Negative
جيد jyd	Good	إهمال <hmAl	Negligence
ممتاز mmtAz	Excellent	خطأ xT>	Wrong
رائع rA}E	Fabulous	مؤلم m&lm	Painful

Table 2: Positive and negative keywords used to collect tweets.

4 Lexicon Generation

Two sentiment lexicons were extracted from the datasets of tweets using two different approaches. We call the first **AraSenTi-Trans** and the second **AraSenTi-PMI**. The approaches are presented in the following subsections.

4.1 AraSenTi-Trans

The datasets of tweets were processed using the MADAMIRA tool (Pasha et al., 2014). MADAMIRA is a recent effort by Pasha et al. (2014) that combines some of the best aspects of two previous systems used for Arabic NLP: MADA-Morphological Analysis and Disambiguation of Arabic (Habash and Rambow, 2005; Roth et al., 2008; Habash et al., 2009; Habash et al., 2013) and AMIRA (Diab et al., 2007). MADAMIRA, on the other hand, improves on these two systems with a solution that is more robust, portable, extensible, and faster.

The MADAMIRA tool identifies words into three types: ARABIC, NO_ANALYSIS and NON_ARABIC. This feature was used to eliminate tweets containing non-Arabic words and to distinguish MSA words from dialect words as NO_ANALYSIS words can be identified as dialect words or misspelled words or new words made up by tweepers (twitter users). According to the POS tags provided by MADAMIRA, we extracted only nouns, adjectives, adverbs, verbs and negation particles in an effort to eliminate unwanted stop words.

Then we utilized two popular English sentiment lexicons that were used in previous work on English and Arabic sentiment analysis: the Liu lexicon (Hu and Liu, 2004) and the MPQA lexicon (Wilson et al., 2005).

Most previous papers on Arabic SA that used these lexicons just translated them into Arabic, yet we tried a different approach. MADAMIRA provides an English gloss for each word identified as ARABIC, the gloss could be one, two or three words. We used this gloss to compare with the Liu lexicon and MPQA lexicon using the following heuristics:

- If all the word's glosses are positive in both lexicons or found in one lexicon as positive and do not exist in the other lexicon: classify as positive.
- If all the word's glosses are negative in both lexicons or found in one lexicon as negative and do not exist in the other: classify as negative.
- If the word's glosses have different polarities in the lexicons or are (both) in MPQA: add to both list.
- Else: all remaining words are classified as neutral.

Although this approach could contain some errors, a manual check can be performed to clean up. The manual cleanup is time consuming but it is a one-time effort that requires only a few days (Liu, 2012). Accordingly we gave the automati-

cally generated lists of positive, negative, both, and neutral words to two Arabic native speakers to review and correct the errors. We found that 5% of the neutral words were incorrectly misclassified as neutral while they were sentiment bearing words. Also 10% of the positive words were misclassified as negative, and 15% of the negative words were misclassified as positive. The lists were corrected accordingly. We can conclude that using translated English lexicons does not always give us accurate classification of polarity. This result could be due to mistranslations or cultural differences in classifying sentiment as demonstrated by (Mohammad et al., 2015; Mobarz et al., 2014; Duwairi, 2015). Accordingly, we propose a different approach to generating another lexicon in the following section.

4.2 AraSenti-PMI

The second lexicon was also generated from the dataset of tweets but through calculating the pointwise mutual information (PMI) measure for all words in the positive and negative datasets of tweets. The PMI is a measure of the strength of association between two words in a corpus, i.e. the probability of the two words to co-occur in the corpus (Church and Hanks, 1990). It has been adapted in sentiment analysis as a measure of the frequency of a word occurring in positive text to the frequency of the same word occurring in negative text. Turney, (2002); Turney and Littman, (2002) was the first work that proposed to use this measure in sentiment analysis. They used the once popular AltaVista search engine to find the sentiment of a certain word through calculating the PMI between the word and a set of positive words minus the PMI between the word and a set of negative words. Other works that used PMI to generate sentiment lexicons can be found in (Kiritchenko et al., 2014; Mohammad et al., 2015).

The frequencies of the words in the positive and negative datasets of tweets were calculated respectively then the PMI was calculated for each as follows:

$$PMI(w, pos) = \log_2 \frac{freq(w,pos)*N}{freq(w)*freq(pos)} \quad (1)$$

where *freq(w,pos)* is the frequency of the word *w* in the positive tweets, *freq(w)* is the frequency of the word *w* in the dataset, *freq(pos)* is the total number of tokens in the positive tweets and *N* is the total number of tokens in the dataset. The PMI of the word associated with negative tweets is calculated in the same way *PMI(w,neg).* The sentiment score for word *w* will be:

$$Sentiment\ Score(w)=PMI(w,pos)-PMI(w,neg) \quad (2)$$

This was calculated for all words that occurred in the dataset five times or more, the reason for this is that the PMI is a poor estimator of low-frequency words (Kiritchenko et al., 2014), so words occurring less than 5 times were excluded. Also for words that are found in the set of positive tweets but not in the set of negative tweets or vice versa, Equation 2 would give us a sentiment score of ∞, which would highly affect the calculation of the sentiment of the whole tweet. Since the absence of a word from the negative dataset does not require that the word's sentiment is positive or vice versa; as such we calculated the sentiment score of such words as in Equation 1, *PMI(w,pos)* for words occurring only in the positive tweets and *PMI(w,neg)* for words occurring only in the negative tweets.

4.3 Lexicons Coverage

The number of positive and negative entries in each of the lexicons is shown in Table 3. The details of the lexicon of (Mohammad et al., 2015) are also shown since this lexicon will be used in the experiments in the following section for evaluation and comparison purposes. Mohammad et al., (2015) generated three lexicons, however they demonstrated that the Dialectal Arabic Hashtag Lexicon (DAHL) gave the best results and accordingly we use this lexicon in the experiments in this paper. From Table 3, we can see the high coverage of the generated lexicons AraSenti-Trans and AraSenti-PMI when compared to DAHL. In addition we manually examined the three lexicons of (Mohammad et al., 2015) and found that they were not cleaned. They contained non-Arabic words and hashtags that do not convey sentiment. This put a question mark on the validity of the lexicons and the number of entries reported. Our datasets were cleaned from non-Arabic words and punctuation, so the generated lexicons all contain valid Arabic words.

Lexicon	*Positive*	*Negative*	*Total*
AraSenti-Trans	59,525	71,817	131,342
AraSenti-PMI	56,938	37,023	93,961
DAHL	11,947	8,179	20,126

Table 3: Details of the generated lexicons and the lexicon they will be compared to.

5 Evaluation

To evaluate the performance of the tweet-specific lexicons, we performed a set of experiments using a simple lexicon-based approach, hence no training and/or tuning is required. We performed a two-way classification on the datasets (positive or negative). We leave the problem of three and four way classification (positive, negative, neutral, mixed) for future work. We evaluated the generated lexicons on a dataset of 10,133 tweets extracted from the larger datasets of tweets EMO-TWEET and KEY-TWEET. The tweets were manually annotated by three annotators that are Arabic native speakers. The conflict between annotators was resolved by majority voting. We will call this dataset AraSenTi-Tweet. We also evaluated the generated lexicons on two external datasets of tweets: ASTD by (Nabil et al., 2015) and RR by (Refaee and Rieser, 2014). We extracted only the tweets that were labeled as positive or negative from these datasets. The details of all the datasets used in the experiments are illustrated in Table 4. We plan to release the dataset and the generated lexicons for the public.

Dataset	*Positive*	*Negative*	*Total*
AraSenti-Tweet	4329	5804	10133
ASTD	797	1682	2479
RR	876	1941	2817

Table 4: Datasets used in the evaluation of the generated lexicons.

Negation significantly affects the sentiment of its scope and consequently affects the evaluation of the lexicons. Accordingly, we propose to evaluate the generated lexicons in two settings: with and without negation handling. We also compare the performance of the generated lexicons with a lexicon that was generated in a very similar approach to one of the lexicons.

Since the datasets are unbalanced, we will report the performance measures of the macro-averaged F-score (F_{avg}), precision (P) and recall (R) of the positive and negative classes as follows:

$$P = TP/(TP+FP) \quad (3)$$

$$R = TP/(TP+FN) \quad (4)$$

$$F = 2*PR/P+R \quad (5)$$

where in the case of the positive class: TP is the number of positive tweets classified correctly as positive (true positive), FP is the number of negative tweets falsely classified as positive (false positive), and FN is the number of positive tweets falsely classified as negative (false negatives). The same holds for the negative class. Then the F-score is calculated as:

$$F_{avg} = \frac{F_{pos}+F_{neg}}{2} \quad (6)$$

5.1 Setup A: No Negation Handling

For the **AraSenTi-Trans** lexicon, we use the simple method of counting the number of positive and negative words in the tweet and whichever is the greatest denotes the sentiment of the tweet. The results of applying this method on the different datasets are illustrated in Table 5.

As for the **AraSenTi-PMI** lexicon, the sentiment score of all words in the tweet were summed up. The natural threshold to classify the data into positive or negative would be zero, since positive scores denote positive sentiment and negative scores denote negative sentiment. However, according to (Kiritchenko et al., 2014) other thresholds could give better results. Consequently, we experimented with the value of this threshold. We set it to 0, 0.5,and 1 and found that the best results were obtained when setting the threshold to 1. As such if the sum of the sentiment scores of the words in a tweet is greater than one, then the tweet is classified as positive, otherwise the tweet is classified as negative.

5.2 Setup B:Negation Handling

We also experimented with handling negation in the tweet, by compiling a list of negation particles found in the tweets and checking if the tweet contains a negation particle or not.

For the **AraSenTi-Trans** lexicon, if the tweet contains a negation particle and a positive word, we do not increment the positive word counter. However, for tweets containing negative words and negation particles we found that not incrementing the negative word counter degraded the accuracy, so we opted to increment the negative word counter even if a negation particle is found in the tweet.

Moreover, we experimented with adjusting the score of negation particles in the **AraSenTi-PMI** lexicon. After several experiments, we found that adjusting the score of the negation particles to -1 was the setting that gave the best performance.

6 Discussion and Results

The results of the first experimental setup for the two generated lexicons AraSenti-Trans and AraSenti-PMI are presented in Table 5. For the RR dataset and AraSenti-Tweet dataset, the superiority of the AraSenti-PMI lexicon is evident. The F_{avg} of applying the AraSenti-PMI lexicon on the RR dataset is 63.6% while the F_{avg} of applying the AraSenti-PMI lexicon on the AraSenti-Tweet dataset is 88.92%. As for the ASTD dataset, applying the AraSenti-Trans lexicon gave better results with an F_{avg} of 59.8%.

In Table 6, the results of the lexicon-based method with negation handling are presented. The results of using the DAHL lexicon on the same datasets are also reported for comparison.

First of all, the effect of negation handling on performance is significant, with increases of (1-4%) on all datasets. Although the two lexicons AraSenti-Trans and AraSenti-PMI handled negation differently but the increase for every dataset was almost the same: the ASTD dataset +4%, the RR dataset +1% and the AraSenti-Tweet dataset +2% and +1% respectively.

When comparing the performance of the generated lexicons AraSenti-Trans and AraSenti-PMI with the DAHL lexicon, we find that our lexicons presented better classification results on all datasets.

Finally, although the two lexicons were extracted from the same dataset, we find that their performance varied on the different datasets. The best performance for the ASTD dataset was when the AraSenti-Trans lexicon was used. However, the best performance for the RR and AraSenti-Tweet datasets was when the AraSenti-PMI lexicon was used. Moreover, albeit the simple lexicon-based method used in the evaluation, we find that the performance is encouraging. Several enhancements could be made such as incorporating Arabic valence shifters and certain linguistic rules to handle them.

	Lexicon									
DataSet	AraSenti-Trans					AraSenti-PMI				
	Positve		Negative		F_{avg}	Positve		Negative		F_{avg}
	P	R	P	R		P	R	P	R	
ASTD	43.92	90.21	90.74	45.42	**59.80**	37.24	77.79	78.26	37.87	50.70
RR	40.66	89.95	89.99	40.75	56.05	46.01	73.74	83.72	60.95	**63.60**
AraSenti-Tweet	63.14	95.43	94.48	58.44	74.11	85.73	89.37	91.81	88.9	**88.92**

Table 5: Results of the first experimental setup without negation handling on the generated lexicons AraSenti-Trans and AraSenti-PMI.

	Lexicon														
DataSet	AraSenti-Trans					AraSenti-PMI					DAHL				
	Positve		Negative		F_{avg}	Positve		Negative		F_{avg}	Positve		Negative		F_{avg}
	P	R	P	R		P	R	P	R		P	R	P	R	
ASTD	46.24	86.32	89	52.44	**63.10**	38.06	56.59	73.26	56.36	54.61	36.4	43.16	70.47	64.27	53.36
RR	41.31	86.3	87.84	44.67	57.55	52.03	49.77	77.77	79.29	**64.70**	38.06	38.58	72.11	71.66	55.10
AraSenti-Tweet	66.27	90.76	90.49	65.54	76.31	91.16	84.57	89.08	93.88	**89.58**	76.35	62.88	75.53	85.48	74.58

Table 6: Results of the second experimental setup with negation handling on the generated lexicons AraSenti-Trans and AraSenti-PMI and on the external lexicon DAHL

7 Conclusion

In this paper, two large-scale Arabic sentiment lexicons were generated from a large dataset of Arabic tweets. The significance of these lexicons lies in their ability to capture the idiosyncratic nature of social media text. Moreover, their high coverage suggests the possibility of using them in different genres such as product reviews. This is a possible future research direction.

The performance of the lexicons on external datasets also suggests their ability to be used in classifying new datasets. However, there is much room for improvement given the simple method

used in evaluation. This simple lexicon-based method could be further enhanced by incorporating Arabic valence shifters and certain linguistic rules to handle them. We also plan to make the classification multi-way: positive, negative, neutral and mixed.

Acknowledgments

This Project was funded by the National Plan for Science, Technology and Innovation (MAARIFAH), King Abdulaziz City for Science and Technology, Kingdom of Saudi Arabia, Award Number (GSP-36-332).

References

Muhammad Abdul-Mageed and Mona Diab. 2014. SANA: A Large Scale Multi-Genre, Multi-Dialect Lexicon for Arabic Subjectivity and Sentiment Analysis. In *In Proceedings of the Language Resources and Evaluation Conference (LREC)*, Reykjavik, Iceland.

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. In *LREC*, volume 10, pages 2200–2204.

Gilbert Badaro, Ramy Baly, Hazem Hajj, Nizar Habash, and Wassim El-Hajj. 2014. A large scale Arabic sentiment lexicon for Arabic opinion mining. *ANLP 2014*:165.

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. 2008. Building a sentiment summarizer for local service reviews. In *WWW Workshop on NLP in the Information Explosion Era*, volume 14, pages 339–348.

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

Kareem Darwish and Walid Magdy. 2014. Arabic Information Retrieval. *Foundations and Trends in Information Retrieval*, 7(4):239–342.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 241–249. Association for Computational Linguistics.

Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. 2007. Automated methods for processing arabic text: from tokenization to base phrase chunking. *Arabic Computational Morphology: Knowledge-based and Empirical Methods. Kluwer/Springer*.

Rehab M Duwairi. 2015. Sentiment analysis for dialectical Arabic. In *6th International Conference on Information and Communication Systems (ICICS), 2015*, pages 166–170. IEEE.

Ramy Eskander and Owen Rambow. 2015. SLSA: A Sentiment Lexicon for Standard Arabic. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2545–2550, Lisbon,Purtogal, September. ACL.

Andrea Esuli and Fabrizio Sebastiani. 2005. Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 617–624. ACM.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06)*, volume 6, pages 417–422.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*:1–12.

Nizar Habash and Owen Rambow. 2005. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics.

Nizar Habash, Owen Rambow, and Ryan Roth. 2009. Mada+ tokan: A toolkit for arabic tokenization, diacritization, morphological disambiguation, pos tagging, stemming and lemmatization. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt*, pages 102–109.

Nizar Habash, Ryan Roth, Owen Rambow, Ramy Eskander, and Nadi Tomeh. 2013. Morphological Analysis and Disambiguation for Dialectal Arabic. In *HLT-NAACL*, pages 426–432. Citeseer.

Nizar Y Habash. 2010. Introduction to Arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th annual meeting of the association for computational linguistics and eighth conference of the european chapter of the as-*

sociation for computational linguistics, pages 174–181. Association for Computational Linguistics.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Internet World Stats. 2015. Internet World Stats. November.

Jaap Kamps. 2004. Using wordnet to measure semantic orientations of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Hanaa Mobarz, Mohsen Rashown, and Ibrahim Farag. 2014. Using Automated Lexical Resources in Arabic Sentence Subjectivity. *International Journal of Artificial Intelligence & Applications*, 5(6):1.

Saif M Mohammad, Mohammad Salameh, and Svetlana Kiritchenko. 2015. How Translation Alters Sentiment. *Journal of Artificial Intelligence Research*, 54:1–20.

Mahmoud Nabil, Mohamed Aly, and Amir F Atiya. 2015. ASTD: Arabic Sentiment Tweets Dataset. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2515–2519.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC 2010)*, Valleta,Malta. European Language Resources Association (ELRA).

Arfath Pasha, Mohamed Al-Badrashiny, Ahmed El Kholy, Ramy Eskander, Mona Diab, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan Roth. 2014. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *In Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Delip Rao and Deepak Ravichandran. 2009. Semi-supervised polarity lexicon induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 675–682. Association for Computational Linguistics.

Eshrag Refaee and Verena Rieser. 2014. An Arabic Twitter Corpus for Subjectivity and Sentiment Analysis. In *In Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ryan Roth, Owen Rambow, Nizar Habash, Mona Diab, and Cynthia Rudin. 2008. Arabic morphological tagging, diacritization, and lemmatization using lexeme models and feature ranking. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 117–120. Association for Computational Linguistics.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014. Building Large-Scale Twitter-Specific Sentiment Lexicon: A Representation Learning Approach. In *COLING*, pages 172–182.

Peter D Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

Peter Turney and Michael L Littman. 2002. Unsupervised learning of semantic orientation from a hundred-billion-word corpus. Technical report, National Research Council Canada, NRC Institute for Information Technology; National Research Council Canada.

Gbolahan K Williams and Sarabjot Singh Anand. 2009. Predicting the Polarity Strength of Adjectives Using WordNet. In *Third International AAAI Conference on Weblogs and Social Media*.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Unsupervised Multi-Author Document Decomposition Based on Hidden Markov Model

Khaled Aldebei **Xiangjian He** **Wenjing Jia**
Global Big Data Technologies Centre
University of Technology Sydney
Australia
{Khaled.Aldebei,Xiangjian.He,Wenjing.Jia}@uts.edu.au

Jie Yang
Lab of Pattern Analysis
and Machine Intelligence
Shanghai Jiaotong University
China
Jieyang@sjtu.edu.cn

Abstract

This paper proposes an unsupervised approach for segmenting a multi-author document into authorial components. The key novelty is that we utilize the sequential patterns hidden among document elements when determining their authorships. For this purpose, we adopt Hidden Markov Model (HMM) and construct a sequential probabilistic model to capture the dependencies of sequential sentences and their authorships. An unsupervised learning method is developed to initialize the HMM parameters. Experimental results on benchmark datasets have demonstrated the significant benefit of our idea and our approach has outperformed the state-of-the-arts on all tests. As an example of its applications, the proposed approach is applied for attributing authorship of a document and has also shown promising results.

1 Introduction

Authorship analysis is a process of inspecting documents in order to extract authorial information about these documents. It is considered as a general concept that embraces several types of authorship subjects, including *authorship verification*, *plagiarism detection* and *author attribution*. Authorship verification (Brocardo et al., 2013; Potha and Stamatatos, 2014) decides whether a given document is written by a specific author. Plagiarism detection (Stein et al., 2011; Kestemont et al., 2011) seeks to expose the similarity between two texts. However, it is unable to determine if they are written by the same author. In author attribution (Juola, 2006; Savoy, 2015), a real author of an anonymous document is predicted using labeled documents of a set of candidate authors.

Another significant subject in authorship analysis, which has received comparatively less attention from research community, is *authorship-based document decomposition* (ABDD). This subject is to group the sentences of a multi-author document to different classes, of which each contains the sentences written by only one author. Many applications can take advantage of such a subject, especially those in forensic investigation, which aim to determine the authorship of sentences in a multi-author document. Furthermore, this kind of subject is beneficial for detecting plagiarism in a document and defining contributions of authors in a multi-author document for commercial purpose. ABDD can also be applied to identify which source (regarded as an 'author' in this paper) a part of a document is copied from when the document is formed by taking contents from various sources.

In despite of the benefits of ABDD, there has been little research reported on this subject. Koppel et al. (2011) are the first researchers who implemented an unsupervised approach for ABDD. However, their approach is restricted to Hebrew documents only. The authors of Akiva and Koppel (2013) addressed the drawbacks of the above approach by proposing a generic unsupervised approach for ABDD. Their approach utilized distance measurements to increase the precision and accuracy of clustering and classification phases, respectively. The accuracy of their approach is highly dependent on the number of au-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 706–714,
Berlin, Germany, August 7-12, 2016.

thors. When the number of authors increases, the accuracy of the approach is significantly dropped. Giannella (2015) presented an improved approach for ABDD when the number of authors of the document is known or unknown. In his approach, a Bayesian segmentation algorithm is applied, which is followed by a segment clustering algorithm. However, the author tested his approach by using only documents with a few transitions among authors. Furthermore, the accuracy of the approach is very sensitive to the setting of its parameters. In Aldebei et al. (2015), the authors presented an unsupervised approach ABDD by exploiting the differences in the posterior probabilities of a Naive-Bayesian model in order to increase the precision and the classification accuracy, and to be less dependent on the number of authors in comparing with the approach in Akiva and Koppel (2013). Their work was tested on documents with up to 400 transitions among authors and the accuracy of their approach was not sensitive to the setting of parameters, in contrast with the approach in Giannella (2015). However, the performance of their approach greatly depends on a threshold, of which the optimal value for an individual document is not easy to find.

Some other works have focused on segmenting a document into components according to their topics. For applications where the topics of documents are unavailable, these topic-based solutions will fail. In this paper, the ABDD approach is independent of documents' topics.

All of the existing works have assumed that the observations (i.e., sentences) are independent and identically distributed (i.i.d.). No consideration has been given to the contextual information between the observations. However, in some cases, the i.i.d. assumption is deemed as a poor one (Rogovschi et al., 2010). In this paper, we will relax this assumption and consider sentences of a document as a sequence of observations. We make use of the contextual information hidden between sentences in order to identify the authorship of each sentence in a document. In other words, the authorships of the "previous" and "subsequent" sentences have relationships with the authorship of the current sentence. Therefore, in this paper, a well-known sequential model, Hidden Markov Model (HMM), is used for modelling the sequential patterns of the document in order to describe the authorship relationships.

The contributions of this article are summarized as follows.

1. We capture the dependencies between consecutive elements in a document to identify different authorial components and construct an HMM for classification. It is for the first time the sequential patterns hidden among document elements is considered for such a problem.

2. To build and learn the HMM model, an unsupervised learning method is first proposed to estimate its initial parameters, and it does not require any information of authors or document's context other than how many authors have contributed to write the document.

3. Different from the approach in Aldebei et al. (2015), the proposed unsupervised approach no longer relies on any predetermined threshold for ABDD.

4. Comprehensive experiments are conducted to demonstrate the superior performance of our ideas on both widely-used artificial benchmark datasets and an authentic scientific document. As an example of its applications, the proposed approach is also applied for attributing authorship on a popular dataset. The proposed approach can not only correctly determine the author of a disputed document but also provide a way for measuring the confidence level of the authorship decision for the first time.

The rest of this article is organised as follows. Section 2 reviews the HMM. Section 3 presents the details of our proposed approach, including the processes for initialization and learning of HMM parameters, and the Viterbi decoding process for classification. Experiments are conducted in Section 4, followed by the conclusion in Section 5.

2 Overview of HMM

In this paper, we adopt the widely used sequential model, the Hidden Markov Model (HMM) (Eddy, 1996), to classify sentences of a multi-author document according to their authorship. The HMM is a probabilistic

model which describes the statistical dependency between a sequence of observations $O = \{o_1, o_2, \cdots, o_T\}$ and a sequence of hidden states $Q = \{q_1, q_2, \cdots, q_T\}$. The observations can either be discrete variables, where each o_i takes a value from a set of M symbols $W = \{w_1, \cdots, w_M\}$, or be continuous variables. On the other hand, each q_i takes one possible value from a set of N symbols, $S = \{s_1, \cdots, s_N\}$.

The behaviour of the HMM can be determined by three parameters shown as follows.

1. Initial state probabilities $\boldsymbol{\pi} = \{\pi_1, \cdots, \pi_N\}$, where $\pi_n = p(q_1 = s_n)$ and $s_n \in S$, for $n = 1, 2, \cdots, N$.

2. Emission probabilities $\boldsymbol{B}$, where each emission probability $b_n(o_t) = p(o_t|q_t = s_n)$, for $t = 1, 2, \cdots, T$ and $n = 1, 2, \cdots, N$.

3. State transition probabilities $\boldsymbol{A}$. It is assumed that the state transition probability has a time-homogeneous property, i.e., it is independent of the time t. Therefore, a probability $p(q_t = s_l|q_{t-1} = s_n)$ can be represented as a_{nl}, for $t = 1, 2, \cdots, T$ and $l, n = 1, 2, \cdots, N$.

3 The Proposed Approach

The ABDD proposed in this paper can be formulated as follows. Given a multi-author document C, written by N co-authors, it is assumed that each sentence in the document is written by one of the N co-authors. Furthermore, each co-author has written long successive sequences of sentences in the document. The number of authors N is known beforehand, while typically no information about the document contexts and co-authors is available. Our objective is to define the sentences of the document that are written by each co-author.

Our approach consists of three steps shown as follows.

1. Estimate the initial values of the HMM parameters $\{\boldsymbol{\pi}, \boldsymbol{B}, \boldsymbol{A}\}$ with a novel unsupervised learning method.

2. Learn the values of the HMM parameters using the $Baum - Welch$ algorithm (Baum, 1972; Bilmes and others, 1998).

3. Apply the $Viterbi$ algorithm (Forney Jr, 1973) to find the most likely authorship of each sentence.

3.1 Initialization

In our approach, we assume that we do not know anything about the document C and the authors, except the number of co-authors of the document (i.e., N). This approach applies an HMM in order to classify each sentence in document C into a class corresponding to its co-author. The step (see Sub-section 3.2) for learning of HMM parameters $\{\boldsymbol{\pi}, \boldsymbol{B}, \boldsymbol{A}\}$ is heavily dependent on the initial values of these parameters (Wu, 1983; Xu and Jordan, 1996; Huda et al., 2006). Therefore, a good initial estimation of the HMM parameters can help achieve a higher classification accuracy.

We take advantage of the sequential information of data and propose an unsupervised approach to estimate the initial values of the HMM parameters. The detailed steps of this approach are shown as follows.

1. The document C is divided into *segments*. Each segment has 30 successive sentences, where the i^{th} segment comprises the i^{th} 30 successive sentences of the document. This will produce s segments, where $s =$ Ceiling($|C|/30$) with $|C|$ representing the total number of sentences in the document. The number of sentences in each segment (i.e., 30) is chosen in such a way that each segment is long enough for representing a particular author's writing style, and also the division of the document gives an adequate number of segments in order to be used later for estimating the initial values of HMM parameters.

2. We select the words appearing in the document for more than two times. This produces a set of D words. For each segment, create a D-dimensional vector where the i^{th} element in the vector is one (zero) if the i^{th} element in the selected word set does (not) appear in the segment. Therefore, s binary D-dimensional vectors are generated, and the set of these vectors is denoted by $X = \{x_1, \cdots, x_s\}$.

3. A multivariate Gaussian Mixture Models (GMMs) (McLachlan and Peel, 2004) is used to cluster the D-dimensional vectors X into N components denoted by $\{s_1, s_2, \cdots, s_N\}$. Note that the number of components is equal to the number of co-authors of the document. Based on the GMMs, each vector, x_i, gets a label representing the Gaussian component that this vector x_i is assigned to, for $i = 1, 2, \cdots, s$.

4. Again, we represent each segment as a binary vector using a new feature set containing all words appearing in the document for at least once. Assuming the number of elements in the new feature set is D', s binary D'-dimensional vectors are generated, and the set of these vectors is denoted by $X' = \{x'_1, \cdots, x'_s\}$. Each vector x'_i will have the same label of vector x_i, for $i = 1, 2, \cdots, s$.

5. We construct a Hidden Markov model with a sequence of observations O' and its corresponding sequence of hidden states Q'. In this model, O' represents the resulted segment vectors X' of the previous step. Formally, observation o'_i, is the i^{th} binary D'-dimensional vector x'_i, that represents the i^{th} segment of document C. In contrast, Q' represents the corresponding authors of the observation sequence O'. Each q'_i symbolizes the most likely author of observation o'_i. According to Steps 3 and 4 of this sub-section, each x'_i representing o'_i takes one label from a set of N elements, and the label represents its state, for $i = 1, 2, \cdots, s$.

By assigning the most likely states to all hidden states (i.e., $q'_i, i = 1, 2, \cdots, s$), the state transition probabilities $\boldsymbol{A}$ are estimated.

As long as there is only one sequence of states in our model, the initial probability of each state is defined as the fraction of times that the state appears in the sequence Q', so $\pi_n = \frac{Count(q'=s_n)}{Count(q')}$, for $n = 1, 2, \cdots, N$.

6. Given the sequence X', and the set of all possible values of labels, the conditional probability of feature f_k in X' given a label s_n, $p(f_k|s_n)$, is computed, for $k = 1, 2, \cdots, D'$ and $n = 1, 2, \cdots, N$.

7. The document C is partitioned into sentences. Let $z = |C|$ represent the number of sentences in the document. We represent each sentence as a binary feature vector using the same feature set used in Step 4. Therefore, z binary D'-dimensional vectors, denoted by $O = \{o_1, \cdots, o_z\}$, are generated. By using the conditional probabilities resulted in Step 6, the initial values of $\boldsymbol{B}$ are computed as $p(o_i|s_n) = \prod_{k=1}^{D'} o_i^{f_k} p(f_k|s_n)$, where $o_i^{f_k}$ represents the value of feature f_k in sentence vector o_i, for $i = 1, 2, \cdots, z$ and $n = 1, 2, \cdots, N$.

In this approach, we use *add-one smoothing* (Martin and Jurafsky, 2000) for avoiding zero probabilities of $\boldsymbol{A}$ and $\boldsymbol{B}$. Furthermore, we take the logarithm function of the probability in order to simplify its calculations.

The initial values of the $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{\pi}$ are now available. In next sub-section, the learning process of these parameter values is performed.

3.2 Learning HMM

After estimating the initial values for the parameters of HMM, we now find the parameter values that maximize likelihood of the observed data sequence (i.e., sentence sequence). The learning process of the HMM parameter values is performed as follows.

1. Construct a Hidden Markov model with a sequence of observations, O, and a corresponding sequence of hidden states, Q. In this model, O represents the resulted sentence vectors (Step 7 in the previous Sub-section). Formally, the observation o_i, is the i^{th} binary D'-dimensional vector and it represents the i^{th} sentence of document C. In contrast, Q represents the corresponding authors of observation sequence O. Each q_i symbolizes the most likelihood author of observation o_i, for $i = 1, 2, \cdots, z$

2. The Baum-Welch algorithm is applied to learn the HMM parameter values. The algorithm, also known as the $forward-backward$ algorithm (Rabiner, 1989), has two steps, i.e., *E-step* and *M-step*. The *E-step* finds the expected author sequence (Q) of the observation sequence (O), and the *M-step* updates the HMM parameter values according to the state assignments. The learning procedure starts with the initial values of HMM parameters, and then the cycle of these two steps continues until a convergence is achieved in $\boldsymbol{\pi}$, $\boldsymbol{B}$ and $\boldsymbol{A}$.

The learned HMM parameter values will be used in the next sub-section in order to find the best sequence of authors for the given sentences.

3.3 Viterbi Decoding

For a Hidden Markov model, there are more than one sequence of states in generating the observation sequence. The Viterbi decoding algorithm (Forney Jr, 1973) is used to determine the best sequence of states for generat-

ing observation sequence. Therefore, by using the Hidden Markov model that is constructed in previous sub-section and the learned HMM parameter values, the Viterbi decoding algorithm is applied to find the best sequence of authors for the given sentences.

4 Experiments

In this section, we demonstrate the performance of our proposed approach by conducting experiments on benchmark datasets as well as one authentic document. Furthermore, an application on authorship attribution is presented using another popular dataset.

4.1 Datasets

Three benchmark corpora widely used for authorship analysis are used to evaluate our approach. Furthermore, an authentic document is also examined.

The first corpus consists of five Biblical books written by Ezekiel, Isaiah, Jeremiah, Proverbs and Job, respectively. All of these books are written in Hebrew. The five books belong to two types of literature genres. The first three books are related to prophecy literature and the other two books are related to a wisdom literature.

The second corpus consists of blogs written by the Nobel Prize-winning economist Gary S. Becker and the renowned jurist and legal scholar Richard A. Posner. This corpus, which is titled "The Becker-Posner Blogs" (`www.becker-posner-blog.com`), contains 690 blogs. On average, each blog has 39 sentences talking about particular topic. The Becker-Posner Blogs dataset, which is considered as a very important dataset for authorship analysis, provides a good benchmark for testing the proposed approach in a document where the topics of authors are not distinguishable. For more challenging documents, Giannella (2015) has manually selected six single-topic documents from Becker-Posner blogs. Each document is a combination of Becker and Posner blogs that are talking about only one topic. The six merged documents with their topics and number of sentences of each alternative author are shown in Table 1.

Topics	Author order and number of sentences per author
Tenure (Ten)	Posner(73), Becker(36), Posner(33), Becker(19)
Senate Filibuster (SF)	Posner(39), Becker(36), Posner(28), Becker(24)
Tort Reform (TR)	Posner(29), Becker(31), Posner(24)
Profiling (Pro)	Becker(35), Posner(19), Becker(21)
Microfinance (Mic)	Posner(51), Becker(37), Posner(44), Becker(33)
Traffic Congestion (TC)	Becker(57), Posner(33), Becker(20)

Table 1: The 6 merged single-topic documents of Becker-Posner blogs.

The third corpus is a group of New York Times articles of four columnists. The articles are subjected to different topics. In our experiments, all possible multi-author documents of articles of these columnists are created. Therefore, this corpus permits us to examine the performance of our approach in documents written by more than two authors.

The fourth corpus is a very early draft of a scientific article co-authored by two PhD students each being assigned a task to write some full sections of the paper. We employ this corpus in order to evaluate the performance of our approach on an authentic document. For this purpose, we have disregarded its titles, author names, references, figures and tables. After that, we get 313 sentences which are written by two authors, where Author 1 has written 131 sentences and Author 2 has written 182 sentences.

4.2 Results on Document Decomposition

The performance of the proposed approach is evaluated through a set of comparisons with four state-of-the-art approaches on the four aforementioned datasets.

The experiments on the first three datasets, excluding the six single-topic documents, are applied using a set of artificially merged multi-author documents. These documents are created by using the same method that has been used by Aldebei et al. (2015). This method aims to combine a group of documents of N authors into a single merged document. Each of these documents is written by only one author. The merged document process starts by selecting a random author from an author set. Then, the first r successive and unchosen sentences from the documents of the selected author are gleaned, and are merged with the first r successive and unchosen sentences from the documents of another randomly selected au-

thor. This process is repeated till all sentences of authors' documents are gleaned. The value of r of each transition is selected randomly from a uniform distribution varying from 1 to V. Furthermore, we follow Aldebei et al. (2015) method and assign the value of 200 to V.

Bible Books

We utilize the bible books of five authors and create artificial documents by merging books of any two possible authors. This produces 10 multi-author documents of which four have the same type of literature and six have different type of literature. Table 2 shows the comparisons of classification accuracies of these 10 documents by using our approach and the approaches developed by Koppel et al. (2011), Akiva and Koppel (2013)-500CommonWords, Akiva and Koppel (2013)-SynonymSet and Aldebei et al. (2015).

	Doc.	1	2	3	4	5
Different	Eze-Job	85.8%	98.9%	95.0%	99.0%	99.4%
	Eze-Prov	77.0%	99.0%	91.0%	98.0%	98.8%
	Isa-Prov	71.0%	95.0%	85.0%	98.0%	98.7%
	Isa-Job	83.0%	98.8%	89.0%	99.0%	99.4%
	Jer-Job	87.2%	98.2%	93.0%	98.0%	98.5%
	Jer-Prov	72.2%	97.0%	75.0%	99.0%	99.5%
	Overall	***79.4%***	***97.8%***	***88.0%***	***98.5%***	***99.1%***
Same	Job-Prov	85.0%	94.0%	82.0%	95.0%	98.2%
	Isa-Jer	72.0%	66.9%	82.9%	71.0%	72.1%
	Isa-Eze	79.0%	80.0%	88.0%	83.0%	83.2%
	Jer-Eze	82.0%	97.0%	96.0%	97.0%	97.3%
	Overall	***79.5%***	***84.5%***	***87.2%***	***86.5%***	***87.7%***

Table 2: Classification accuracies of merged documents of different literature or the same literature bible books using the approaches of 1- Koppel et al. (2011), 2- Akiva and Koppel (2013)-500CommonWords, 3- Akiva and Koppel (2013)-SynonymSet, 4- Aldebei et al. (2015) and 5- our approach.

As shown in Table 2, the results of our approach are very promising. The overall classification accuracies of documents of the same literature or different literature are better than the other four state-of-the-art approaches.

In our approach, we have proposed an unsupervised method to estimate the initial values of the HMM parameters (i.e., $\boldsymbol{\pi}$, $\boldsymbol{B}$ and $\boldsymbol{A}$) using segments. Actually, the initial values of the HMM parameters are sensitive factors to the convergence and accuracy of the learning process. Most of the previous works using HMM have estimated these values by clustering the original data, i.e., they have clustered sentences rather than segments. Figure 1 compares the results of using segments with the results of using sentences for estimating the initial parameters of HMM in the proposed approach for the 10 merged Bible documents in terms of the accuracy results and number of iterations till convergence, respectively. From Figures 1, one can notice that the accuracy results obtained by using segments for estimating the initial HMM parameters are significantly higher than using sentences for all merged documents. Furthermore, the number of iterations required for convergence for each merged document using segments is significantly smaller than using sentences.

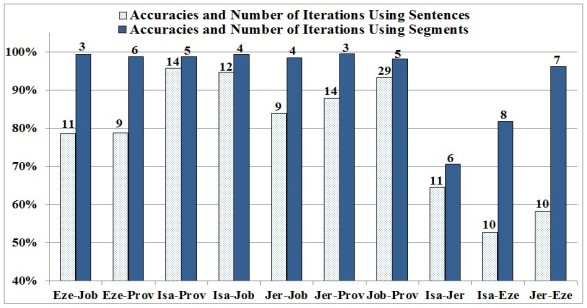


Figure 1: Comparisons between using segments and using sentences in the unsupervised method for estimating the initial values of the HMM of our approach in terms of accuracy (representd as the cylinders) and number of iterations required for convergence (represented as the numbers above cylinders) using the 10 merged Bible documents.

Becker-Posner Blogs (Controlling for Topics)

In our experiments, we represent Becker-Posner blogs in two different terms. The first term is as in Aldebei et al. (2015) and Akiva and Koppel (2013) approaches, where the whole blogs are exploited to create one merged document. The resulted merged document contains 26,922 sentences and more than 240 switches between the two authors. We obtain an accuracy of 96.72% when testing our approach in the merged document. The obtained result of such type of document, which does not have topic indications to differentiate between authors, is delightful. The first set of cylinders labelled "Becker-Posner" in Figure 2 shows the comparisons of classification accuracies of our approach and the approaches of Akiva and Koppel (2013) and Aldebei et al.

(2015) when the whole blogs are used to create one merged document. As shown in Figure 2, our approach yields better classification accuracy than the other two approaches.

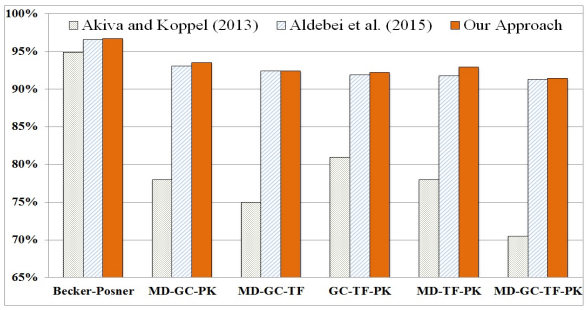


Figure 2: Classification accuracy comparisons between our approach and the approaches presented in Akiva and Koppel (2013) and Aldebei et al. (2015) in Becker-Posner documents, and documents created by three or four *New York Times* columnists (TF = Thomas Friedman, PK = Paul Krugman, MD = Maureeen Dowd, GC = Gail Collins).

The second term is as in the approach of Giannella (2015), where six merged single-topic documents are formed. Due to comparatively shorter lengths of these documents, the number of resulted segments that are used for the unsupervised learning in Sub-section 3.1 is clearly not sufficient. Therefore, instead of splitting each document into segments of 30 sentences length each, we split it into segments of 10 sentences length each. Figure 3 shows the classification accuracies of the six documents using our approach and the approach presented in Giannella (2015). It is observed that our proposed approach has achieved higher classification accuracy than Giannella (2015) in all of the six documents.

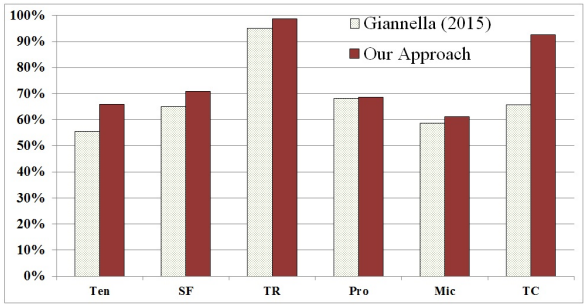


Figure 3: Classification accuracy comparisons between our approach and the approach presented in (Giannella, 2015) in the six single-topic documents of Becker-Posner blogs.

New York Times Articles (N > 2)

We perform our approach on *New York Times* articles. For this corpus, the experiments can be classified into three groups. The first group is for those merged documents that are created by combining articles of any pair of the four authors. The six resulted documents have on average more than 250 switches between authors. The classification accuracies of these documents are between 93.9% and 96.3%. It is notable that the results are very satisfactory for all documents. For comparisons, the classification accuracies of the same documents using the approach presented in Aldebei et al. (2015) range from 93.3% to 96.1%. Furthermore, some of these documents have produced an accuracy lower than 89.0% using the approach of Akiva and Koppel (2013).

The second group is for those merged documents that are created by combining articles of any three of the four authors. The four resulted documents have on average more than 350 switches among the authors. The third group is for the document that are created by combining articles of all four columnists. The resulted merged document has 46,851 sentences and more than 510 switches among authors. Figure 2 shows the accuracies of the five resulted documents regarding the experiments of the last two groups. Furthermore, it shows the comparisons of our approach and the approaches presented in Aldebei et al. (2015) and Akiva and Koppel (2013). It is noteworthy that the accuracies of our approach are better than the other two approaches in all of the five documents.

Authentic Document

In order to demonstrate that our proposed approach is applicable on genuine documents as well, we have applied the approach on first draft of a scientific paper written by two Ph.D. students (Author 1 and Author 2) in our research group. Each student was assigned a task to write some full sections of the paper. Author 1 has contributed 41.9% of the document and Author 2 contributed 58.1%. Table 3 shows the number of correctly assigned sentences of each author and the classification accuracy resulted using the proposed approach. Table 3 also displays the authors' contributions predicted using our approach. As

Author	Classification Accuracy	Predicted Contribution
1	98.5%	47.6%
2	89.0%	52.4%
Accuracy	**93.0%**	

Table 3: The classification accuracies and predicted contributions of the two authors of the scientific paper using the proposed approach.

shown in Table 3, the proposed approach has achieved an overall accuracy of 93.0% for the authentic document.

4.3 Results on Authorship Attribution

One of the applications that can take advantage of the proposed approach is the authorship attribution (i.e., determining a real author of an anonymous document given a set of labeled documents of candidate authors). The *Federalist Papers* dataset have been employed in order to examine the performance of our approach for this application. This dataset is considered as a benchmark in authorship attribution task and has been used in many studies related to this task (Juola, 2006; Savoy, 2013; Savoy, 2015). The *Federalist Papers* consist of 85 articles published anonymously between 1787 and 1788 by Alexander Hamilton, James Madison and John Jay to persuade the citizens of the State of New York to ratify the Constitution. Of the 85 articles, 51 of them were written by Hamilton, 14 were written by Madison and 5 were written by Jay. Furthermore, 3 more articles were written jointly by Hamilton and Madison. The other 12 articles (i.e., articles 49-58 and 62-63), the famous "anonymous articles", have been alleged to be written by Hamilton or Madison.

To predict a real author of the 12 anonymous articles, we use the first five undisputed articles of both authors, Hamilton and Madison. Note that we ignore the articles of Jay because the anonymous articles are alleged to be written by Hamilton or Madison. The five articles of Hamilton (articles 1 and 6-9) are combined with the five articles of Madison (articles 10, 14 and 37-39) in a single merged document where all the articles of Hamilton are inserted into the first part of the merged document and all the articles of Madison are inserted into the second part of the merged document. The merged document has 10 undisputed articles covering eight different topics (i.e., each author has four different topics). Before applying the authorship attribution on the 12 anonymous articles, we have tested our approach on the resulted merged document and an accuracy of 95.2% is achieved in this document. Note that, the authorial components in this document are not thematically notable.

For authorship attribution of the 12 anonymous articles, we add one anonymous article each time on the middle of the merged document, i.e., between Hamilton articles part and Madison articles part. Then, we apply our approach on the resulted document, which has 11 articles, to determine to which part the sentences of the anonymous article are classified to be sectences of Hamilton or Madison. As the ground truth for our experiments, all of these 12 articles can be deemed to have been written by Madison becuase the results of all recent state-of-the-art studies testing on these articles on authorship attribution have classified the articles to Madison's. Consistent with the state-of-the-art approaches, these 12 anonymous articles are also correctly classified to be Madison's using the proposed approach. Actually, all sentences of articles 50,52-58 and 62-63 are classified as Madison's sentences, and 81% of the sentences of article 49 and 80% of article 51 are classified as Madison's sentences. These percentages can be deemed as the confidence levels (i.e., 80% conferdence for articles 49, 81% for 51, and 100% confidences for all other articles) in making our conclusion of the authorship contributions.

5 Conclusions

We have developed an unsupervised approach for decomposing a multi-author document based on authorship. Different from the state-of-the-art approaches, we have innovatively made use of the sequential information hidden among document elements. For this purpose, we have used HMM and constructed a sequential probabilistic model, which is used to find the best sequence of authors that represents the sentences of the document. An unsupervised learning method has also been developed to estimate the initial parameter values of HMM. Comparative experiments conducted on benchmark datasets have demonstrated the effectiveness of our ideas with superior perfor-

mance achieved on both artificial and authentic documents. An application of the proposed approach on authorship attribution has also achieved perfect results of 100% accuracies together with confidence measurement for the first time.

References

[Akiva and Koppel2013] Navot Akiva and Moshe Koppel. 2013. A generic unsupervised method for decomposing multi-author documents. *Journal of the American Society for Information Science and Technology*, 64(11):2256–2264.

[Aldebei et al.2015] Khaled Aldebei, Xiangjian He, and Jie Yang. 2015. Unsupervised decomposition of a multi-author document based on naive-bayesian model. *ACL, Volume 2: Short Papers*, page 501.

[Baum1972] Leonard E Baum. 1972. An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, 3:1–8.

[Bilmes and others1998] Jeff A Bilmes et al. 1998. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. *International Computer Science Institute*, 4(510):126.

[Brocardo et al.2013] Marcelo Luiz Brocardo, Issa Traore, Shatina Saad, and Isaac Woungang. 2013. Authorship verification for short messages using stylometry. In *Computer, Information and Telecommunication Systems (CITS), 2013 International Conference on*, pages 1–6. IEEE.

[Eddy1996] Sean R Eddy. 1996. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365.

[Forney Jr1973] G David Forney Jr. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

[Giannella2015] Chris Giannella. 2015. An improved algorithm for unsupervised decomposition of a multi-author document. *Journal of the Association for Information Science and Technology*.

[Huda et al.2006] Md Shamsul Huda, Ranadhir Ghosh, and John Yearwood. 2006. A variable initialization approach to the em algorithm for better estimation of the parameters of hidden markov model based acoustic modeling of speech signals. In *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, pages 416–430. Springer.

[Juola2006] Patrick Juola. 2006. Authorship attribution. *Foundations and Trends in information Retrieval*, 1(3):233–334.

[Kestemont et al.2011] Mike Kestemont, Kim Luyckx, and Walter Daelemans. 2011. Intrinsic plagiarism detection using character trigram distance scores. *Proceedings of the PAN*.

[Koppel et al.2011] Moshe Koppel, Navot Akiva, Idan Dershowitz, and Nachum Dershowitz. 2011. Unsupervised decomposition of a document into authorial components. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1356–1364. Association for Computational Linguistics.

[Martin and Jurafsky2000] James H Martin and Daniel Jurafsky. 2000. Speech and language processing. *International Edition*.

[McLachlan and Peel2004] Geoffrey McLachlan and David Peel. 2004. *Finite mixture models*. John Wiley & Sons.

[Potha and Stamatatos2014] Nektaria Potha and Efstathios Stamatatos. 2014. A profile-based method for authorship verification. In *Artificial Intelligence: Methods and Applications*, pages 313–326. Springer.

[Rabiner1989] Lawrence R Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rogovschi et al.2010] Nicoleta Rogovschi, Mustapha Lebbah, and Younes Bennani. 2010. Learning self-organizing mixture markov models. *Journal of Nonlinear Systems and Applications*, 1:63–71.

[Savoy2013] Jacques Savoy. 2013. The federalist papers revisited: A collaborative attribution scheme. *Proceedings of the American Society for Information Science and Technology*, 50(1):1–8.

[Savoy2015] Jacques Savoy. 2015. Estimating the probability of an authorship attribution. *Journal of the Association for Information Science and Technology*.

[Stein et al.2011] Benno Stein, Nedim Lipka, and Peter Prettenhofer. 2011. Intrinsic plagiarism analysis. *Language Resources and Evaluation*, 45(1):63–82.

[Wu1983] CF Jeff Wu. 1983. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103.

[Xu and Jordan1996] Lei Xu and Michael I Jordan. 1996. On convergence properties of the em algorithm for gaussian mixtures. *Neural computation*, 8(1):129–151.

Automatic Text Scoring Using Neural Networks

Dimitrios Alikaniotis
Department of Theoretical and Applied Linguistics
University of Cambridge
Cambridge, UK
da352@cam.ac.uk

Helen Yannakoudakis
The ALTA Institute
Computer Laboratory
University of Cambridge
Cambridge, UK
hy260@cl.cam.ac.uk

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
Cambridge, UK
mr472@cl.cam.ac.uk

Abstract

Automated Text Scoring (ATS) provides a cost-effective and consistent alternative to human marking. However, in order to achieve good performance, the predictive features of the system need to be manually engineered by human experts. We introduce a model that forms word representations by learning the extent to which specific words contribute to the text's score. Using Long-Short Term Memory networks to represent the meaning of texts, we demonstrate that a fully automated framework is able to achieve excellent results over similar approaches. In an attempt to make our results more interpretable, and inspired by recent advances in visualizing neural networks, we introduce a novel method for identifying the regions of the text that the model has found more discriminative.

1 Introduction

Automated Text Scoring (ATS) refers to the set of statistical and natural language processing techniques used to automatically score a text on a marking scale. The advantages of ATS systems have been established since Project Essay Grade (PEG) (Page, 1967; Page, 1968), one of the earliest systems whose development was largely motivated by the prospect of reducing labour-intensive marking activities. In addition to providing a cost-effective and efficient approach to large-scale grading of (extended) text, such systems ensure a consistent application of marking criteria, therefore facilitating equity in scoring.

There is a large body of literature with regards to ATS systems of text produced by non-native English-language learners (Page, 1968; Attali and Burstein, 2006; Rudner and Liang, 2002; Elliot, 2003; Landauer et al., 2003; Briscoe et al., 2010; Yannakoudakis et al., 2011; Sakaguchi et al., 2015, among others), overviews of which can be found in various studies (Williamson, 2009; Dikli, 2006; Shermis and Hammer, 2012). Implicitly or explicitly, previous work has primarily treated text scoring as a supervised text classification task, and has utilized a large selection of techniques, ranging from the use of syntactic parsers, via vectorial semantics combined with dimensionality reduction, to generative and discriminative machine learning.

As multiple factors influence the quality of texts, ATS systems typically exploit a large range of textual features that correspond to different properties of text, such as grammar, vocabulary, style, topic relevance, and discourse coherence and cohesion. In addition to lexical and part-of-speech (POS) $n$grams, linguistically deeper features such as types of syntactic constructions, grammatical relations and measures of sentence complexity are among some of the properties that form an ATS system's internal marking criteria. The final representation of a text typically consists of a vector of features that have been manually selected and tuned to predict a score on a marking scale.

Although current approaches to scoring, such as regression and ranking, have been shown to achieve performance that is indistinguishable from that of human examiners, there is substantial manual effort involved in reaching these results on different domains, genres, prompts and so forth. Linguistic features intended to capture the aspects of writing to be assessed are hand-selected and tuned for specific domains. In order to perform well on different data, separate models with distinct feature sets are typically tuned.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 715–725,
Berlin, Germany, August 7-12, 2016.

Prompted by recent advances in deep learning and the ability of such systems to surpass state-of-the-art models in similar areas (Tang, 2015; Tai et al., 2015), we propose the use of recurrent neural network models for ATS. Multi-layer neural networks are known for automatically learning useful features from data, with lower layers learning basic feature detectors and upper levels learning more high-level abstract features (Lee et al., 2009). Additionally, recurrent neural networks are well-suited for modeling the compositionality of language and have been shown to perform very well on the task of language modeling (Mikolov et al., 2011; Chelba et al., 2013). We therefore propose to apply these network structures to the task of scoring, in order to both improve the performance of ATS systems and learn the required feature representations for each dataset automatically, without the need for manual tuning. More specifically, we focus on predicting a holistic score for extended-response writing items.[1]

However, automated models are not a panacea, and their deployment depends largely on the ability to examine their characteristics, whether they measure what is intended to be measured, and whether their internal marking criteria can be interpreted in a meaningful and useful way. The deep architecture of neural network models, however, makes it rather difficult to identify and extract those properties of text that the network has identified as discriminative. Therefore, we also describe a preliminary method for visualizing the information the model is exploiting when assigning a specific score to an input text.

2 Related Work

In this section, we describe a number of the more influential and/or recent approaches in automated text scoring of non-native English-learner writing.

Project Essay Grade (Page, 1967; Page, 1968; Page, 2003) is one of the earliest automated scoring systems, predicting a score using linear regression over vectors of textual features considered to be proxies of writing quality. Intelligent Essay Assessor (Landauer et al., 2003) uses Latent Semantic Analysis to compute the semantic similarity between texts at specific grade points and a test text, which is assigned a score based on the ones in the training set to which it is most similar. Lonsdale and Strong-Krause (2003) use the Link Grammar parser (Sleator and Templerley, 1995) to analyse and score texts based on the average sentence-level scores calculated from the parser's cost vector.

The Bayesian Essay Test Scoring sYstem (Rudner and Liang, 2002) investigates multinomial and Bernoulli Naive Bayes models to classify texts based on shallow content and style features. e-Rater (Attali and Burstein, 2006), developed by the Educational Testing Service, was one of the first systems to be deployed for operational scoring in high-stakes assessments. The model uses a number of different features, including aspects of grammar, vocabulary and style (among others), whose weights are fitted to a marking scheme by regression.

Chen et al. (2010) use a voting algorithm and address text scoring within a weakly supervised bag-of-words framework. Yannakoudakis et al. (2011) extract deep linguistic features and employ a discriminative learning-to-rank model that outperforms regression.

Recently, McNamara et al. (2015) used a hierachical classification approach to scoring, utilizing linguistic, semantic and rhetorical features, among others. Farra et al. (2015) utilize variants of logistic and linear regression and develop models that score persuasive essays based on features extracted from opinion expressions and topical elements.

There have also been attempts to incorporate more diverse features to text scoring models. Klebanov and Flor (2013) demonstrate that essay scoring performance is improved by adding to the model information about percentages of highly associated, mildly associated and dis-associated pairs of words that co-exist in a given text. Somasundaran et al. (2014) exploit lexical chains and their interaction with discourse elements for evaluating the quality of persuasive essays with respect to discourse coherence. Crossley et al. (2015) identify student attributes, such as standardized test scores, as predictive of writing success and use them in conjunction with textual features to develop essay scoring models.

In 2012, Kaggle,[2] sponsored by the Hewlett Foundation, hosted the Automated Student Assessment Prize (ASAP) contest, aiming to demon-

[1] The task is also referred to as Automated Essay Scoring. Throughout this paper, we use the terms *text* and *essay* (scoring) interchangeably.

[2] http://www.kaggle.com/c/asap-aes/

strate the capabilities of automated text scoring systems (Shermis, 2015). The dataset released consists of around twenty thousand texts (60% of which are marked), produced by middle-school English-speaking students, which we use as part of our experiments to develop our models.

3 Models

3.1 C&W Embeddings

Collobert and Weston (2008) and Collobert et al. (2011) introduce a neural network architecture (Fig. 1a) that learns a distributed representation for each word w in a corpus based on its local context. Concretely, suppose we want to learn a representation for some target word w_t found in an n-sized sequence of words $\mathcal{S} = (w_1, \ldots, w_t, \ldots, w_n)$ based on the other words which exist in the same sequence ($\forall w_i \in \mathcal{S} \,|\, w_i \neq w_t$). In order to derive this representation, the model learns to discriminate between $\mathcal{S}$ and some 'noisy' counterpart $\mathcal{S}'$ in which the target word w_t has been substituted for a randomly sampled word from the vocabulary: $\mathcal{S}' = (w_1, \ldots, w_c, \ldots, w_n \,|\, w_c \sim \mathcal{V})$. In this way, every word w is more predictive of its local context than any other random word in the corpus.

Every word in $\mathcal{V}$ is mapped to a real-valued vector in Ω via a mapping function $C(\cdot)$ such that $C(w_i) = \langle \mathbf{M}_{\star i} \rangle$, where $\mathbf{M} \in \mathbb{R}^{D \times |\mathcal{V}|}$ is the embedding matrix and $\langle \mathbf{M}_{\star i} \rangle$ is the ith column of $\mathbf{M}$. The network takes $\mathcal{S}$ as input by concatenating the vectors of the words found in it; $\mathbf{s}_t = \langle C(w_1)^\intercal \| \ldots \| C(w_t)^\intercal \| \ldots \| C(w_n)^\intercal \rangle \in \mathbb{R}^{nD}$. Similarly, $\mathcal{S}'$ is formed by substituting $C(w_t)$ for $C(w_c) \sim \mathbf{M} \,|\, w_c \neq w_t$.

The input vector is then passed through a hard $\tanh$ layer defined as,

$$htanh(x) = \begin{cases} -1 & x < -1 \\ x & -1 \leqslant x \leqslant 1 \\ 1 & x > 1 \end{cases} \quad (1)$$

which feeds a single linear unit in the output layer. The function that is computed by the network is ultimately given by (4):

$$\mathbf{s}_t = \langle \mathbf{M}_{\star 1}^\intercal \| \ldots \| \mathbf{M}_{\star t}^\intercal \| \ldots \| \mathbf{M}_{\star n}^\intercal \rangle^\intercal \quad (2)$$

$$\mathbf{i} = \sigma(\mathbf{W}_{hi}\mathbf{s}_t + \mathbf{b}_h) \quad (3)$$

$$f(\mathbf{s}_t) = \mathbf{W}_{oh}\mathbf{i} + \mathbf{b}_o \quad (4)$$

$$f(\mathbf{s}), \mathbf{b}_o \in \mathbb{R}^1$$
$$\mathbf{W}_{oh} \in \mathbb{R}^{H \times 1}$$
$$\mathbf{W}_{hi} \in \mathbb{R}^{D \times H}$$
$$\mathbf{s} \in \mathbb{R}^D$$
$$\mathbf{b}_o \in \mathbb{R}^H$$

where $\mathbf{M}, \mathbf{W}_{oh}, \mathbf{W}_{hi}, \mathbf{b}_o, \mathbf{b}_h$ are learnable parameters, D, H are hyperparameters controlling the size of the input and the hidden layer, respectively; σ is the application of an element-wise non-linear function ($htanh$ in this case).

The model learns word embeddings by ranking the activation of the true sequence $\mathcal{S}$ higher than the activation of its 'noisy' counterpart $\mathcal{S}'$. The objective of the model then becomes to minimize the hinge loss which ensures that the activations of the original and 'noisy' $n$grams will differ by at least 1:

$$loss_{context}(target,\ corrupt) = [1 - f(\mathbf{s}_t) + f(\mathbf{s}_{ck})]_+,\ \forall k \in \mathbb{Z}^E \quad (5)$$

where E is another hyperparameter controlling the number of 'noisy' sequences we give along with the correct sequence (Mikolov et al., 2013; Gutmann and Hyvärinen, 2012).

3.2 Augmented C&W model

Following Tang (2015), we extend the previous model to capture not only the local linguistic environment of each word, but also how each word contributes to the overall score of the essay. The aim here is to construct representations which, along with the linguistic information given by the linear order of the words in each sentence, are able to capture *usage* information. Words such as *is, are, to, at* which appear with any essay score are considered to be *under*-informative in the sense that they will activate equally both on high and low scoring essays. Informative words, on the other hand, are the ones which would have an impact on the essay score (e.g., spelling mistakes).

In order to capture those *score-specific word embeddings* (SSWEs), we extend (4) by adding a further linear unit in the output layer that performs linear regression, predicting the essay score. Using (2), the activations of the network (presented in Fig. 1b) are given by:

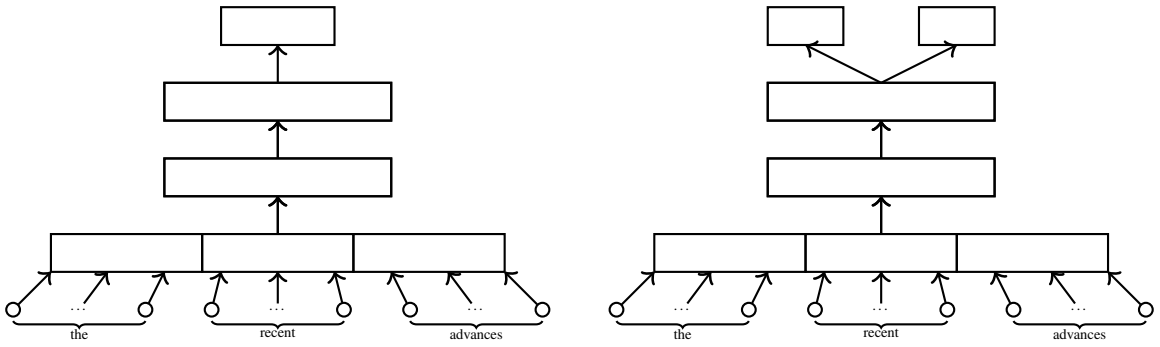


Figure 1: Architecture of the original C&W model (left) and of our extended version (right).

$$f_{ss}(\mathbf{s}) = \mathbf{W}_{oh_1}\mathbf{i} + \mathbf{b}_{o_1} \quad (6)$$

$$f_{context}(\mathbf{s}) = \mathbf{W}_{oh_2}\mathbf{i} + \mathbf{b}_{o_2} \quad (7)$$

$$f_{ss}(\mathbf{s}) \in [\min(score), \max(score)]$$
$$\mathbf{b}_{o_1} \in \mathbb{R}^1$$
$$\mathbf{W}_{oh_1} \in \mathbb{R}^{1\times H}$$

The error we minimize for f_{ss} (where ss stands for *score specific*) is the *mean squared error* between the predicted $\hat{y}$ and the actual essay score y:

$$loss_{score}(\mathbf{s}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \quad (8)$$

From (5) and (8) we compute the overall loss function as a weighted linear combination of the two loss functions (9), back-propagating the error gradients to the embedding matrix $\mathbf{M}$:

$$loss_{overall}(\mathbf{s}) = \begin{array}{c}\alpha \cdot loss_{context}(\mathbf{s}, \mathbf{s}') \\ + (1-\alpha)\cdot loss_{score}(\mathbf{s})\end{array} \quad (9)$$

where α is the hyper-parameter determining how the two error functions should be weighted. α values closer to 0 will place more weight on the score-specific aspect of the embeddings, whereas values closer to 1 will favour the contextual information.

Fig. 2 shows the advantage of using SSWEs in the present setting. Based solely on the information provided by the linguistic environment, words such as *computer* and *laptop* are going to be placed together with their mis-spelled counterparts *copmuter* and *labtop* (Fig. 2a). This, however, does not reflect the fact that the mis-spelled words tend to appear in lower scoring essays. Using SSWEs, the correctly spelled words are pulled apart in the vector space from the incorrectly spelled ones, retaining, however, the information that *labtop* and *copmuter* are still contextually related (Fig. 2b).

3.3 Long-Short Term Memory Network

We use the SSWEs obtained by our model to derive continuous representations for each essay. We treat each essay as a sequence of tokens and explore the use of uni- and bi-directional (Graves, 2012) Long-Short Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) in order to embed these sequences in a vector of fixed size. Both uni- and bi-directional LSTMs have been effectively used for embedding long sequences (Hermann et al., 2015). LSTMs are a kind of recurrent neural network (RNN) architecture in which the output at time t is conditioned on the input $\mathbf{s}$ both at time t and at time $t-1$:

$$\mathbf{y}_t = \mathbf{W}_{yh}\mathbf{h}_t + \mathbf{b}_y \quad (10)$$

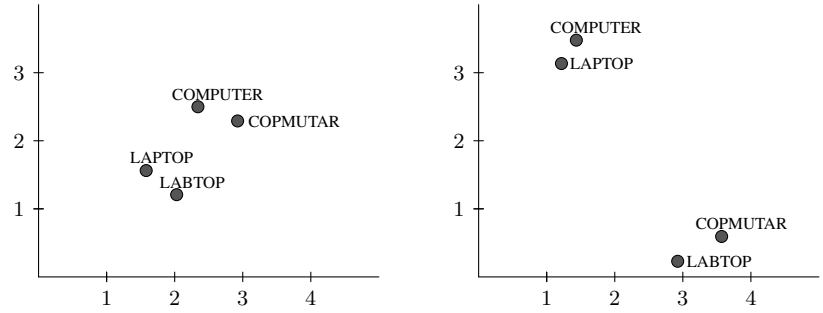
$$\mathbf{h}_t = \mathcal{H}(\mathbf{W}_{hs}\mathbf{s}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \quad (11)$$

where $\mathbf{s}_t$ is the input at time t, and $\mathcal{H}$ is usually an element-wise application of a non-linear function. In LSTMs, $\mathcal{H}$ is substituted for a composite function defining $\mathbf{h}_t$ as:

$$\mathbf{i}_t = \begin{array}{c}\sigma(\mathbf{W}_{is}\mathbf{s}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \\ \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i)\end{array} \quad (12)$$

$$\mathbf{f}_t = \begin{array}{c}\sigma(\mathbf{W}_{fs}\mathbf{s}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \\ \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f)\end{array} \quad (13)$$

$$\mathbf{c}_t = \begin{array}{c}\mathbf{i}_t \odot g(\mathbf{W}_{cs}\mathbf{s}_t + \mathbf{W}_{ch}\mathbf{h}_{t-1} + \mathbf{b}_c) + \\ \mathbf{f}_t \odot \mathbf{c}_{t-1}\end{array} \quad (14)$$



(a) Standard neural embeddings

(b) Score-specific word embeddings

Figure 2: Comparison between standard and score-specific word embeddings. By virtue of appearing in similar environments, standard neural embeddings will place the correct and the incorrect spelling closer in the vector space. However, since the mistakes are found in lower scoring essays, SSWEs are able to discriminate between the correct and the incorrect versions without loss in contextual meaning.

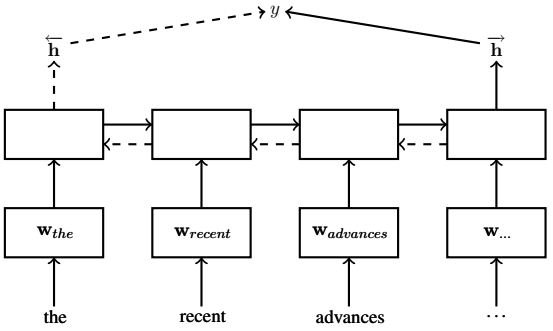


Figure 3: A single-layer *Long Short Term Memory* (LSTM) network. The word vectors $\mathbf{w}_i$ enter the input layer one at a time. The hidden layer that has been formed at the last timestep is used to predict the essay score using linear regression. We also explore the use of bi-directional LSTMs (dashed arrows). For 'deeper' representations, we can stack more LSTM layers after the hidden layer shown here.

$$\mathbf{o}_t = \sigma(\mathbf{W}_{os}\mathbf{s}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o) \quad (15)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot h(\mathbf{c}_t) \quad (16)$$

where g, σ and h are element-wise non-linear functions such as the *logistic sigmoid* ($\frac{1}{1+e^{-x}}$) and the *hyperbolic tangent* ($\frac{e^{2z}-1}{e^{2z}+1}$); $\odot$ is the Hadamard product; $\mathbf{W}, \mathbf{b}$ are the learned weights and biases respectively; and *i*, *f*, *o* and *c* are the input, forget, output gates and the cell activation vectors respectively.

Training the LSTM in a uni-directional manner (i.e., from left to right) might leave out important information about the sentence. For example, our interpretation of a word at some point t_i might be different once we know the word at t_{i+5}. An effective way to get around this issue has been to train the LSTM in a bidirectional manner. This requires doing both a forward and a backward pass of the sequence (i.e., feeding the words from left to right and from right to left). The hidden layer element in (10) can therefore be re-written as the concatenation of the forward and backward hidden vectors:

$$\mathbf{y}_t = \mathbf{W}_{yh} \begin{pmatrix} \overleftarrow{\mathbf{h}}_t^\intercal \\ \overrightarrow{\mathbf{h}}_t^\intercal \end{pmatrix} + \mathbf{b}_y \quad (17)$$

We feed the embedding of each *word* found in each essay to the LSTM one at a time, zero-padding shorter sequences. We form D-dimensional *essay* embeddings by taking the activation of the LSTM layer at the timestep where the last word of the essay was presented to the network. In the case of bi-directional LSTMs, the two independent passes of the essay (from left to right and from right to left) are concatenated together to predict the essay score. These essay embeddings are then fed to a linear unit in the output layer which predicts the essay score (Fig. 3). We use the *mean square error* between the predicted and the gold score as our loss function, and optimize with RMSprop (Dauphin et al., 2015), propagating the errors back to the *word* embeddings.[3]

[3]The maximum time for jointly training a particular SSWE + LSTM combination took about 55–60 hours on an Amazon EC2 g2.2xlarge instance (average time was 27–30 hours).

3.4 Other Baselines

We train a Support Vector Regression model (see Section 4), which is one of the most widely used approaches in text scoring. We parse the data using the RASP parser (Briscoe et al., 2006) and extract a number of different features for assessing the quality of the essays. More specifically, we use character and part-of-speech unigrams, bigrams and trigrams; word unigrams, bigrams and trigrams where we replace open-class words with their POS; and the distribution of common nouns, prepositions, and coordinators. Additionally, we extract and use as features the rules from the phrase-structure tree based on the top parse for each sentence, as well as an estimate of the error rate based on manually-derived error rules.

Ngrams are weighted using *tf–idf*, while the rest are count-based and scaled so that all features have approximately the same order of magnitude. The final input vectors are unit-normalized to account for varying text-length biases.

Further to the above, we also explore the use of the Distributed Memory Model of Paragraph Vectors (PV-DM) proposed by Le and Mikolov (2014), as a means to directly obtain essay embeddings. PV-DM takes as input word vectors which make up $n$gram sequences and uses those to predict the next word in the sequence. A feature of PV-DM, however, is that each 'paragraph' is assigned a unique vector which is used in the prediction. This vector, therefore, acts as a 'memory', retaining information from all contexts that have appeared in this paragraph. Paragraph vectors are then fed to a linear regression model to obtain essay scores (we refer to this model as `doc2vec`).

Additionally, we explore the effect of our score-specific method for learning word embeddings, when compared against three different kinds of word embeddings:

- `word2vec` embeddings (Mikolov et al., 2013) trained on our training set (see Section 4).
- Publicly available `word2vec` embeddings (Mikolov et al., 2013) pre-trained on the Google News corpus (ca. 100 billion words), which have been very effective in capturing solely contextual information.
- Embeddings that are constructed on the fly by the LSTM, by propagating the errors from its hidden layer back to the embedding matrix (i.e., we do not provide any pre-trained word embeddings).[4]

4 Dataset

The Kaggle dataset contains 12.976 essays ranging from 150 to 550 words each, marked by two raters (Cohen's $\kappa = 0.86$). The essays were written by students ranging from Grade 7 to Grade 10, comprising eight distinct sets elicited by eight different prompts, each with distinct marking criteria and score range.[5] For our experiments, we use the resolved combined score between the two raters, which is calculated as the average between the two raters' scores (if the scores are close), or is determined by a third expert (if the scores are far apart). Currently, the state-of-the-art on this dataset has achieved a Cohen's $\kappa = 0.81$ (using quadratic weights). However, the test set was released without the gold score annotations, rendering any comparisons futile, and we are therefore restricted in splitting the given training set to create a new test set.

The sets where divided as follows: 80% of the entire dataset was reserved for training/validation, and 20% for testing. 80% of the training/validation subset was used for actual training, while the remaining 20% for validation (in absolute terms for the entire dataset: 64% training, 16% validation, 20% testing). To facilitate future work, we release the ids of the validation and test set essays we used in our experiments, in addition to our source code and various hyperparameter values.[6]

5 Experiments

5.1 Results

The hyperparameters for our model were as follows: sizes of the layers H, D, the learning rate η, the window size n, the number of 'noisy' sequences E and the weighting factor α. Also the hyperparameters of the LSTM were the size of the LSTM layer D_{LSTM} as well as the dropout rate r.

[4] Another option would be to use standard C&W embeddings; however, this is equivalent to using `SSWE`s with $\alpha = 1$, which we found to produce low results.

[5] Five prompts employed a holistic scoring rubric, one was scored with a two-trait rubric, and two were scored with a multi-trait rubric, but reported as a holistic score (Shermis and Hammer, 2012).

[6] The code, by-model hyperparameter configurations and the IDs of the testing set are available at `https://github.com/dimalik/ats/`.

Model	Spearman's ρ	Pearson r	RMSE	Cohen's κ
doc2vec	0.62	0.63	4.43	0.85
SVM	0.78	0.77	8.85	0.75
LSTM	0.59	0.60	6.8	0.54
BLSTM	0.7	0.5	7.32	0.36
Two-layer LSTM	0.58	0.55	7.16	0.46
Two-layer BLSTM	0.68	0.52	7.31	0.48
word2vec + LSTM	0.68	0.77	5.39	0.76
word2vec + BLSTM	0.75	0.86	4.34	0.85
word2vec + Two-layer LSTM	0.76	0.71	6.02	0.69
word2vec + Two-layer BLSTM	0.78	0.83	4.79	0.82
$\text{word2vec}_{\text{pre-trained}}$ + Two-layer BLSTM	0.79	0.91	3.2	0.92
SSWE + LSTM	0.8	0.94	2.9	0.94
SSWE + BLSTM	0.8	0.92	3.21	0.95
SSWE + Two-layer LSTM	0.82	0.93	3	0.94
SSWE + Two-layer BLSTM	**0.91**	**0.96**	**2.4**	**0.96**

Table 1: Results of the different models on the Kaggle dataset. All resulting vectors were trained using linear regression. We optimized the parameters using a separate validation set (see text) and report the results on the test set.

Since the search space would be massive for grid search, the best hyperparameters were determined using Bayesian Optimization (Snoek et al., 2012). In this context, the performance of our models in the validation set is modeled as a sample from a Gaussian process (GP) by constructing a probabilistic model for the error function and then exploiting this model to make decisions about where to next evaluate the function. The hyperparameters for our baselines were also determined using the same methodology.

All models are trained on our training set (see Section 4), except the one prefixed '$\text{word2vec}_{\text{pre-trained}}$' which uses pre-trained embeddings on the Google News Corpus. We report the Spearman's rank correlation coefficient ρ, Pearson's product-moment correlation coefficient r, and the root mean square error (RMSE) between the predicted scores and the gold standard on our test set, which are considered more appropriate metrics for evaluating essay scoring systems (Yannakoudakis and Cummins, 2015). However, we also report Cohen's κ with quadratic weights, which was the evaluation metric used in the Kaggle competition. Performance of the models is shown in Table 1.

In terms of correlation, SVMs produce competitive results ($\rho = 0.78$ and $r = 0.77$), outperforming doc2vec, LSTM and BLSTM, as well as their deep counterparts. As described above, the SVM model has rich linguistic knowledge and consists of hand-picked features which have achieved excellent performance in similar tasks (Yannakoudakis et al., 2011). However, in terms of RMSE, it is among the lowest performing models (8.85), together with 'BLSTM' and 'Two-layer BLSTM'. Deep models in combination with word2vec (i.e., 'word2vec + Two-layer LSTM' and 'word2vec + Two-layer BLSTM') and SVMs are comparable in terms of r and ρ, though not in terms of RMSE, where the former produce better results, with RMSE improving by half (4.79). doc2vec also produces competitive RMSE results (4.43), though correlation is much lower ($\rho = 0.62$ and $r = 0.63$).

The two BLSTMs trained with word2vec embeddings are among the most competitive models in terms of correlation and outperform all the models, except the ones using pre-trained embeddings and SSWEs. Increasing the number of hidden layers and/or adding bi-directionality does not always improve performance, but it clearly helps in this case and performance improves compared to their uni-directional counterparts.

Using pre-trained word embeddings improves the results further. More specifically, we found '$\text{word2vec}_{\text{pre-trained}}$ + Two-layer BLSTM' to be the best configuration, increasing correlation to 0.79 ρ and 0.91 r, and reducing RMSE to 3.2. We note however that this is not an entirely

fair comparison as these are trained on a much larger corpus than our training set (which we use to train our models). Nevertheless, when we use our SSWEs models we are able to outperform '$\texttt{word2vec}_{\text{pre-trained}}$ + Two-layer BLSTM', even though our embeddings are trained on fewer data points. More specifically, our best model ('SSWE + Two-layer BLSTM') improves correlation to $\rho = 0.91$ and $r = 0.96$, as well as RMSE to 2.4, giving a maximum increase of around 10% in correlation. Given the results of the pre-trained model, we believe that the performance of our best SSWE model will further improve should more training data be given to it.[7]

5.2 Discussion

Our SSWE + LSTM approach having no prior knowledge of the grammar of the language or the domain of the text, is able to score the essays in a very human-like way, outperforming other state-of-the-art systems. Furthermore, while we tuned the models' hyperparameters on a separate validation set, we did not perform any further preprocessing of the text other than simple tokenization.

In the essay scoring literature, text length tends to be a strong predictor of the overall score. In order to investigate any possible effects of essay length, we also calculate the correlation between the gold scores and the length of the essays. We find that the correlations on the test set are relatively low ($r = 0.3, \rho = 0.44$), and therefore conclude that there are no such strong effects.

As described above, we used Bayesian Optimization to find optimal hyperparameter configurations in fewer steps than in regular grid search. Using this approach, the optimization model showed some clear preferences for some parameters which were associated with better scoring models:[8] the number of 'noisy' sequences E, the weighting factor α and the size of the LSTM layer D_{LSTM}. The optimal α value was consistently set to 0.1, which shows that our SSWE approach was necessary to capture the *usage* of the words. Performance dropped considerably as α increased (less weight on SSWEs and more on the contextual aspect). When using $\alpha = 1$, which is equivalent to using the basic C&W model, we found that performance was considerably lower (e.g., correlation dropped to $\rho = 0.15$).

The number of 'noisy' sequences was set to 200, which was the highest possible setting we considered, although this might be related more to the size of the corpus (see Mikolov et al. (2013) for a similar discussion) rather than to our approach. Finally, the optimal value for D_{LSTM} was 10 (the lowest value investigated), which again may be corpus-dependent.

6 Visualizing the black box

In this section, inspired by recent advances in (de-) convolutional neural networks in computer vision (Simonyan et al., 2013) and text summarization (Denil et al., 2014), we introduce a novel method of generating interpretable visualizations of the network's performance. In the present context, this is particularly important as one advantage of the manual methods discussed in § 2 is that we are able to know on what grounds the model made its decisions and which features are most discriminative.

At the outset, our goal is to assess the 'quality' of our word vectors. By 'quality' we mean the level to which a word appearing in a particular context would prove to be problematic for the network's prediction. In order to identify 'high' and 'low' quality vectors, we perform a single pass of an essay from left to right and let the LSTM make its score prediction. Normally, we would provide the gold scores and adjust the network weights based on the error gradients. Instead, we provide the network with a *pseudo-score* by taking the maximum score this specific essay can take[9] and provide this as the 'gold' score. If the word vector is of 'high' quality (i.e., associated with higher scoring texts), then there is going to be little adjustment to the weights in order to predict the highest score possible. Conversely, providing the minimum possible score (here 0), we can assess how 'bad' our word vectors are. Vectors which require minimal adjustment to reach the lowest score are considered of 'lower' quality. Note that since we do a complete pass over the network (without doing any weight updates), the vector quality is going to be *essay dependent*.

[7] Our approach outperforms all the other models in terms of Cohen's κ too.

[8] For the best scoring model the hyperparameters were as follows: $D = 200, H = 100, \eta = 1e-7, n = 9, E = 200, \alpha = 0.1, D_{LSTM} = 10, r = 0.5$.

[9] Note the in the Kaggle dataset essays from different essay sets have different maximum scores. Here we take as $\tilde{y}_{\max}$ the essay set maximum rather than the global maximum.

. . . way to show that Saeng is a determined . . .
. . . sometimes I do . Being patience is being . . .
. . . which leaves the reader satisfied . . .
. . . is in this picture the cyclist is riding a dry and area which could mean that it is very and the looks to be going down hill there looks to be a lot of turns . . .
. . . The only reason im putting this in my own way is because know one is patient in my family . . . .
. . . Whether they are building hand-eye coordination , researching a country , or family and friends through @CAPS3 , @CAPS2 , @CAPS6 the internet is highly and I hope you feel the same way .

Table 2: Several example visualizations created by our LSTM. The full text of the essay is shown in black and the 'quality' of the word vectors appears in color on a range from dark red (low quality) to dark green (high quality).

Concretely, using the network function $f(\mathbf{x})$ as computed by Eq. (12) – (17), we can approximate the loss induced by feeding the *pseudo-scores* by taking the *magnitude* of each error vector (18) – (19). Since $\lim_{\|\mathbf{w}\|_2 \to 0} \hat{y} = y$, this magnitude should tell us how much an embedding needs to change in order to achieve the gold score (here pseudo-score). In the case where we provide the minimum as a *pseudo-score*, a $\|\mathbf{w}\|_2$ value closer to zero would indicate an incorrectly used word. For the results reported here, we combine the magnitudes produced from giving the maximum and minimum *pseudo-scores* into a single score, computed as $L(\tilde{y}_{\max}, f(\mathbf{x})) - L(\tilde{y}_{\min}, f(\mathbf{x}))$, where:

$$L(\tilde{y}, f(\mathbf{x})) \approx \|\mathbf{w}\|_2 \tag{18}$$

$$\mathbf{w} = \nabla L(\mathbf{x}) \triangleq \left.\frac{\partial L}{\partial \mathbf{x}}\right|_{(\tilde{y}, f(\mathbf{x}))} \tag{19}$$

where $\|\mathbf{w}\|_2$ is the vector Euclidean norm $\mathbf{w} = \sqrt{\sum_{i=1}^{N} w_i^2}$; $L(\cdot)$ is the *mean squared error* as in Eq. (8); and $\tilde{y}$ is the essay *pseudo-score*.

We show some examples of this visualization procedure in Table 2. The model is capable of providing positive feedback. Correctly placed punctuation or long-distance dependencies (as in Sentence 6 *are . . . researching*) are particularly favoured by the model. Conversely, the model does not deal well with proper names, but is able to cope with POS mistakes (e.g., *Being patience* or *the internet is highly and . . .*). However, as seen in Sentence 3 the model is not perfect and returns a false negative in the case of *satisfied*.

One potential drawback of this approach is that the gradients are calculated only after the end of the essay. This means that if a word appears multiple times within an essay, sometimes correctly and sometimes incorrectly, the model would not be able to distinguish between them. Two possible solutions to this problem are to either provide the gold score at each timestep which results into a very computationally expensive endeavour, or to feed sentences or phrases of smaller size for which the scoring would be more consistent.[10]

7 Conclusion

In this paper, we introduced a deep neural network model capable of representing both local contextual and usage information as encapsulated by essay scoring. This model yields *score-specific word embeddings* used later by a recurrent neural network in order to form essay representations.

We have shown that this kind of architecture is able to surpass similar state-of-the-art systems, as well as systems based on manual feature engineering which have achieved results close to the upper bound in past work. We also introduced a novel way of exploring the basis of the network's internal scoring criteria, and showed that such models are interpretable and can be further exploited to provide useful feedback to the author.

Acknowledgments

The first author is supported by the Onassis Foundation. We would like to thank the three anonymous reviewers for their valuable feedback.

[10] We note that the same visualization technique can be used to show the 'goodness' of phrases/sentences. Within the phrase setting, after feeding the last word of the phrase to the network, the LSTM layer will contain the phrase embedding. Then, we can assess the 'goodness' of this embedding by evaluating the error gradients after predicting the highest/lowest score.

References

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-Rater v.2.0. *Journal of Technology, Learning, and Assessment*, 4(3):1–30.

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL*, volume 6.

Ted Briscoe, Ben Medlock, and Øistein E. Andersen. 2010. Automated assessment of ESOL free text examinations. Technical Report UCAM-CL-TR-790, University of Cambridge, Computer Laboratory, nov.

Ciprian Chelba, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *arXiv preprint*.

YY Chen, CL Liu, TH Chang, and CH Lee. 2010. An Unsupervised Automated Essay Scoring System. *IEEE Intelligent Systems*, pages 61–67.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. *Proceedings of the Twenty-Fifth international conference on Machine Learning*, pages 160–167, July.

Ronan Collobert, Jason Weston, Leon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. Mar.

Scott Crossley, Laura K Allen, Erica L Snow, and Danielle S McNamara. 2015. Pssst... textual features... there is more to automatic essay scoring than just you! In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pages 203–207. ACM.

Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. Feb.

Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. Jun.

Semire Dikli. 2006. An overview of automated scoring of essays. *Journal of Technology, Learning, and Assessment*, 5(1).

S. Elliot. 2003. Intellimetric™: From here to validity. In M. D. Shermis and J. Burnstein, editors, *Automated Essay Scoring: A Cross-Disciplinary Perspective*, pages 71–86. Lawrence Erlbaum Associates.

Noura Farra, Swapna Somasundaran, and Jill Burstein. 2015. Scoring persuasive essays using opinions and their targets. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 64–74.

Alex Graves. 2012. *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer Berlin Heidelberg.

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13:307–361, February.

Karl Moritz Hermann, Tom Koisk, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. Jun.

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

Beata Beigman Klebanov and Michael Flor. 2013. Word association profiles and their use for automated scoring of essays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1148–1158.

Thomas K. Landauer, Darrell Laham, and Peter W. Foltz. 2003. Automated scoring and annotation of essays with the Intelligent Essay Assessor. In M.D. Shermis and J.C. Burstein, editors, *Automated essay scoring: A cross-disciplinary perspective*, pages 87–112.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. May.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. *Proceedings of the 26th Annual International Conference on Machine Learning ICML 09*.

Deryle Lonsdale and D. Strong-Krause. 2003. Automated rating of ESL essays. In *Proceedings of the HLT-NAACL 2003 Workshop: Building Educational Applications Using Natural Language Processing*.

Danielle S McNamara, Scott A Crossley, Rod D Roscoe, Laura K Allen, and Jianmin Dai. 2015. A hierarchical classification approach to automated essay scoring. *Assessing Writing*, 23:35–59.

Tomáš Mikolov, Stefan Kombrink, Anoop Deoras, Lukáš Burget, and Jan Černocký. 2011. RNNLM-Recurrent neural network language modeling toolkit. In *ASRU 2011 Demo Session*.

Tomas Mikolov, I Sutskever, K Chen, G S Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.

Ellis B. Page. 1967. Grading essays by computer: progress report. In *Proceedings of the Invitational Conference on Testing Problems*, pages 87–100.

Ellis B. Page. 1968. The use of the computer in analyzing student essays. *International Review of Education*, 14(2):210–225, June.

E.B. Page. 2003. Project essay grade: PEG. In M.D. Shermis and J.C. Burstein, editors, *Automated essay scoring: A cross-disciplinary perspective*, pages 43–54.

L.M. Rudner and Tahung Liang. 2002. Automated essay scoring using Bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2):3–21.

Keisuke Sakaguchi, Michael Heilman, and Nitin Madnani. 2015. Effective feature integration for automated short answer scoring. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*.

M Shermis and B Hammer. 2012. Contrasting state-of-the-art automated scoring of essays: analysis. Technical report, The University of Akron and Kaggle.

Mark D Shermis. 2015. Contrasting state-of-the-art in the machine scoring of short-form constructed responses. *Educational Assessment*, 20(1):46–65.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. 12.

D.D.K. Sleator and D. Templerley. 1995. Parsing English with a link grammar. *Proceedings of the 3rd International Workshop on Parsing Technologies, ACL.*

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. Jun.

Swapna Somasundaran, Jill Burstein, and Martin Chodorow. 2014. Lexical chaining for measuring discourse coherence quality in test-taker essays. In *COLING*, pages 950–961.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. Sep.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. Feb.

Duyu Tang. 2015. Sentiment-specific representation learning for document-level sentiment analysis. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining - WSDM '15*. Association for Computing Machinery (ACM).

D. M. Williamson. 2009. A framework for implementing automated scoring. Technical report, Educational Testing Service.

Helen Yannakoudakis and Ronan Cummins. 2015. Evaluating the performance of automated text scoring systems. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics (ACL).

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 180–189.

Improved Semantic Parsers For If-Then Statements

I. Beltagy
The University of Texas at Austin
beltagy@cs.utexas.edu

Chris Quirk
Microsoft Research
chrisq@microsoft.com

Abstract

Digital personal assistants are becoming both more common and more useful. The major NLP challenge for personal assistants is machine understanding: translating natural language user commands into an executable representation. This paper focuses on understanding rules written as If-Then statements, though the techniques should be portable to other semantic parsing tasks. We view understanding as structure prediction and show improved models using both conventional techniques and neural network models. We also discuss various ways to improve generalization and reduce overfitting: synthetic training data from paraphrase, grammar combinations, feature selection and ensembles of multiple systems. An ensemble of these techniques achieves a new state of the art result with 8% accuracy improvement.

1 Introduction

The ability to instruct computers using natural language clearly allows novice users to better use modern information technology. Work in semantic parsing has explored mapping natural language to some formal domain-specific programming languages such as database queries (Woods, 1977; Zelle and Mooney, 1996; Berant et al., 2013; Andreas et al., 2016; Yin et al., 2016), commands to robots (Kate et al., 2005), operating systems (Branavan et al., 2009), and spreadsheets (Gulwani and Marron, 2014). This paper explores the use of neural network models (NN) and conventional models for semantic parsing. Recently approaches using neural networks have shown great improvements in a number of areas such as parsing (Vinyals et al., 2015), machine translation (Devlin et al., 2014), and image captioning (Karpathy and Fei-Fei, 2015). We are among the first to apply neural network methods to semantic parsing tasks (Grefenstette et al., 2014; Dong and Lapata, 2016).

There are several benchmark datasets for semantic parsing, the most well known of which is Geoquery (Zelle and Mooney, 1996). We target an If-Then dataset (Quirk et al., 2015) for several reasons. First, it is both directly applicable to the end-user task of training personal digital assistants. Second, the training data, drawn from the site `http://ifttt.com`, is comparatively quite large, containing nearly 100,000 recipe-description pairs. That said, it is several orders of magnitude smaller than the data for other tasks where neural networks have been successful. Machine translation datasets, for instance, may contain billions of tokens. NN methods appear "data-hungry". They require larger datasets to outperform sparse linear approaches with careful feature engineering, as evidenced in work on syntactic parsing (Vinyals et al., 2015). This makes it interesting to compare NN models with conventional models on this dataset.

As in most prior semantic parsing attempts, we model natural language understanding as a structure prediction problem. Each modeling decision predicts some small component of the target structure, conditioned on the whole input and all prior decisions. Because this is a real-world task, the vocabulary is large and varied, with many words appearing only rarely. Overfitting is a clear danger. We explore several methods to improve generalization. A classic method is to apply feature selection. Synthetic data generated by paraphrasing helps augment the data available. Adjusting the conditional structure of our model also makes sense, as does creating ensembles of the best performing approaches.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 726–736,
Berlin, Germany, August 7-12, 2016.

An ensemble of the resulting systems achieves a new state-of-the-art result, with an absolute improvement of 8% in accuracy. We compare the performance of a neural network model with logistic regression, and explore in detail the contribution of each of them, and why the logistic regression is performing better than the neural network.

2 Related Work

2.1 Semantic Parsing

Semantic parsing is the task of translating natural language to a meaning representation language that the machine can execute. Various semantic parsing tasks have been proposed before, including querying a database (Zelle and Mooney, 1996), following navigation instructions (Chen, 2012), translating to Abstract Meaning Representation (AMR) (Artzi et al., 2015), as well as the If-Then task we explore. Meaning representation languages vary with the task. In database queries, the meaning representation language is either the native query language (e.g. SQL or Prolog), or some alternative that can be deterministically transformed into the native query language. To follow navigation instructions, the meaning representation language is comprised of sequences of valid actions: turn left, turn right, move forward, etc. For parsing If-Then rules, the meaning representation is an abstract syntax tree (AST) in a very simple language. Each root node expands into a "trigger" and "action" pair. These nodes in turn expand into a set of supported triggers and actions. We model these trees as an (almost) context free grammar[1] that generates valid If-Then tasks.

A number of semantic parsing approaches have been proposed, but most fit into the following broad divisions. First, approaches driven by Combinatory Categorical Grammar (CCG) have proven successful at several semantic parsing tasks. This approach is attractive in that it simultaneously provides syntactic and semantic parses of a natural language utterance. Syntactic structure helps constrain and guide semantic interpretation. CCG relies heavily on a *lexicon* that specifies both the syntactic category and formal semantics of each lexical item in the language. In many instantiations, the lexicon is learned from the training data (Zettlemoyer and Collins, 2005) and grounds directly in the meaning representation.

Another approach is to view the semantic parsing task as a machine translation task, where the source language is natural language commands and the target language is the meaning representation. Several approaches have applied standard machine translation techniques to semantic parsing (Wong and Mooney, 2006; Andreas et al., 2013; Ratnaparkhi, 1999) with successful results.

More recently, neural network approaches have been developed for semantic parsing, and especially for querying a database. A neural network is trained to translate the query and the database into some continuous representation then use it to answer the query (Andreas et al., 2016; Yin et al., 2016).

2.2 If-Then dataset

We use a semantic parsing dataset collected from `http://ifttt.com`, first introduced in Quirk et al. (2015). This website publishes a large set of *recipes* in the form of If-Then rules. Each recipe was authored by a website user to automate simple tasks. For instance, a recipe could send you a message every time you are tagged on a picture on Facebook. From a natural language standpoint, the most interesting part of this data is that alongside each recipe, there is a short natural language description intended to name or advertise the task. This provides a naturalistic albeit often noisy source of parallel data for training semantic parsing systems. Some of these descriptions faithfully represent the program. Others are underspecified or suggestive, with many details of the recipe are not uniquely specified or omitted altogether. The task is to predict the correct If-Then code given a natural language description.

As for the code, If-Then statements follow the format

```
If TriggerChannel.
       TriggerFunction (args)
Then ActionChannel.
       ActionFunction (args)
```

Every If-Then statement has exactly one *trigger* and one *action*. Each trigger and action consist of both a *channel* and a *function*. The channel represents a connection to a service, website, or device

[1] Information at the leaves of the action may use parameters drawn from the trigger. For instance, consider a rule that says "text me the daily weather report." The trigger is a new weather report, and the action is to send an SMS. The contents of that SMS are generated by the trigger, which is no longer context free.

(e.g., Facebook, Android, or ESPN) and provides a set of functions relevant to that channel. Finally, each of these functions may take a number of arguments: to receive a trigger when it becomes sunny, we need to specify the location to watch. The resulting dataset after cleaning and separation contains 77,495 training recipes, 5,171 development recipes and 4,294 testing recipes.

2.3 Semantic parsing for If-Then rules

Both CCG and MT-inspired approaches assume a fairly strong correspondence between the words in the natural language request and the concepts in the meaning representation. That is, most words in the description should correspond to some concept in the code, and most concepts in the code should correspond to some word in the description. However, prior work on this dataset (Quirk et al., 2015) found that this strong correspondence is often missing. The descriptions may mention only the most crucial or interesting concepts; the remainder of the meaning representation must be inferred from context. The best performing methods focused primarily on generating well-formed meaning representations, conditioning their decisions on the source language.

Quirk et al. (2015) proposed two models that rely on a grammar to generate all valid ASTs. The first model learns a simple classifier for each production in the grammar, treating the sentence as a bag of features. No alignment between the language and meaning representation is assumed. The second method attempts to learn a correspondence between the language and the code, jointly learning to select the correct productions in the meaning representation grammar. Although the latter approach is more appealing from a modeling standpoint, empirically it doesn't perform substantially better than the alignment-free model. Furthermore the alignment-free model is much simpler to implement and optimize. Therefore, we build upon the alignment-free approach.

2.4 Neural Networks

Neural network approaches have recently made great strides in several natural language processing tasks, including machine translation and dependency parsing. Partially these gains are due to better generalization ability. Until recently, the NLP community leaned heavily on feature-rich approaches that allow models to learn complex relationships from data. However, important features, such as indicator features for words and phrases, were often very sparse. Furthermore, the best systems often relied on manually-induced feature combinations (Bohnet, 2010). Multi-layer neural networks have several advantages. Words (or, more generally, features) are first embedded into a continuous space where similar features land in nearby locations; this helps lead to lexical generalization. The additional hidden layers can model feature interactions in complex ways, obviating the need for manual feature template induction. Feed-forward neural networks with relatively simple structure have shown great gains in both dependency parsing (Chen and Manning, 2014) and machine translation (Devlin et al., 2014) without the need for complex feature templates and large models. Our NN models here are inspired by these effective approaches.

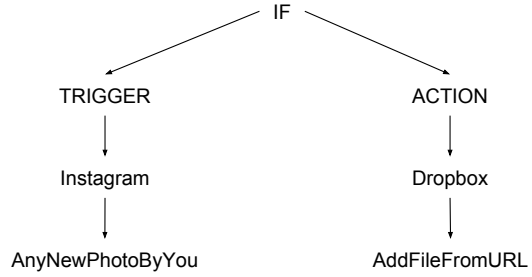


Figure 1: Derivation tree of If-Then statement of the recipe *Autosave your Instagram photos to Dropbox*. Arguments of the functions AnyNewPhotoByYou and AddFileFromURL are ignored.

3 Approach

We next describe the details of how If-Then recipes are constructed given natural language descriptions. As in prior work, we treat semantic parsing as a structure prediction task. First we describe the structure and features of the model, then expand on the details of inference.

3.1 Grammar

Along the lines of Quirk et al. (2015), we build a context-free grammar baseline. This grammar generates only well-formed meaning representations. In the case of this dataset, meaning representations always consist of a root production with two children: a trigger and an action. Both trigger and action first generate a channel, then a function matching that action. Optionally we may also generate the arguments of these functions; we do not

evaluate these selections as they are often idiosyncratic and specific to the user. For example, the recipe *Autosave your Instagram photos to Dropbox* has the following meaning representation:

```
IF Instagram.AnyNewPhotoByYou
THEN Dropbox.AddFileFromURL (
  FileURL={SourceUrl},
  FileName={Caption},
  DropboxFolderPath=IFTTT/Instagram
)
```

If we ignore the function arguments, the resulting meaning representation is:

```
IF Instagram.AnyNewPhotoByYou
THEN Dropbox.AddFileFromURL
```

This examples shows also that most of the function arguments are not crucial for the representation of the If-Then statement.[2]

The grammar we use has productions corresponding to every channel and every function. Figure 1 shows an example derivation tree D. This grammar consists of 892 productions: 128 trigger channels, 487 trigger functions, 99 action channels and 178 action functions.[3]

3.2 Model

Our goal is to learn a model of derivation trees D given a natural sentences S. To predict the derivation for a sentence, we seek the derivation D with maximum probability given the sentence $P(D|S)$.

For the purposes of modeling, we prefer to work with sequences rather than trees. Given a derivation tree D, we transform it into a sequence of productions $R(D) = r_1, \ldots, r_n$ by a top-down, left-to-right tree traversal: r_1 is the top-most production, and r_n is the bottom right production. The sentence S is represented as a set of features $f(S)$.

The derivation score $P(D|S)$ is a function of the productions of D and those features $f(S)$:

$$P(D|S) = \prod_{r_i \in R(D)} P(r_i|r_1, \ldots, r_{i-1}, f(S)) \quad (1)$$

The score of a derivation tree given the sentence is the product of probabilities of its productions.

[2]Arguments are still important for a few If-Then recipes. For instance, in *If there is snow tomorrow send a notification*, "snow" is an argument to the function Tomorrow'sForecastCallsFor. We are not handling such cases in this work.

[3]For this task, it is possible to model the programs as a 4-tuple, but using the grammar approach allows us to port the same technique to other semantic parsing tasks.

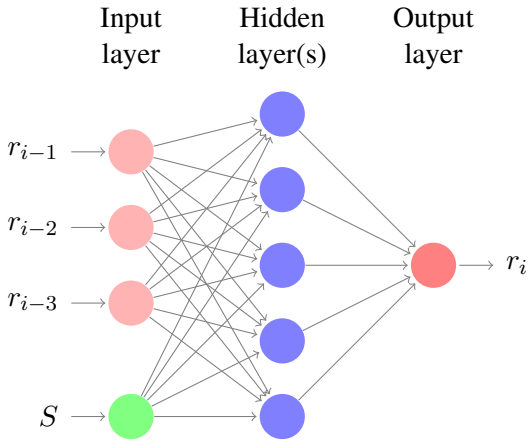


Figure 2: Architecture of the feed-forward neural networks used in this paper. When predicting rule r_i, the prior rules and the whole sentence are used as input. Separate parameters are learned for each position i.

The probability of selecting production r_i given the sentence S is dependent on the features of the sentence as well as the previous productions $r_1, \ldots, r_{i-1}$; namely, all those productions that are above and to the left of the current production. Conditioning on previous productions helps predicting the next one because it captures the conditional dependencies between the productions of the derivation tree, an improvement over prior work (Quirk et al., 2015). In particular, we can model which combinations of triggers and actions are more compatible, both function and channel.

3.3 Training

To learn the derivation score $P(D|S)$, we need to learn probability of productions $P(r_i|r_1, \ldots, r_{i-1}, f(S))$. We learn this probability using a multiclass classifier where the output classes are the possible productions in the grammar. The classifier is trained to predict the next production given previous productions and the sentence features.

Each sentence S is represented with a sparse feature vector $f(S)$. We used a simple set of features: word unigrams and bigrams, character trigrams, and Brown clusters (Liang, 2005). Each sentence is represented as a large sparse k-hot vector, where k is the number of features representing S, $|f(S)|$. We use a simple one-hot representation of prior rules.

For training, we explored two approaches: a standard logistic regression classifier, and a feed

forward neural network classifier. [4] As for network structure, we evaluated models with either one or two 200-dimensional hidden layers (with sigmoid activation function) followed by a softmax output layer to produce a probability for each production. We tried more than two hidden layers and larger hidden layer size, but the results were similar or worse likely because training becomes more difficult. Figure 2 shows the architecture of the network we use. For training, we used a variant of stochastic gradient descent called RMSprop (Dauphin et al., 2015) that adjusts the learning rate for each parameter adaptively, along with a global learning rate of 1^{-3}. The minibatch size was 100, with dropout regularization for hidden layers at 0.5 along with an L2 regularizer with weight 0.005. Each of these parameters were tuned on the validation set, though we found learning to be robust to minor variations in these parameters. All of the neural networks were implemented with Theanets (Johnson, 2015).

Note that history features $r_1, \ldots, r_{i-1}$ in classifier training are always correct. The model is akin to a MEMM, rather than a CRF. We make this simplifying assumption for tractability, like many neural network approaches (Devlin et al., 2014).

3.4 Inference

When, at test time, we are given a new sentence, we would like to infer its most probable derivation tree D. Classifiers trained as in the prior section give probability distributions over productions given the sentence and all prior productions $P(r_i|r_1, \ldots, r_{i-1}, f(S))$. Were the distribution to be context free, we could rely on algorithms similar to Earley parsing (Earley, 1970) to find the max derivation. However, the dependency on prior productions breaks the context free assumption. Therefore, we resort to approximate inference, namely beam search. Each partial hypothesis is grouped into a beam based on the number of productions it contains; we use a beam width of 8, and search for the highest scoring hypothesis.

4 Improving generalization

The data set we use for training and testing is primarily English but contains a broad vocabulary as well as many sentences from other languages such as Chinese, Arabic, and Russian. Thus, a seemingly large dataset of nearly eighty thousand examples is likely to suffer from overfitting. In this section, we discuss a few attempts to improve generalization in the sparse data setting.

4.1 Synthetic data using paraphrases

Arguably the best, though most expensive, way to reduce overfitting is to collect more training data. In our case, the training data available is limited and difficult to create. We propose to augment the training data in an automatic though potentially noisy way by generating synthetic training pairs.

The main idea is that two semantically equivalent sentences should have the same meaning representation. Given an existing training pair, replacing the pair's linguistic description with a paraphrase leads to a new synthetic training pair. For example, a recipe like *Autosave your Instagram **photos** to Dropbox* can be paraphrased to *Autosave your Instagram **pictures** to Dropbox* while retaining the meaning representation:

```
IF Instagram.AnyNewPhotoByYou
THEN Dropbox.AddFileFromURL.
```

We first explore paraphrases using WordNet synonyms. Every word in the sentence can be replaced by one of its synonyms that is picked randomly (a word is a synonym of itself). For words with multiple senses, we group all synonyms of all senses, then retain only those synonyms already in the vocabulary of the training data. This has two advantages. First, we do not increase the vocabulary size and therefore avoid overfitting. Second, this acts as a simple form of word sense disambiguation. This adds around 50,000 additional training examples.

Next, we consider augmenting the data using the Paraphrase Database (Ganitkevitch et al., 2013). Each original description is converted into a lattice. The original word at each position is left in place with a constant score. For each word or phrase in the description found PPDB, we add one arc for each paraphrase, parameterized by the PPDB score of that phrase. The resulting lattice represents many possible paraphrases of the input. We select at most 10 diverse paths through this lattice using the method of Gimpel et al. (2013).[5] This adds around 470,000 training examples.

[4] We tried the sequence-to-sequence model with LSTMs (Sutskever et al., 2014) to map word sequence to the derivation tree productions, but the results were always lower than the feed forward network. This is probably because of the lack of enough training data.

[5] We use a trigram language model, and a weight of 4.

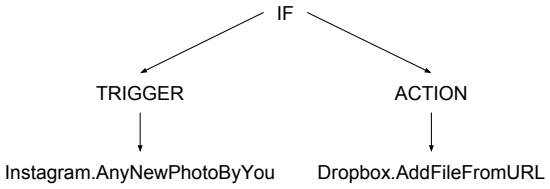


Figure 3: Derivation tree of IFTTT statement of the recipe *Autosave your Instagram photos to Dropbox* using the second grammar.

4.2 Alternative grammar formulation

We rely on a grammar to generate all valid meaning representations and learn models over the productions of this grammar. Different factorizations of the grammar lead to different model distributions. Our primary grammar is described in Section 3.1. A second, alternate grammar formulation has fewer levels but more productions: it combines the channel and function into a single production, in both the trigger and the action. Figure 3 shows an example derivation tree using this grammar. The size of this grammar is 780 productions (552 triggers + 228 actions).

An advantage of this grammar is that it cannot assign probability mass to invalid ASTs, where the function is not applicaable to the channel. On the other hand, this grammar likely does not generalize as well as the first grammar. The first grammar effectively has much more data about each channel, which likely improves accuracy. Function predictions can condition on hopefully accurate channel predictions. It can also benefit from the fact that some function names are shared among channels. From that perspective, the second grammar has fewer training instances for each outcome.

4.3 Feature selection

The training set contains approximately 77K training examples, yet the number of distinct features types (word unigrams and bigrams, character trigrams, Brown clusters) is approximately 230K. Only 80K features occur in the training set more than once. This ratio suggests overfitting may be a major issue.Feature selection likely can improve these issues. We used only simple count cutoffs, including only features that occur in the training set more than once and more than twice. Including features that occur more than once led to improvements in practice.

4.4 Ensemble

Finally, we explore improving generalization by building ensembles of multiple systems. Even if systems overfit, they likely overfit in different ways. When systems agree, they are likely to agree on the correct answer. Combining their results will suffer less from overfitting. We use simple majority voting as an ensemble strategy, resolving ties in an arbitrary but deterministic way.

5 Evaluation

We evaluate the performance of the systems by providing the model with descriptions unseen during training. Free parameters of the models were tuned using the development set. The separation of data into training, development, and test follows Quirk et al. (2015). Two evaluation metrics are used: accuracy on just channel selection and accuracy of both channel and function.

Two major families of approaches are considered: a baseline logistic regression classifier from scikit-learn (Pedregosa et al., 2011), as well as a feed-forward neural network. We explore a number of variations, including feature selection and grammar formulation.

5.1 Comparison systems

Our default system was described in section 3, not including improvements from section 4 unless otherwise noted. The grammar uses the primary formulation from section 3.1. Neural network models use a single hidden layer by default; we also explore two hidden layers.

We evaluate two approaches for generating synthetic data. The first approach, leaning primarily on WordNet to generate up to one paraphrase for each instance, is labeled WN. The second approach using Paraphrase Database to generate up to ten paraphrases is labeled PPDB.

The Alternate grammar line uses the section 4.2 grammar, and otherwise default configurations (no synthetic data, single hidden layer for NN).

Feature selection again uses the default configuration, but uses only those features that occurred more than once in the training data.

Finally we explore ensembles of all approaches. First, we combine all variations within the same model family; next, we bring all systems together. To evaluate the impact of individual systems, we also present results with specific systems removed.

System	Channel accuracy		Full tree accuracy	
	NN	LR	NN	LR
Quirk et al. (2015) w/o alignment	-	46.30	-	33.00
Quirk et al. (2015) with alignment	-	47.40	-	34.50
Default configurations	52.93	53.73	39.66	41.87
Two hidden layers	46.81	-	32.77	-
No hidden layers	50.05	-	38.47	-
Synthetic data (WN)	52.45	53.68	38.64	41.55
Synthetic data (PPDB)	51.86	52.96	38.86	40.63
Alternate grammar	50.09	52.42	39.10	41.15
Feature selection	52.91	53.31	39.29	41.34
Ensemble of systems above	53.98	53.73	41.06	41.85
Ensemble NN + LR	54.31		**42.55**	
Ensemble NN + LR (w/o alternate grammar)	**54.38**		41.90	
Ensemble NN + LR (w/o synthetic data)	53.98		42.41	

Table 1: Accuracy of the Neural Network (NN) and Logistic Regression (LR) implementations of our system with various configurations. Channel-only and full tree (channel+function) accuracies are listed.

5.2 Results

Table 1 shows the accuracy of each evaluated system, and Table 2 explores system performance on important subsets of the data. The first columns present accuracy of just the channel, and the last columns present the channel and the function together (the full derivation). We achieve new state-of-the-art results, showing a 7% absolute improvement on the channel-only accuracy and 8% absolute improvement on the full derivation tree in the most difficult condition.

5.3 Discussion

Partly these improved results are driven by better features. Adding more robust representations of the input (e.g. Brown clusters) and conditioning on prior structure of the tree leads to more consistent and coherent trees.

One key observation is that the logistic regression classifier consistently outperforms the neural network, though by a small margin. We suspect two main causes: optimization difficulties and training size. To compare the optimization algorithms, Table 1 shows the result of a neural network with no hidden layers, which is effectively identical to a logistic regression model. Stochastic gradient descent used to train the neural network did not perform as well as the LIBLINEAR (Fan et al., 2008) solver used to train the logistic regression, because the loss function was not optimized as well. Optimization problems are even more likely with hidden layers, since the objective is no longer convex.

Second, the training data is small by neural network standards. Prior attempts to use neural networks for parsing required larger amounts of training data to exceed the state-of-the-art. Non-linear models are able to capture regularities that linear models cannot, but may require more training data to do so. Table 1 shows that a network with a single hidden layer outperforms a one with two hidden layers. This additional hidden layer seems to make learning harder (even with layer-wise pretraining). We also ran an additional experiment, limiting both NN and LR to use word unigram features, and varying the vocabulary size by frequency thresholding; the results are in table 3. LR models were more effective when all features were present, likely due to their convex objective and simple regularization. NN models, on the other hand, actually outperform LR models when limited to more common vocabulary items. Given more data, NN could likely find representations that outperformed manual feature engineering.

Although we only considered feed-forward nerual networks, results on recurrent architectures Dong and Lapata (2016) are in accordance with our findings. Their LSTM-based approach does not achieve great gains on this data set because: "*user curated descriptions are often of low quality, and thus align very loosely to their corresponding ASTs*". Even though this training set is larger than other semantic parsing datasets, the vocabulary, sentence structures, and even languages here are much more diverse, which make it difficult for the NN to learn useful representations. Dong and Lapata (2016) tried to reduce the impact of this problem by evaluating only on the English sub-

	Channel	Full tree
All: 4,294 recipes		
posclass	47.4	34.5
D&L	—	—
NN	52.9	39.7
LR	53.7	41.9
Ensemble	**54.3**	**42.6**
oracleturk	48.8	37.8
Omit non-English: 3,744 recipes		
posclass	50.0	36.9
D&L	54.3	39.2
NN	55.1	41.2
LR	56.0	44.3
Ensemble	**56.8**	**44.5**
oracleturk	56.0	43.5
Omit non-English, unintelligible: 2,433 recipes		
posclass	67.2	50.4
D&L	68.8	50.5
NN	71.3	53.7
LR	71.9	56.6
Ensemble	**72.7**	**57.1**
oracleturk	86.2	59.4
≥3 agree with gold: 760 recipes		
posclass	81.4	71.0
D&L	87.8	75.2
NN	88.0	74.3
LR	88.8	**82.5**
Ensemble	**89.1**	82.2
oracleturk	100.0	100.0

Table 2: System comparisons on various subsets of the data. Following Quirk et al. (2015), we also evaluation on illustrative subsets. "posclass" represents the best system from prior work. D&L is the best-performing system from Dong and Lapata (2016). NN and LR are the single best neural network, logistic regression models, and Ensemble is the combination of all systems. "oracleturk" represents cases where at least one turker agreed with the gold standard.

set of the data. Interestingly, our carefully built feed-forward networks outperform their approach in almost every subset.

Although the neural network with one hidden layer does not outperform logistic regression in a feature rich setting, it makes substantially different predictions. An ensemble of their outputs achieves better accuracy than either system individually.

Our techniques for improving generalization do not improve individual systems. Yet when all techniques are combined in an ensemble, the resulting predictions are better. Furthermore, an ensemble without the synthetic data or without the alternate grammar has lower accuracy: each technique contributes to the final result.

System	Full tree accuracy	
	NN	LR
All words	35.79	37.03
Count ≥ 2	37.01	36.91
Count ≥ 3	37.07	36.59

Table 3: Accuracy of NN and LR limited to word unigram features, with three vocabulary sizes: all words, words occurring at least twice in the training data (13,971 words), and those occurring at least three times in the training data (8,974 words).

5.4 Comparison of logistic regression and neural network approaches

We performed a detailed exploration of the cases where either the LR model was correct and the NN model was wrong, or vice versa. Table 4 breaks these errors into a number of cases:

- **Swapped trigger and action**. Here the system misinterpreted a rule, swapping the trigger for the action. An example NN swap was "Backup Pinboard entries to diigo"; an example LR swap was "Like a photo on tumblr and upload it to your flickr photostream ."
- **Duplicated**. In this case, the system used the same channel for both trigger and action, despite clear evidence in the language. For instance, the LR model incorrectly used Facebook as both the trigger and channel in this recipe: "New photo on Facebook addec to my Pryv". The NN model correctly identified Pryv as the target channel, despite the typo in the recipe.
- **Missed word cue**. In many cases there was a clear "cue word" in the language that should have forced a correct channel, but the model picked the wrong one. For instance, in "tweet # stared youtube video", the trigger should be starred YouTube videos, but the NN model incorrectly selected feeds.
- **Missed multi-word cue**. Sometimes the cue was a multi-word phrase, such as "One Drive". The NN model tended to miss these cues.
- **Missed inference**. In certain cases the cue was more of a loose inference. Words such as "payment" and "refund" should tend to

Error type	NN errors	LR errors
Swapped trigger and action	4	4
Duplicated	3	4
Missed word cue	8	8
Missed multi-word cue	2	0
Missed inference	8	0
Related channel	5	8
Grand Total	30	24

Table 4: Count of error cases by type for NN and LR models, in their default configurations. This table only counts those instances in the most clean set (where three or more turkers agree with the gold program) where exactly one system made an error.

refer to triggers from the Square payment provider; the NN seemed to struggle on these cases.

- **Related channel**. Often the true channel is very difficult to pick: should the system use iOS location or Android location? NN models seemed to do better on these cases, perhaps picking up on some latent cues in the data that were not immediately evident to the authors.

In general, a slightly more powerful NN model with access to more relevant data might overcome some of the issues above.

We also explored correlations with errors and a number of other criteria, such as text length and frequency of the channels and functions, but found no substantial differences. In general, the remaining errors are often plausible given the noisy input.

6 Future Work

We have achieved a new state-of-the-art on this dataset, though derivation tree accuracy remains low, around 42%. While some errors are caused by training data noise and others are due to noisy test instances, there is still room for improvement.

We believe synthetic data is a promising direction. Initial attempts show small improvements; better results may be within reach given more tuning. This may enable gains with recurrent architectures (e.g., LSTMs).

The networks here rely primarily on word-based features. Character-based models have resulted in improved syntactic parsing results (Ballesteros et al., 2015). We believe that noisy data such as the If-Then corpus would benefit from character modelings, since the models could be more robust to spelling errors and variations.

Another important future work direction is to model the arguments of the If-Then statements. However, that requires segmenting the arguments into those that are general across all users, and those that are specific to the recipe's author. Likely this would require further annotation of the data.

7 Conclusion

In this paper, we address a semantic parsing task, namely translating sentences to If-Then statements. We model the task as structure prediction, and show improved models using both neural networks and logistic regression. We also discussed various ways to improve generalization and reduce overfitting, including adding synthetic training data by paraphrasing sentences, using multiple grammars, applying feature selection and ensembling multiple systems. We achieve a new state-of-the-art with 8% absolute accuracy improvement.

References

Jacob Andreas, Andreas Vlachos, and Stephen Clark. 2013. Semantic parsing as machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 47–52, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Learning to compose neural networks for question answering. In *NAACL 2016*.

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG semantic parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal, September. Association for Computational Linguistics.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-13)*.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*, Singapore.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP-14)*.

David L Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 430–439. Association for Computational Linguistics.

Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. 2015. RMSProp and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1370–1380, Baltimore, Maryland, June. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *arXiv:1601.01280*.

Jay Earley. 1970. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, February.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

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

Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. 2013. A systematic exploration of diversity in machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1100–1111, Seattle, Washington, USA, October. Association for Computational Linguistics.

Edward Grefenstette, Phil Blunsom, Nando de Freitas, and Karl Moritz Hermann. 2014. A deep architecture for semantic parsing. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing*, pages 22–27, Baltimore, MD, June. Association for Computational Linguistics.

Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *SIGMOD*.

Leif Johnson. 2015. Theanets. `https://github.com/lmjohns3/theanets`.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pages 1062–1068, Pittsburgh, PA, July.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, Massachusetts Institute of Technology.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Chris Quirk, Raymond Mooney, and Michel Galley. 2015. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine learning*, 34(1-3):151–175.

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

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, R. Garnett, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2755–2763. Curran Associates, Inc.

Yuk Wah Wong and Raymond Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 439–446, New York City, USA, June. Association for Computational Linguistics.

William A. Woods. 1977. Lunar rocks in natural English: Explorations in natural language question answering. In Antonio Zampoli, editor, *Linguistic Structures Processing*. Elsevier North-Holland, New York.

Pengcheng Yin, Zhengdong Lu, and Ben Kao Hang Li. 2016. Neural enquirer: Learning to query tables with natural language. In *ICLR 2016*.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1050–1055, Portland, OR, August.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *In Proceedings of the 21st Conference on Uncertainty in AI*, pages 658–666.

Universal Dependencies for Learner English

Yevgeni Berzak
CSAIL MIT
berzak@mit.edu

Jessica Kenney
EECS & Linguistics MIT
jessk@mit.edu

Carolyn Spadine
Linguistics MIT
cspadine@mit.edu

Jing Xian Wang
EECS MIT
jxwang@mit.edu

Lucia Lam
MECHE MIT
lucci@mit.edu

Keiko Sophie Mori
Linguistics MIT
ksmori@mit.edu

Sebastian Garza
Linguistics MIT
sjgarza@mit.edu

Boris Katz
CSAIL MIT
boris@mit.edu

Abstract

We introduce the Treebank of Learner English (TLE), the first publicly available syntactic treebank for English as a Second Language (ESL). The TLE provides manually annotated POS tags and Universal Dependency (UD) trees for 5,124 sentences from the Cambridge First Certificate in English (FCE) corpus. The UD annotations are tied to a pre-existing error annotation of the FCE, whereby full syntactic analyses are provided for both the original and error corrected versions of each sentence. Further on, we delineate ESL annotation guidelines that allow for consistent syntactic treatment of ungrammatical English. Finally, we benchmark POS tagging and dependency parsing performance on the TLE dataset and measure the effect of grammatical errors on parsing accuracy. We envision the treebank to support a wide range of linguistic and computational research on second language acquisition as well as automatic processing of ungrammatical language[1].

1 Introduction

The majority of the English text available worldwide is generated by non-native speakers (Crystal, 2003). Such texts introduce a variety of challenges, most notably grammatical errors, and are of paramount importance for the scientific study of language acquisition as well as for NLP. Despite the ubiquity of non-native English, there is currently no publicly available syntactic treebank for English as a Second Language (ESL).

To address this shortcoming, we present the Treebank of Learner English (TLE), a first of its kind resource for non-native English, containing 5,124 sentences manually annotated with POS tags and dependency trees. The TLE sentences are drawn from the FCE dataset (Yannakoudakis et al., 2011), and authored by English learners from 10 different native language backgrounds. The treebank uses the Universal Dependencies (UD) formalism (De Marneffe et al., 2014; Nivre et al., 2016), which provides a unified annotation framework across different languages and is geared towards multilingual NLP (McDonald et al., 2013). This characteristic allows our treebank to support computational analysis of ESL using not only English based but also multilingual approaches which seek to relate ESL phenomena to native language syntax.

While the annotation inventory and guidelines are defined by the English UD formalism, we build on previous work in learner language analysis (Dıaz-Negrillo et al., 2010; Dickinson and Ragheb, 2013) to formulate an additional set of annotation conventions aiming at a uniform treatment of ungrammatical learner language. Our annotation scheme uses a two-layer analysis, whereby a distinct syntactic annotation is provided for the *original* and the *corrected* version of each sentence. This approach is enabled by a pre-existing error annotation of the FCE (Nicholls, 2003) which is used to generate an error corrected variant of the dataset. Our inter-annotator agreement results provide evidence for the ability of the annotation scheme to support consistent annotation of ungrammatical structures.

[1]The treebank is available at universaldependencies.org. The annotation manual used in this project and a graphical query engine are available at esltreebank.org.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 737–746,
Berlin, Germany, August 7-12, 2016.

Finally, a corpus that is annotated with both grammatical errors and syntactic dependencies paves the way for empirical investigation of the relation between grammaticality and syntax. Understanding this relation is vital for improving tagging and parsing performance on learner language (Geertzen et al., 2013), syntax based grammatical error correction (Tetreault et al., 2010; Ng et al., 2014), and many other fundamental challenges in NLP. In this work, we take the first step in this direction by benchmarking tagging and parsing accuracy on our dataset under different training regimes, and obtaining several estimates for the impact of grammatical errors on these tasks.

To summarize, this paper presents three contributions. First, we introduce the first large scale syntactic treebank for ESL, manually annotated with POS tags and universal dependencies. Second, we describe a linguistically motivated annotation scheme for ungrammatical learner English and provide empirical support for its consistency via inter-annotator agreement analysis. Third, we benchmark a state of the art parser on our dataset and estimate the influence of grammatical errors on the accuracy of automatic POS tagging and dependency parsing.

The remainder of this paper is structured as follows. We start by presenting an overview of the treebank in section 2. In sections 3 and 4 we provide background information on the annotation project, and review the main annotation stages leading to the current form of the dataset. The ESL annotation guidelines are summarized in section 5. Inter-annotator agreement analysis is presented in section 6, followed by parsing experiments in section 7. Finally, we review related work in section 8 and present the conclusion in section 9.

2 Treebank Overview

The TLE currently contains 5,124 sentences (97,681 tokens) with POS tag and dependency annotations in the English Universal Dependencies (UD) formalism (De Marneffe et al., 2014; Nivre et al., 2016). The sentences were obtained from the FCE corpus (Yannakoudakis et al., 2011), a collection of upper intermediate English learner essays, containing error annotations with 75 error categories (Nicholls, 2003). Sentence level segmentation was performed using an adaptation of the NLTK sentence tokenizer[2]. Under-segmented sentences were split further manually. Word level tokenization was generated using the Stanford PTB word tokenizer[3].

The treebank represents learners with 10 different native language backgrounds: Chinese, French, German, Italian, Japanese, Korean, Portuguese, Spanish, Russian and Turkish. For every native language, we randomly sampled 500 automatically segmented sentences, under the constraint that selected sentences have to contain at least one grammatical error that is not punctuation or spelling.

The TLE annotations are provided in two versions. The first version is the *original sentence* authored by the learner, containing grammatical errors. The second, *corrected sentence* version, is a grammatical variant of the original sentence, generated by correcting all the grammatical errors in the sentence according to the manual error annotation provided in the FCE dataset. The resulting corrected sentences constitute a parallel corpus of standard English. Table 1 presents basic statistics of both versions of the annotated sentences.

	original	corrected
sentences	5,124	5,124
tokens	97,681	98,976
sentence length	19.06 (std 9.47)	19.32 (std 9.59)
errors per sentence	2.67 (std 1.9)	-
authors	924	
native languages	10	

Table 1: Statistics of the TLE. Standard deviations are denoted in parenthesis.

To avoid potential annotation biases, the annotations of the treebank were created manually *from scratch*, without utilizing any automatic annotation tools. To further assure annotation quality, each annotated sentence was reviewed by two additional annotators. To the best of our knowledge, TLE is the first large scale English treebank constructed in a completely manual fashion.

3 Annotator Training

The treebank was annotated by six students, five undergraduates and one graduate. Among the undergraduates, three are linguistics majors and two are engineering majors with a linguistic minor. The graduate student is a linguist specializing in syntax. An additional graduate student in NLP participated in the final debugging of the dataset.

[2]http://www.nltk.org/api/nltk.tokenize.html

[3]http://nlp.stanford.edu/software/tokenizer.shtml

Prior to annotating the treebank sentences, the annotators were trained for about 8 weeks. During the training, the annotators attended tutorials on dependency grammars, and learned the English UD guidelines[4], the Penn Treebank POS guidelines (Santorini, 1990), the grammatical error annotation scheme of the FCE (Nicholls, 2003), as well as the ESL guidelines described in section 5 and in the annotation manual.

Furthermore, the annotators completed six annotation exercises, in which they were required to annotate POS tags and dependencies for practice sentences from scratch. The exercises were done individually, and were followed by group meetings in which annotation disagreements were discussed and resolved. Each of the first three exercises consisted of 20 sentences from the UD gold standard for English, the English Web Treebank (EWT) (Silveira et al., 2014). The remaining three exercises contained 20-30 ESL sentences from the FCE. Many of the ESL guidelines were introduced or refined based on the disagreements in the ESL practice exercises and the subsequent group discussions. Several additional guidelines were introduced in the course of the annotation process.

During the training period, the annotators also learned to use a search tool that enables formulating queries over word and POS tag sequences as regular expressions and obtaining their annotation statistics in the EWT. After experimenting with both textual and graphical interfaces for performing the annotations, we converged on a simple text based format described in section 4.1, where the annotations were filled in using a spreadsheet or a text editor, and tested with a script for detecting annotation typos. The annotators continued to meet and discuss annotation issues on a weekly basis throughout the entire duration of the project.

4 Annotation Procedure

The formation of the treebank was carried out in four steps: annotation, review, disagreement resolution and targeted debugging.

4.1 Annotation

In the first stage, the annotators were given sentences for annotation from scratch. We use a CoNLL based textual template in which each word is annotated in a separate line. Each line contains 6 columns, the first of which has the word index (IND) and the second the word itself (WORD). The remaining four columns had to be filled in with a Universal POS tag (UPOS), a Penn Treebank POS tag (POS), a head word index (HIND) and a dependency relation (REL) according to version 1 of the English UD guidelines.

The annotation section of the sentence is preceded by a metadata header. The first field in this header, denoted with SENT, contains the FCE error coded version of the sentence. The annotators were instructed to verify the error annotation, and add new error annotations if needed. Corrections to the sentence segmentation are specified in the SEGMENT field[5]. Further down, the field TYPO is designated for literal annotation of spelling errors and ill formed words that happen to form valid words (see section 5.2).

The example below presents a pre-annotated original sentence given to an annotator.

```
#SENT=That time I had to sleep in <ns type=
"MD"><c>a</c></ns> tent.
#SEGMENT=
#TYPO=


#IND   WORD    UPOS    POS     HIND    REL
1      That
2      time
3      I
4      had
5      to
6      sleep
7      in
8      tent
9      .
```

Upon completion of the original sentence, the annotators proceeded to annotate the corrected sentence version. To reduce annotation time, annotators used a script that copies over annotations from the original sentence and updates head indices of tokens that appear in both sentence versions. Head indices and relation labels were filled in only if the head word of the token appeared in both the original and corrected sentence versions. Tokens with automatically filled annotations included an additional # sign in a seventh column of each word's annotation. The # signs had to be removed, and the corresponding annotations either approved or changed as appropriate. Tokens that did not appear in the original sentence version were annotated from scratch.

[4]http://universaldependencies.org/#en

[5]The released version of the treebank splits the sentences according to the markings in the SEGMENT field when those apply both to the original and corrected versions of the sentence. Resulting segments without grammatical errors in the original version are currently discarded.

4.2 Review

All annotated sentences were randomly assigned to a second annotator (henceforth *reviewer*), in a double blind manner. The reviewer's task was to mark all the annotations that they would have annotated differently. To assist the review process, we compiled a list of common annotation errors, available in the released annotation manual.

The annotations were reviewed using an *active* editing scheme in which an explicit action was required for all the existing annotations. The scheme was introduced to prevent reviewers from overlooking annotation issues due to passive approval. Specifically, an additional # sign was added at the seventh column of each token's annotation. The reviewer then had to either "sign off" on the existing annotation by erasing the # sign, or provide an alternative annotation following the # sign.

4.3 Disagreement Resolution

In the final stage of the annotation process all annotator-reviewer disagreements were resolved by a third annotator (henceforth *judge*), whose main task was to decide in favor of the annotator or the reviewer. Similarly to the review process, the judging task was carried out in a double blind manner. Judges were allowed to resolve annotator-reviewer disagreements with a third alternative, as well as introduce new corrections for annotation issues overlooked by the reviewers.

Another task performed by the judges was to mark acceptable *alternative annotations* for ambiguous structures determined through review disagreements or otherwise present in the sentence. These annotations were specified in an additional metadata field called AMBIGUITY. The ambiguity markings are provided along with the resolved version of the annotations.

4.4 Final Debugging

After applying the resolutions produced by the judges, we queried the corpus with debugging tests for specific linguistics constructions. This additional testing phase further reduced the number of annotation errors and inconsistencies in the treebank. Including the training period, the treebank creation lasted over a year, with an aggregate of more than 2,000 annotation hours.

5 Annotation Scheme for ESL

Our annotations use the existing inventory of English UD POS tags and dependency relations, and follow the standard UD annotation guidelines for English. However, these guidelines were formulated with grammatical usage of English in mind and do not cover non canonical syntactic structures arising due to grammatical errors[6]. To encourage consistent and linguistically motivated annotation of such structures, we formulated a complementary set of ESL annotation guidelines.

Our ESL annotation guidelines follow the general principle of *literal reading*, which emphasizes syntactic analysis according to the observed language usage. This strategy continues a line of work in SLA which advocates for centering analysis of learner language around morpho-syntactic surface evidence (Ragheb and Dickinson, 2012; Dickinson and Ragheb, 2013). Similarly to our framework, which includes a parallel annotation of corrected sentences, such strategies are often presented in the context of multi-layer annotation schemes that also account for error corrected sentence forms (Hirschmann et al., 2007; Dıaz-Negrillo et al., 2010; Rosen et al., 2014).

Deploying a strategy of literal annotation within UD, a formalism which enforces cross-linguistic consistency of annotations, will enable meaningful comparisons between non-canonical structures in English and canonical structures in the author's native language. As a result, a key novel characteristic of our treebank is its ability to support cross-lingual studies of learner language.

5.1 Literal Annotation

With respect to POS tagging, literal annotation implies adhering as much as possible to the observed morphological forms of the words. Syntactically, argument structure is annotated according to the usage of the word rather than its typical distribution in the relevant context. The following list of conventions defines the notion of literal reading for some of the common non canonical structures associated with grammatical errors.

Argument Structure

Extraneous prepositions We annotate all nominal dependents introduced by extraneous prepositions

[6]The English UD guidelines do address several issues encountered in informal genres, such as the relation "goeswith", which is used for fragmented words resulting from typos.

as nominal modifiers. In the following sentence, "him" is marked as a nominal modifier (*nmod*) instead of an indirect object (*iobj*) of "give".

```
#SENT=...I had to give <ns type="UT"><i>to</i> </ns>
him water...
...
21      I          PRON      PRP      22      nsubj
22      had        VERB      VBD      5       parataxis
23      to         PART      TO       24      mark
24      give       VERB      VB       22      xcomp
25      to         ADP       IN       26      case
26      him        PRON      PRP      24      nmod
27      water      NOUN      NN       24      dobj
...
```

Omitted prepositions We treat nominal dependents of a predicate that are lacking a preposition as arguments rather than nominal modifiers. In the example below, "money" is marked as a direct object (*dobj*) instead of a nominal modifier (*nmod*) of "ask". As "you" functions in this context as a second argument of "ask", it is annotated as an indirect object (*iobj*) instead of a direct object (*dobj*).

```
#SENT=...I have to ask you <ns type="MT">
<c>for</c></ns> the money <ns type= "RT">
<i>of</i><c>for</c></ns> the tickets back.
...
12      I          PRON      PRP      13      nsubj
13      have       VERB      VBP      2       conj
14      to         PART      TO       15      mark
15      ask        VERB      VB       13      xcomp
16      you        PRON      PRP      15      iobj
17      the        DET       DT       18      det
18      money      NOUN      NN       15      dobj
19      of         ADP       IN       21      case
20      the        DET       DT       21      det
21      tickets    NOUN      NNS      18      nmod
22      back       ADV       RB       15      advmod
23      .          PUNCT     .        2       punct
```

Tense

Cases of erroneous tense usage are annotated according to the morphological tense of the verb. For example, below we annotate "shopping" with present participle VBG, while the correction "shop" is annotated in the corrected version of the sentence as VBP.

```
#SENT=...when you <ns type="TV"><i>shopping</i>
<c>shop</c></ns>...
...
4       when       ADV       WRB      6       advmod
5       you        PRON      PRP      6       nsubj
6       shopping   VERB      VBG      12      advcl
...
```

Word Formation

Erroneous word formations that are contextually plausible and can be assigned with a PTB tag are annotated literally. In the following example, "stuffs" is handled as a plural count noun.

```
#SENT=...into fashionable <ns type="CN">
<i>stuffs</i><c>stuff</c></ns>...
...
7       into       ADP       IN       9       case
8       fashionable ADJ      JJ       9       amod
9       stuffs     NOUN      NNS      2       ccomp
...
```

Similarly, in the example below we annotate "necessaryiest" as a superlative.

```
#SENT=The necessaryiest things...
1       The            DET       DT       3       det
2       necessaryiest  ADJ       JJS      3       amod
3       things         NOUN      NNS      0       root
...
```

5.2 Exceptions to Literal Annotation

Although our general annotation strategy for ESL follows literal sentence readings, several types of word formation errors make such readings uninformative or impossible, essentially forcing certain words to be annotated using some degree of interpretation (Rosén and De Smedt, 2010). We hence annotate the following cases in the original sentence according to an interpretation of an intended word meaning, obtained from the FCE error correction.

Spelling

Spelling errors are annotated according to the correctly spelled version of the word. To support error analysis of automatic annotation tools, misspelled words that happen to form valid words are annotated in the metadata field TYPO for POS tags with respect to the most common usage of the misspelled word form. In the example below, the TYPO field contains the typical POS annotation of "where", which is clearly unintended in the context of the sentence.

```
#SENT=...we <ns type="SX"><i>where</i>
<c>were</c></ns> invited to visit...
#TYPO=5 ADV WRB
...
4       we         PRON      PRP      6       nsubjpass
5       where      AUX       VBD      6       auxpass
6       invited    VERB      VBN      0       root
7       to         PART      TO       8       mark
8       visit      VERB      VB       6       xcomp
...
```

Word Formation

Erroneous word formations that cannot be assigned with an existing PTB tag are annotated with respect to the correct word form.

```
#SENT=I am <ns type="IV"><i>writting</i>
<c>writing</c></ns>...
1       I          PRON      PRP      3       nsubj
2       am         AUX       VBP      3       aux
3       writting   VERB      VBG      0       root
...
```

In particular, ill formed adjectives that have a plural suffix receive a standard adjectival POS tag. When applicable, such cases also receive an additional marking for unnecessary agreement in the error annotation using the attribute "ua".

```
#SENT=...<ns type="IJ" ua=true>
<i>interestings</i><c>interesting</c></ns> things...
```

```
...
6       interestings  ADJ     JJ      7       amod
7       things        NOUN    NNS     3 dobj
...
```

Wrong word formations that result in a valid, but contextually implausible word form are also annotated according to the word correction. In the example below, the nominal form "sale" is likely to be an unintended result of an ill formed verb. Similarly to spelling errors that result in valid words, we mark the typical literal POS annotation in the TYPO metadata field.

```
#SENT=...they do not <ns type="DV"><i>sale</i>
<c>sell</c></ns> them...
#TYPO=15 NOUN NN

...
12      they        PRON    PRP     15      nsubj
13      do          AUX     VBP     15      aux
14      not         PART    RB      15      neg
15      sale        VERB    VB      0       root
16      them        PRON    PRP     15      dobj
...
```

Taken together, our ESL conventions cover many of the annotation challenges related to grammatical errors present in the TLE. In addition to the presented overview, the complete manual of ESL guidelines used by the annotators is publicly available. The manual contains further details on our annotation scheme, additional annotation guidelines and a list of common annotation errors. We plan to extend and refine these guidelines in future releases of the treebank.

6 Editing Agreement

We utilize our two step review process to estimate agreement rates between annotators[7]. We measure agreement as the fraction of annotation tokens approved by the editor. Table 2 presents the agreement between annotators and reviewers, as well as the agreement between reviewers and the judges. Agreement measurements are provided for both the original the corrected versions of the dataset.

Overall, the results indicate a high agreement rate in the two editing tasks. Importantly, the gap between the agreement on the original and corrected sentences is small. Note that this result is obtained despite the introduction of several ESL annotation guidelines in the course of the annotation process, which inevitably increased the number of edits related to grammatical errors. We interpret this outcome as evidence for the effectiveness of the ESL annotation scheme in supporting consistent annotations of learner language.

[7]All experimental results on agreement and parsing exclude punctuation tokens.

Annotator-Reviewer	UPOS	POS	HIND	REL
original	98.83	98.35	97.74	96.98
corrected	99.02	98.61	97.97	97.20
Reviewer-Judge				
original	99.72	99.68	99.37	99.15
corrected	99.80	99.77	99.45	99.28

Table 2: Inter-annotator agreement on the entire TLE corpus. Agreement is measured as the fraction of tokens that remain unchanged after an editing round. The four evaluation columns correspond to universal POS tags, PTB POS tags, unlabeled attachment, and dependency labels. Cohen's Kappa scores (Cohen, 1960) for POS tags and dependency labels in all evaluation conditions are above 0.96.

7 Parsing Experiments

The TLE enables studying parsing for learner language and exploring relationships between grammatical errors and parsing performance. Here, we present parsing benchmarks on our dataset, and provide several estimates for the extent to which grammatical errors degrade the quality of automatic POS tagging and dependency parsing.

Our first experiment measures tagging and parsing accuracy on the TLE and approximates the global impact of grammatical errors on automatic annotation via performance comparison between the original and error corrected sentence versions. In this, and subsequent experiments, we utilize version 2.2 of the Turbo tagger and Turbo parser (Martins et al., 2013), state of the art tools for statistical POS tagging and dependency parsing.

Table 3 presents tagging and parsing results on a test set of 500 TLE sentences (9,591 original tokens, 9,700 corrected tokens). Results are provided for three different training regimes. The first regime uses the training portion of version 1.3 of the EWT, the UD English treebank, containing 12,543 sentences (204,586 tokens). The second training mode uses 4,124 training sentences (78,541 original tokens, 79,581 corrected tokens) from the TLE corpus. In the third setup we combine these two training corpora. The remaining 500 TLE sentences (9,549 original tokens, 9,695 corrected tokens) are allocated to a development set, not used in this experiment. Parsing of the test sentences was performed on predicted POS tags.

The EWT training regime, which uses out of domain texts written in standard English, provides the lowest performance on all the evaluation met-

Test set	Train Set	UPOS	POS	UAS	LA	LAS
TLE_{orig}	EWT	91.87	94.28	86.51	88.07	81.44
TLE_{corr}	EWT	92.9	95.17	88.37	89.74	83.8
TLE_{orig}	TLE_{orig}	95.88	94.94	87.71	89.26	83.4
TLE_{corr}	TLE_{corr}	96.92	95.17	89.69	90.92	85.64
TLE_{orig}	EWT+TLE_{orig}	93.33	95.77	90.3	91.09	86.27
TLE_{corr}	EWT+TLE_{corr}	94.27	96.48	92.15	92.54	88.3

Table 3: Tagging and parsing results on a test set of 500 sentences from the TLE corpus. EWT is the English UD treebank. TLE_{orig} are original sentences from the TLE. TLE_{corr} are the corresponding error corrected sentences.

rics. An additional factor which negatively affects performance in this regime are systematic differences in the EWT annotation of possessive pronouns, expletives and names compared to the UD guidelines, which are utilized in the TLE. In particular, the EWT annotates possessive pronoun UPOS as PRON rather than DET, which leads the UPOS results in this setup to be lower than the PTB POS results. Improved results are obtained using the TLE training data, which, despite its smaller size, is closer in genre and syntactic characteristics to the TLE test set. The strongest PTB POS tagging and parsing results are obtained by combining the EWT with the TLE training data, yielding 95.77 POS accuracy and a UAS of 90.3 on the original version of the TLE test set.

The dual annotation of sentences in their original and error corrected forms enables estimating the impact of grammatical errors on tagging and parsing by examining the performance gaps between the two sentence versions. Averaged across the three training conditions, the POS tagging accuracy on the original sentences is lower than the accuracy on the sentence corrections by 1.0 UPOS and 0.61 POS. Parsing performance degrades by 1.9 UAS, 1.59 LA and 2.21 LAS.

To further elucidate the influence of grammatical errors on parsing quality, table 4 compares performance on tokens in the original sentences appearing inside grammatical error tags to those appearing outside such tags. Although grammatical errors may lead to tagging and parsing errors with respect to any element in the sentence, we expect erroneous tokens to be more challenging to analyze compared to grammatical tokens.

This comparison indeed reveals a substantial difference between the two types of tokens, with an average gap of 5.0 UPOS, 6.65 POS, 4.67 UAS, 6.56 LA and 7.39 LAS. Note that differently from

Tokens	Train Set	UPOS	POS	UAS	LA	LAS
Ungrammatical	EWT	87.97	88.61	82.66	82.66	74.93
Grammatical	EWT	92.62	95.37	87.26	89.11	82.7
Ungrammatical	TLE_{orig}	90.76	88.68	83.81	83.31	77.22
Grammatical	TLE_{orig}	96.86	96.14	88.46	90.41	84.59
Ungrammatical	EWT+TLE_{orig}	89.76	90.97	86.32	85.96	80.37
Grammatical	EWT+TLE_{orig}	94.02	96.7	91.07	92.08	87.41

Table 4: Tagging and parsing results on the original version of the TLE test set for tokens marked with grammatical errors (Ungrammatical) and tokens not marked for errors (Grammatical).

the global measurements in the first experiment, this analysis, which focuses on the local impact of remove/replace errors, suggests a stronger effect of grammatical errors on the dependency labels than on the dependency structure.

Finally, we measure tagging and parsing performance relative to the fraction of sentence tokens marked with grammatical errors. Similarly to the previous experiment, this analysis focuses on remove/replace rather than insert errors.

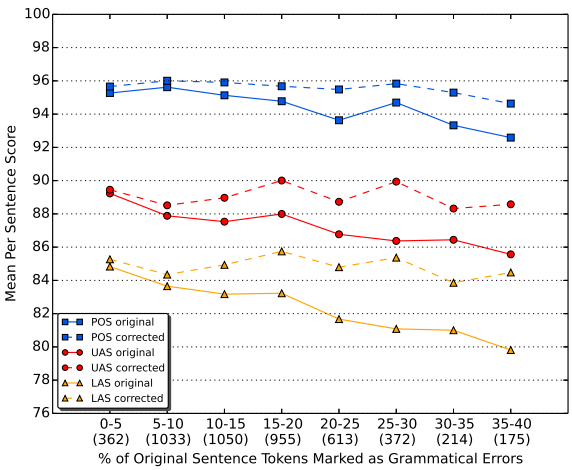


Figure 1: Mean per sentence POS accuracy, UAS and LAS of the Turbo tagger and Turbo parser, as a function of the percentage of original sentence tokens marked with grammatical errors. The tagger and the parser are trained on the EWT corpus, and tested on all 5,124 sentences of the TLE. Points connected by continuous lines denote performance on the original TLE sentences. Points connected by dashed lines denote performance on the corresponding error corrected sentences. The number of sentences whose errors fall within each percentage range appears in parenthesis.

Figure 1 presents the average sentential performance as a function of the percentage of tokens in the original sentence marked with grammati-

cal errors. In this experiment, we train the parser on the EWT training set and test on the entire TLE corpus. Performance curves are presented for POS, UAS and LAS on the original and error corrected versions of the annotations. We observe that while the performance on the corrected sentences is close to constant, original sentence performance is decreasing as the percentage of the erroneous tokens in the sentence grows.

Overall, our results suggest a negative, albeit limited effect of grammatical errors on parsing. This outcome contrasts a study by Geertzen et al. (2013) which reported a larger performance gap of 7.6 UAS and 8.8 LAS between sentences with and without grammatical errors. We believe that our analysis provides a more accurate estimate of this impact, as it controls for both sentence content and sentence length. The latter factor is crucial, since it correlates positively with the number of grammatical errors in the sentence, and negatively with parsing accuracy.

8 Related Work

Previous studies on learner language proposed several annotation schemes for both POS tags and syntax (Hirschmann et al., 2007; Dıaz-Negrillo et al., 2010; Dickinson and Ragheb, 2013; Rosen et al., 2014). The unifying theme in these proposals is a multi-layered analysis aiming to decouple the observed language usage from conventional structures in the foreign language.

In the context of ESL, Dıaz et al. (2010) propose three parallel POS tag annotations for the *lexical*, *morphological* and *distributional* forms of each word. In our work, we adopt the distinction between morphological word forms, which roughly correspond to our literal word readings, and distributional forms as the error corrected words. However, we account for morphological forms only when these constitute valid existing PTB POS tags and are contextually plausible. Furthermore, while the internal structure of invalid word forms is an interesting object of investigation, we believe that it is more suitable for annotation as word features rather than POS tags. Our treebank supports the addition of such features to the existing annotations.

The work of Ragheb and Dickinson (2009; 2012; 2013) proposes ESL annotation guidelines for POS tags and syntactic dependencies based on the CHILDES annotation framework. This approach, called "morphosyntactic dependencies" is related to our annotation scheme in its focus on surface structures. Differently from this proposal, our annotations are grounded in a parallel annotation of grammatical errors and include an additional layer of analysis for the corrected forms. Moreover, we refrain from introducing new syntactic categories and dependency relations specific to ESL, thereby supporting computational treatment of ESL using existing resources for standard English. At the same time, we utilize a multilingual formalism which, in conjunction with our literal annotation strategy, facilitates linking the annotations to native language syntax.

While the above mentioned studies focus on annotation guidelines, attention has also been drawn to the topic of parsing in the learner language domain. However, due to the shortage of syntactic resources for ESL, much of the work in this area resorted to using surrogates for learner data. For example, in Foster (2007) and Foster et al. (2008) parsing experiments are carried out on synthetic learner-like data, that was created by automatic insertion of grammatical errors to well formed English text. In Cahill et al. (2014) a treebank of secondary level native students texts was used to approximate learner text in order to evaluate a parser that utilizes unlabeled learner data.

Syntactic annotations for ESL were previously developed by Nagata et al. (2011), who annotate an English learner corpus with POS tags and shallow syntactic parses. Our work departs from shallow syntax to full syntactic analysis, and provides annotations on a significantly larger scale. Furthermore, differently from this annotation effort, our treebank covers a wide range of learner native languages. An additional syntactic dataset for ESL, currently not available publicly, are 1,000 sentences from the EFCamDat dataset (Geertzen et al., 2013), annotated with Stanford dependencies (De Marneffe and Manning, 2008). This dataset was used to measure the impact of grammatical errors on parsing by comparing performance on sentences with grammatical errors to error free sentences. The TLE enables a more direct way of estimating the magnitude of this performance gap by comparing performance on the same sentences in their original and error corrected versions. Our comparison suggests that the effect of grammatical errors on parsing is smaller that the one reported in this study.

9 Conclusion

We present the first large scale treebank of learner language, manually annotated and double-reviewed for POS tags and universal dependencies. The annotation is accompanied by a linguistically motivated framework for handling syntactic structures associated with grammatical errors. Finally, we benchmark automatic tagging and parsing on our corpus, and measure the effect of grammatical errors on tagging and parsing quality. The treebank will support empirical study of learner syntax in NLP, corpus linguistics and second language acquisition.

10 Acknowledgements

We thank Anna Korhonen for helpful discussions and insightful comments on this paper. We also thank Dora Alexopoulou, Andrei Barbu, Markus Dickinson, Sue Felshin, Jeroen Geertzen, Yan Huang, Detmar Meurers, Sampo Pyysalo, Roi Reichart and the anonymous reviewers for valuable feedback on this work. This material is based upon work supported by the Center for Brains, Minds, and Machines (CBMM), funded by NSF STC award CCF-1231216.

References

Aoife Cahill, Binod Gyawali, and James V Bruno. 2014. Self-training for parsing learner text. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 66–73.

Jacob Cohen. 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37.

David Crystal. 2003. *English as a global language*. Ernst Klett Sprachen.

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

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of LREC*, pages 4585–4592.

Ana Dıaz-Negrillo, Detmar Meurers, Salvador Valera, and Holger Wunsch. 2010. Towards interlanguage pos annotation for effective learner corpora in sla and flt. *Language Forum*, 36(1–2):139–154.

Markus Dickinson and Marwa Ragheb. 2009. Dependency annotation for learner corpora. In *Proceedings of the Eighth Workshop on Treebanks and Linguistic Theories (TLT-8)*, pages 59–70.

Markus Dickinson and Marwa Ragheb. 2013. Annotation for learner English guidelines, v. 0.1. Technical report, Indiana University, Bloomington, IN, June. June 9, 2013.

Jennifer Foster, Joachim Wagner, and Josef Van Genabith. 2008. Adapting a wsj-trained parser to grammatically noisy text. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 221–224. Association for Computational Linguistics.

Jennifer Foster. 2007. Treebanks gone bad. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):129–145.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale l2 databases: The ef-cambridge open language database (efcamdat). In *Proceedings of the 31st Second Language Research Forum. Somerville, MA: Cascadilla Proceedings Project*.

Hagen Hirschmann, Seanna Doolittle, and Anke Lüdeling. 2007. Syntactic annotation of non-canonical linguistic structures.

André FT Martins, Miguel Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *ACL (2)*, pages 617–622. Citeseer.

Ryan T McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith B Hall, Slav Petrov, Hao Zhang, Oscar Täckström, et al. 2013. Universal dependency annotation for multilingual parsing. In *ACL (2)*, pages 92–97. Citeseer.

Ryo Nagata, Edward Whittaker, and Vera Sheinman. 2011. Creating a manually error-tagged and shallow-parsed learner corpus. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1210–1219. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *CoNLL Shared Task*, pages 1–14.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, pages 572–581.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*.

Marwa Ragheb and Markus Dickinson. 2012. Defining syntax for learner language annotation. In *COLING (Posters)*, pages 965–974.

Victoria Rosén and Koenraad De Smedt. 2010. Syntactic annotation of learner corpora. *Systematisk, variert, men ikke tilfeldig*, pages 120–132.

Alexandr Rosen, Jirka Hana, Barbora Štindlová, and Anna Feldman. 2014. Evaluating and automating the annotation of a learner corpus. *Language Resources and Evaluation*, 48(1):65–92.

Beatrice Santorini. 1990. Part-of-speech tagging guidelines for the penn treebank project (3rd revision). *Technical Reports (CIS)*.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel R Bowman, Miriam Connor, John Bauer, and Christopher D Manning. 2014. A gold standard dependency corpus for english. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of the acl 2010 conference short papers*, pages 353–358. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *ACL*, pages 180–189.

Extracting token-level signals of syntactic processing from fMRI - with an application to PoS induction

Joachim Bingel Maria Barrett Anders Søgaard
Centre for Language Technology, University of Copenhagen
Njalsgade 140, 2300 Copenhagen S, Denmark
{bingel, barrett, soegaard}@hum.ku.dk

Abstract

Neuro-imaging studies on reading different parts of speech (PoS) report somewhat mixed results, yet some of them indicate different activations with different PoS. This paper addresses the difficulty of using fMRI to discriminate between linguistic tokens in reading of running text because of low temporal resolution. We show that once we solve this problem, fMRI data contains a signal of PoS distinctions to the extent that it improves PoS induction with error reductions of more than 4%.

1 Introduction

A few recent studies have tried to extract morphosyntactic signals from measurements of human sentence processing and used this information to improve NLP models. Klerke et al. (2016), for example, used eye-tracking recordings to regularize a sentence compression model. More related to this work, Barrett et al. (2016) recently used eye-tracking recordings to induce PoS models. However, a weakness of eye-tracking data is that while eye movement surely does reflect the temporal aspect of cognitive processing, it is only a proxy of the latter and does not directly represent which processes take place in the brain.

A recent neuro-imaging study suggests that concrete nouns and verbs elicit different brain signatures in the frontocentral cortex, and that concrete and abstract nouns elicit different brain activation patterns (Moseley and Pulvermüller, 2014). Also, for example, concrete verbs activate motor and premotor cortex more strongly than concrete nouns, and concrete nouns activate inferior frontal areas more strongly than concrete verbs. A decade earlier, Tyler et al. (2004) showed that the left inferior frontal gyrus was more strongly activated in processing regularly inflected verbs compared to regularly inflected nouns.

Such studies suggest that different parts of our brains are activated when reading different parts of speech (PoS). This would in turn mean that neuro-images of readers carry information about the grammatical structure of what they read. In other words, neuro-imaging provides a partial, noisy annotation of the data with respect to morphosyntactic category.

Say neuro-imaging data of readers was readily available. Would it be of any use to, for example, engineers interested in PoS taggers for low-resource languages? This is far from obvious. In fact, it is well-known that neuro-imaging data from reading is noisy, in part because the reading signal is not always very distinguishable (Tagamets et al., 2000), and also because the content of what we read may elicit certain activation in brain regions e.g. related to sensory processing (Boulenger et al., 2006; González et al., 2006).

Other researchers such as Borowsky et al. (2013) have also questioned that there are differences, claiming to show that the majority of activation is shared between nouns and verbs – including in regions suggested by previous researchers as unique to either nouns or verbs. Berlingeri et al. (2008) argue that only verbs could be associated with unique regions, not nouns.

In this paper we nevertheless explore this question. The paper should be seen as a proof of concept that interesting linguistic signals can be extracted from brain imaging data, and an attempt to show that learning NLP models from such data could be a way of pushing the boundaries of both fields.

Contributions (a) We present a novel technique for extracting syntactic processing signal at the token level from neuro-imaging data that is charac-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 747–755,
Berlin, Germany, August 7-12, 2016.

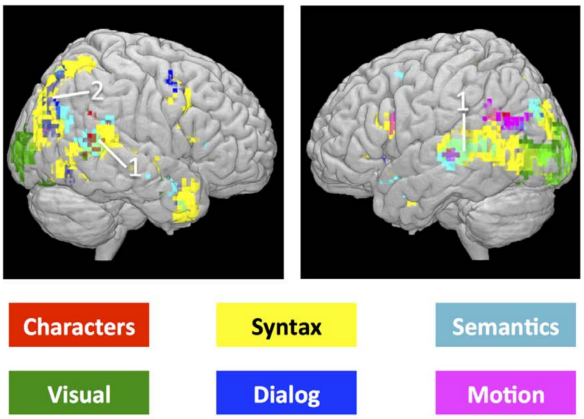


Figure 1: Neural activity by brain region and type of information processed, as measured and rendered by Wehbe et al. (2014).

terized by low temporal resolution. (b) We demonstrate that the fMRI data improves performance of a type-constrained, second order hidden Markov model for PoS induction. Our model leads to an error reduction of more than 4% in tagging accuracy despite very little training data, which to the best of our knowledge is the first positive result on weakly supervised part-of-speech induction from fMRI data in the literature.

2 fMRI

Functional Magnetic Resonance Imaging (fMRI) is a technology for spatial visualization of brain activity. It measures the changes in oxygenation of the blood in the brain, often by use of the blood oxygenation level-dependent contrast (Ogawa et al., 1992), which correlates with neural activity. While the spatial resolution of fMRI is very high, its temporal resolution is low compared to other brain imaging technologies like EEG, which usually returns millisecond records of brain activity, but on the contrary have low spatial resolution. The temporal resolution of fMRI is usually between 0.5Hz and 1Hz. fMRI data contains representations of neural activity of millimeter-sized cubes called *voxels*.

The high spatial resolution may enable us to detect fine differences in brain activation patterns, such as between processing nouns and verbs, but the low temporal resolution is a real challenge when the different tokens are processed serially and quickly after each other, as is the case in reading.

Another inherent challenge when working with fMRI data is the lag between the the reaction to a stimulus and the point when it becomes visible through fMRI. This lag is called the *hemodynamic response latency*. While we know from brain imaging technologies with higher temporal resolution that the neural response to a stimuli happens within milliseconds, it only shows in fMRI data after a certain period of time, which further blurs the low temporal dimension of serial fMRI recordings. This latency has been studied as long as fMRI technology itself. It depends on the blood vessels and varies between e.g. voxels, brain regions, subjects, and tasks. A meta study of the hemodynamic response report latencies between 4 and 14 seconds in healthy adults, though latencies above 11 seconds are less typically reported (Handwerker et al., 2012). According to Handwerker et al. (2012), the precise response shape for a given stimulus and voxel region is hard to predict and remains a challenge when modeling temporal aspects of fMRI data.

Figure 1 visualizes the neural activations in different brain regions as a reaction to the type of information that is processed during reading. See Price (2012) for a thorough review of fMRI language studies.

Wehbe et al. (2014) presented a novel approach to fMRI studies of linguistic processing by studying a more naturalistic reading scenario, and modeling the entire process of reading and story understanding. They used data from 8 subjects reading contextualized, running text: a chapter from a *Harry Potter* book. The central benefit of this approach is that it allows studies of complex text processing closer to a real-life reading experience. Wehbe et al. (2014) used this data to train a comprehensive, generative model that—given a text passage—could predict the fMRI-recorded activity during the reading of this passage. Using the same data, our goal is to model a specific aspect of the story understanding process, i.e. the grammatical processing of words.

3 Data

3.1 Textual data

We use the available fMRI recordings from Wehbe et al. (2014), where 8 adult, native English speakers read chapter 9 from *Harry Potter and the Sorcerer's Stone* in English. The textual data as provided in the data set does not explicitly mark sentence boundaries, neither is punctuation sep-

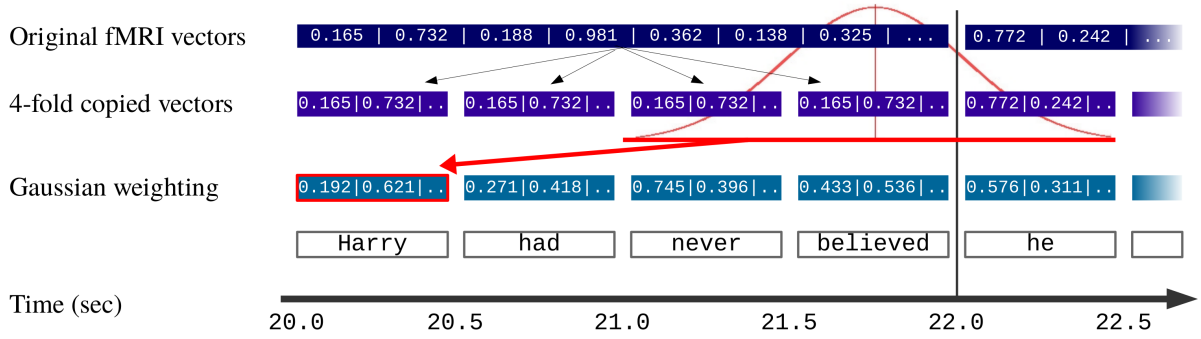


Figure 2: Computation of token-level fMRI vectors from the original fMRI data for the first token "Harry" while accounting for hemodynamic response latency using a Gaussian sliding window over a certain time window (indicated by red horizontal line). The final fMRI vector for "Harry" (red box) is computed as specified in Equation 1. In this example, the time stamp t for the token is $20s$ and the time window stretches from $t + 1s$ to $t + 2.5s$.

arated from the tokens at the end of clauses and sentences. As the temporal alignment between tokens and fMRI recordings (see below) forbids us to detach punctuation marks from their preceding tokens and introduce them as new tokens, we opt to remove all punctuation from the data. In the same process, we use simple heuristics to detect sentence boundaries. Finally, we correct errors in sentence splitting manually.

The chapter counts 4,898 tokens (excluding punctuation) and 1,411 types in 408 sentences.

3.2 fMRI data

The fMRI data from the same data set is available as high-dimensional vectors of flattened third-order tensors, in which each component represents the blood-oxygen-level dependent contrast for a certain voxel in the three-dimensional fMRI image. The resolution of the image is at $3 \times 3 \times 3$ mm, such that the brain activity for the eight subjects is represented by approximately 31,400 voxels on average (standard deviation is 3,607) depending on the size of their brain.

This data is recorded every two seconds during the reading process, in which each token is consecutively displayed for 0.5 seconds on a screen inside the fMRI scanner. Prior to reading, the subjects are asked to focus on a cross displayed at the center of the screen in a warm-up phase of 20 seconds. The chapter is divided into four blocks, separated by additional concentration phases of 20 seconds. Furthermore, paragraphs are separated by a 0.5-seconds display of a cross at the center of the screen.

As mentioned in the preceding section, punctuation marks were not displayed separately, but instead attached to the preceding token. This is arguably motivated through the attempt to create a reading scenario that is as natural as possible within the limitations of an fMRI recording. In similar fashion, contractions such as *don't* or *he's* were represented as one token, just as they appear in the original text.

In order to make the data feasible for our HMM approach (see Section 4), we apply Principal Component Analysis (PCA) to the high-dimensional fMRI vectors. We initially tune the number of principal components, which we describe in Section 5.

3.2.1 Computing token-level fMRI vectors

As outlined above, the time resolution of the fMRI recordings means that every block of four consecutive tokens is time-aligned with a single fMRI image. Naturally, this shared representation of consecutive tokens complicates any language learning at the token level. Furthermore, the hemodynamic response latency inherent to fMRI recordings entails that the image recorded while reading a certain token most probably does not give any clues about the mental state elicited by this stimulus.

We therefore face the dual challenge of

1. inferring token-level information from supra-token recordings, and

2. identifying the lag after which the perceptual effects of reading a given token are visible.

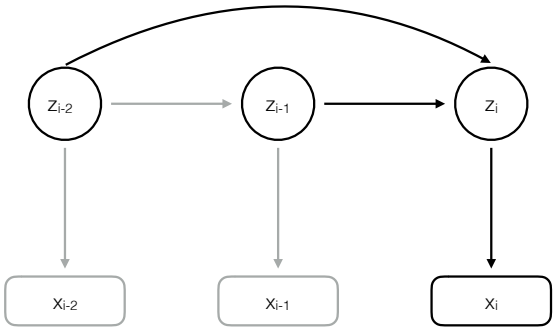


Figure 3: Second-order HMM incorporating transitional probabilities from first and second-degree preceding states.

We address this problem through the following procedure that we illustrate in Figure 2. First, we copy the number of fMRI recordings fourfold, such that every fMRI vector is aligned to exactly one token (excluding the vectors that are recorded while no token was displayed). The representation for a given token is then computed as a weighted average over all fMRI vectors that lie within a certain time window in relation to the token in question. Two consecutive tokens that originally lie within the same block of four thus receive different representations, provided that the window is large enough to transcend the border between two blocks.

The fMRI representation for the token at time stamp t is given by

$$v_t = \frac{1}{|V|} \sum_{k=1}^{|V|} V_k \cdot w_k \tag{1}$$

where V is the series of fMRI vectors within the time window $[t+s, t+e]$, and w is a Gaussian window of $|V|$ points, with a standard deviation of 1. In factoring the Gaussian weight vector into the equation, we lend less weight to the fMRI recordings at the outset and at the end of the time window specified through s (start) and e (end).

4 Model

We use a second-order hidden Markov model (HMM) with Wiktionary-derived type constraints (Li et al., 2012) as our baseline for weakly supervised PoS induction. We use the original implementation by Li et al. (2012). The model is a type-constrained, second order version of the first-order featurized HMM previously introduced by Berg-Kirkpatrick et al. (2010).

In each state z_i, a PoS HMM generates a sequence of words by consecutively generating word emissions x_i and successor states z_{i+1}. The emission probabilities and state transition probabilities are multinomial distributions over words and PoS. The joint probability of a word sequence and a tag sequence is

$$P_\theta(x, z) = P_\theta(z_1) \prod_{i=1} P_\theta(x_i|z_i) \prod_{i=2} P_\theta(z_i|z_{i-1}) \tag{2}$$

Following Berg-Kirkpatrick et al. (2010), the model calculates the probability distribution θ that parameterizes the emission probabilities as the output of a maximum entropy model, which enables unsupervised learning with a rich set of features. We thus let

$$\theta_{x_i,z_i} = \frac{\exp(\mathbf{w}^\intercal \mathbf{f}(x_i, z_i))}{\sum_{x'} \exp(\mathbf{w}^\intercal \mathbf{f}(x', z_i))} \tag{3}$$

where $\mathbf{w}$ is a weight vector and $\mathbf{f}(x_i, z_i)$ is a feature function that will, in our case, consider the fMRI vectors v_t that we computed in section 3.2.1 and a number of basic features that we adopt from the original model (Li et al., 2012). See Section 5 for details.

In addition, we use a second-order HMM, first introduced for PoS tagging in Thede and Harper (1999), in which transitional probabilities are also considered for second-degree subsequent states (cf. figure 3). Here, the joint probability becomes

$$\begin{aligned} P_\theta(x, z) = {} & P_\theta(z_1) P_\theta(x_1|z_1) P_\theta(z_2|z_1) \\ & \prod_{i=2} P_\theta(x_i|z_i) \prod_{i=3} P_\theta(z_i|z_{i-2}, z_{i-1}) \end{aligned} \tag{4}$$

In order to optimize the HMM (including the weight vector $\mathbf{w}$), the model uses the EM algorithm as applied for feature-rich, locally normalized models introduced in Berg-Kirkpatrick et al. (2010), with the important modification that we use type constraints in the E-step, following Li et al. (2012). Specifically, for each state z_i, the emission probability $P(x_i|z_i)$ is initialized randomly for every word type associated with z_i in our tag dictionary (the type constraints). This weakly supervised setup allows us to predict the actual PoS tags instead of abstract states. The M-step is solved using L-BFGS (Liu and Nocedal, 1989)

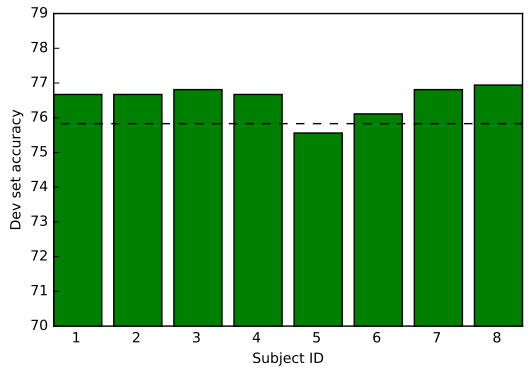


Figure 4: Accuracy on the development set for the different subjects when trained and tested on fMRI data from only this one subject. Dashed line is the development set baseline. Only in one out of eight cases does adding fMRI features lead to worse performance.

EM-HMM Parameters We use the same setting as Li et al. (2012) for the number of EM iterations, fixing this parameter to 30 for all experiments.

5 Experiments

Experimental setup From the neuro-imaging dataset described above, we use 41 sentences (720 tokens) as a development set and 41 sentences (529 tokens) as a test set, and the remaining 326 sentences (corresponding to 80%) for training our model.

Basic features The basic features of all the models (except when explicitly stated otherwise) are based on seven features that we adopt from Li et al. (2012), capturing word form, hyphenation, suffix patterns, capitalization and digits in the token.

Wiktionary Of the 1,411 word types in the corpus, we find that 1,381 (97.84%) are covered by the Wiktionary dump made available by Li et al. (2012),[1] which we use as our type constraints when inducing our models.

5.1 Part-of-speech annotation

Though Wehbe et al. (2014) also provide syntactic information, these are automatic parses that are not suitable for the evaluation of our model. The development and test data are therefore manually annotated for universal part-of-speech tags (Petrov et al., 2011) by two linguistically trained annotators. The development set was annotated by both annotators, who reached an inter-annotator agreement of 0.926 in accuracy and 0.928 in weighted F_1. For the final development and test data, disagreements were resolved by the annotators.

5.2 Non-fMRI baselines

Our first baseline is a second-order HMM with type constraints from Wiktionary; this in all respects the model proposed by Liu et al. (2012), except trained on our small Harry Potter corpus. In a second baseline model, we also incorporate 300-dimensional GloVe word embeddings trained on Wikipedia and the Gigaword corpus (Pennington et al., 2014). We also test a version of the baseline without the basic features to get an estimate of the contribution of this aspect of the setup.

5.3 Token-level fMRI

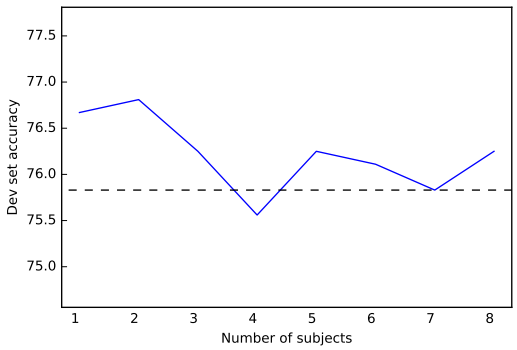
We run a series of experiments with token-level fMRI vectors that we obtain as described in Section 3.2.1. Initially, we train separate models for each of the eight individual subjects, whose performance on the development data are illustrated in Figure 4.

5.3.1 Tuning hyperparameters

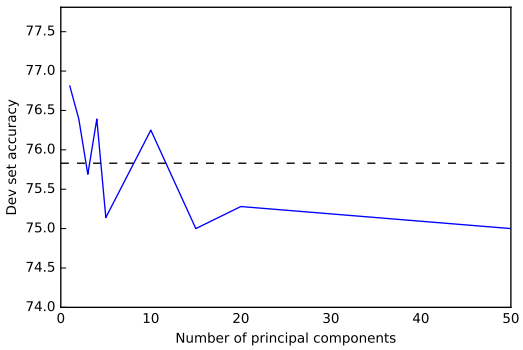
We tune the following hyperparameters on the token-level development set in the following order: the number of subjects to use, the number of principal components per subject, and the time window. For the earlier tuning processes we fix the later hyperparameters to values we consider reasonable, but once we have tuned a hyperparameter, we use the best value from this tuning process for later tuning steps. The initial values are: 10 principal components and a time window of $[t + 0s, t + 6s]$.

Number of subjects To reduce the chance of overfitting, we use fMRI data from several readers in our model. The data from Wehbe et al. (2014) would in theory allow us to average the three-dimensional image space for any number of readers, but this is not feasible if only for the difference in brain sizes between the subjects. It is not feasible, either, to average over the eigenvectors that we obtain from PCA, as the eigenvectors between subjects do not share the same (or any concrete) feature space. We therefore concatenate

[1] https://code.google.com/archive/p/wikily-supervised-pos-tagger/



(a) Learning curve for increasing number of subjects in the model. Fixed hyper-parameters: 10 principal components and a time window of $[t + 0s, t + 6s]$.



(b) Learning curve for increasing number of principal components per subject in the model. Number of principal components $\in \{1, 2, 3, 4, 5, 10, 15, 20, 50\}$ Fixed hyper-parameters: 8 subjects, a time window of $[t + 0s, t + 6s]$.

Figure 5: Exploring two individual hyper-parameters of the model on development set. Dashed lines indicate the development set baseline.

the eigenvectors that we obtain for different subjects, such that the feature vectors grow in length as the number of included subjects increases.

As Figure 5a shows, exploring an increasing number of subjects in the model does not seem to a have consistent effect on development set accuracy. However, we expect an increased robustness from a model that incorporates a greater number of subjects. In all following experiments we therefore use data from all eight readers, but we would also expect a model with fewer subjects to perform reasonably.

Principal components Fixing the number of subjects to eight, we then perform experiments to determine the number of principal components per subject to consider in our model, whose results are visualized in Figure 5b. We observe the first eigenvectors carry a strong signal, while a great number of principal components tends to water down the signal and lead to worse performance. We choose to continue using 10 dimensions in all further experiments.

Time window for token vectors We next run experiments to determine the optimal time window for the computation of the token vectors, using different combinations of start and end times in relation to the token time stamps, but keeping the number of subjects and principal components constant at eight and ten, respectively. These experiments yield three different time windows with an equally good performance on the development set: $[t - 4s, t + 10s]$, $[t + 2s, t + 8s]$ and $[t + 0s, t + 6s]$. Note that due to the Gaussian weighting the centre of the interval gets more weight than the edges and that $[t-4s, t+10s]$ and $[t+0s, t+6s]$ have the same centre, $t+3$. While $[t+2s, t+8s]$ and $[t+0, t+6]$ align better with psycholinguistic expectations, $[t - 4s, t + 10s]$ makes our model less prone to overfitting. We therefore select the model averaging over the largest time window.

5.4 Type-level fMRI aggregates

Next, we aggregate token vectors to compute their type-level averages, in an effort to explore to which degree neural activity is dependent on the read word type rather than the concrete grammatical environment, and whether this can allow our model to draw conclusions about the grammatical class of a token. We compute the type-level aggregates as the component-wise arithmetic mean of the token vectors that we extract using the parameter settings optimized above. Note, however, that out of the 4,898 tokens in the text, 823 (16.9%) occur only once.

6 Results

Table 1 reports the results that we obtain with our final hyper-parameter settings, which are as follows:

Number of subjects	8
Principal components	10
Start of time window	$t - 4s$
End of time window	$t + 10s$

The results show that our model leads to a consid-

	Accuracy
Baseline (Li et al., 2012)	69.57
Baseline+GloVe	69.38
Baseline w/o basic feats	55.53
fMRI (token-level) w/o basic feats	56.99
fMRI (type-level)	70.32
fMRI (token-level)	**70.89**
Error reduction over baseline	04.34

Table 1: Tagging accuracy on test data for the different models. The fMRI model is significantly better than the baseline ($p = 0.014$, Bootstrap).

Class	Prec.	Rec.	F_1	± BL
ADJ	37.50	42.86	40.00	+2.71
ADP	83.67	77.36	80.39	+1.54
ADV	66.00	58.93	62.26	+5.69
CONJ	70.97	70.97	70.97	±0.00
DET	80.49	80.49	80.49	+3.38
NOUN	70.37	76.00	73.08	+0.28
NUM	00.00	00.00	00.00	-20.00
PRON	88.68	74.60	81.03	+4.76
PRT	41.67	41.67	41.67	+11.67
VERB	74.36	76.32	75.32	-0.95

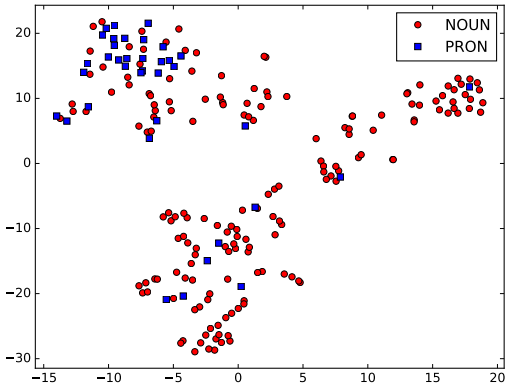
Table 2: Test data tagging performance by part-of-speech class for the best fMRI model. The rightmost column displays the difference in F_1 compared to the baseline model.

erable error reduction over the baseline model as well as the embeddings-enriched baseline model. It also outperforms the model which uses type-level averages over the fMRI recordings. Leaving out the basic features hurts performance, but even without the basic features the fMRI data can reduce error with 3.28% on the test set. In Table 2 we present the performance on the individual PoS classes under our best model.

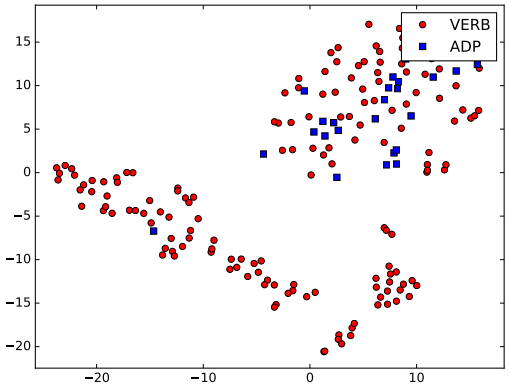
7 Analysis and Discussion

7.1 What's in the fMRI vectors?

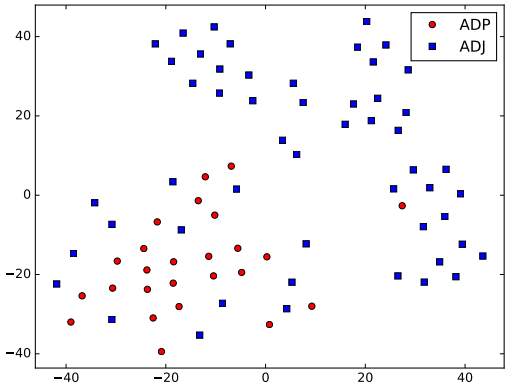
t-SNE (Van der Maaten and Hinton, 2008) is a powerful supervised dimensionality reduction tool for visualizing high-dimensional data in two-dimensional space using Stochastic Neighbor Embedding. In Figure 6, we visualize pairs of PoS classes of the test data in a two-dimensional re-



(a) NOUN and PRON



(b) VERB and ADP



(c) ADP and ADJ

Figure 6: Selected t-SNE visualizations of fMRI vectors for all tokens of a class of the test set. The visualizations show that datapoints of a PoS class tend to cluster in the fMRI vector space.

duction of the embedding space obtained when using the settings of the best fMRI model. The fact that we can discriminate reasonably well between, e.g., nouns and pronouns, verbs and adpositions, as well as adpositions and adjectives on the basis of fMRI data is to the best of our knowledge a new finding.

7.2 Discussion of the results

We showed that by careful model tuning and design it is possible to extract a signal of grammatical processing in the brain from fMRI. The figures that we present in Table 1 reflect, to our knowledge, the first successful results in inferring grammatical function at the token level from fMRI data. Our best model, which we train on the ten principal components from the fMRI recordings of eight readers, achieves an error reduction of over 4% despite a very small amount of training data. We find that our best model uses a very wide window of fMRI recordings to compute the representations for individual tokens, considering all recordings from 4 seconds before the token is displayed until 10 seconds after the token is displayed. Our best explanation for why the incorporation of preceding fMRI measurements is beneficial to our model, is that the grammatical function of a token may be predictable from a reader's cognitive state while reading preceding tokens. However, note that the measurements at the far ends of the time window only factor into the token vector to a small degree as a consequence of the Gaussian weighting. Our experiments further suggest that using token-level information instead of type-level features, such as word embeddings or type averages of fMRI vectors, is helpful for PoS induction that already is type-constrained.

Recently, Huth et al. (2016) found that semantically related words are processed in the same area of the brain. Open questions for future work include whether there is a bigger potential for using fMRI data for semantic rather than syntactic NLP tasks, and whether the signal we find mainly stems from semantic processing differences.

8 Conclusion

This paper presents the first experiments inducing part of speech from fMRI reading data. Cognitive psychologists have debated whether grammatical differences lead to different brain activation patterns. Somewhat surprisingly, we find that the fMRI data contains a strong signal, enabling a 4% error reduction over a state-of-the-art unsupervised PoS tagger. While our approach may not be readily applicable for developing NLP models today, we believe that the presented results may inspire NLP researchers to consider learning models from combinations of linguistic resources and auxiliary, behavioral data that reflects human cognition.

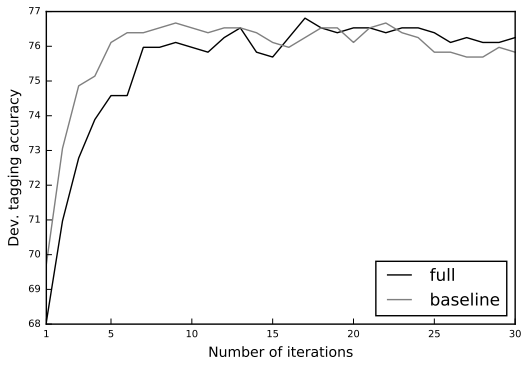


Figure 7: Learning curve of tagging accuracy on the development set as a function of different number of EM iterations for baseline model and the full model for iteration numbers $\in [1, 30]$. Fixed hyper-parameters: 8 subjects, 10 principal components, and a time window of $t - 4s$ to $t + 10s$

Acknowledgements

This research was partially funded by the ERC Starting Grant LOWLANDS No. 313695, as well as by Trygfonden.

Supplementary material

Number of EM iterations As supplementary material, we present the EM learning curve in Figure 7, which shows a steep learning curve at the beginning and relatively stable performance figures after 15 iterations for the full model and 10 iterations for the baseline model.

References

Maria Barrett, Joachim Bingel, Frank Keller, and Anders Søgaard. 2016. Weakly supervised part-of-speech induction using eye-tracking data. In *ACL*.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Cote, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL*, pages 582–590.

Manuela Berlingeri, Davide Crepaldi, Rossella Roberti, Giuseppe Scialfa, Claudio Luzzatti, and Eraldo Paulesu. 2008. Nouns and verbs in the brain: Grammatical class and task specific effects as revealed by fMRI. *Cognitive Neuropsychology*, 25(4):528–558.

Ron Borowsky, Carrie Esopenko, Layla Gould, Naila Kuhlmann, Gordon Sarty, and Jacqueline Cummine. 2013. Localisation of function for noun and verb reading: converging evidence for shared processing from fmri activation and reaction time. *Language and Cognitive Processes*, 28(6):789–809.

Véronique Boulenger, Alice C Roy, Yves Paulignan, Viviane Deprez, Marc Jeannerod, and Tatjana A Nazir. 2006. Cross-talk between language processes and overt motor behavior in the first 200 msec of processing. *Journal of cognitive neuroscience*, 18(10):1607–1615.

Julio González, Alfonso Barros-Loscertales, Friedemann Pulvermüller, Vanessa Meseguer, Ana Sanjuán, Vicente Belloch, and César Ávila. 2006. Reading cinnamon activates olfactory brain regions. *Neuroimage*, 32(2):906–912.

Daniel A Handwerker, Javier Gonzalez-Castillo, Mark D'Esposito, and Peter A Bandettini. 2012. The continuing challenge of understanding and modeling hemodynamic variation in fmri. *Neuroimage*, 62(2):1017–1023.

Alexander G Huth, Wendy A de Heer, Thomas L Griffiths, Frédéric E Theunissen, and Jack L Gallant. 2016. Natural speech reveals the semantic maps that tile human cerebral cortex. *Nature*, 532(7600):453–458.

Sigrid Klerke, Yoav Goldberg, and Anders Søgaard. 2016. Improving sentence compression by learning to predict gaze. In *NAACL*.

Shen Li, João Graça, and Ben Taskar. 2012. Wikily supervised part-of-speech tagging. In *EMNLP*, pages 1389–1398.

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

Xiaohua Liu, Ming Zhou, Furu Wei, Zhongyang Fu, and Xiangyang Zhou. 2012. Joint inference of named entity recognition and normalization. In *ACL*, pages 526–535.

Rachel L Moseley and Friedemann Pulvermüller. 2014. Nouns, verbs, objects, actions, and abstractions: local fmri activity indexes semantics, not lexical categories. *Brain and language*, 132:28–42.

Seiji Ogawa, David W Tank, Ravi Menon, Jutta M Ellermann, Seong G Kim, Helmut Merkle, and Kamil Ugurbil. 1992. Intrinsic signal changes accompanying sensory stimulation: functional brain mapping with magnetic resonance imaging. *Proceedings of the National Academy of Sciences*, 89(13):5951–5955.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. CoRR abs/1104.2086.

Cathy J Price. 2012. A review and synthesis of the first 20years of pet and fmri studies of heard speech, spoken language and reading. *Neuroimage*, 62(2):816–847.

M-A Tagamets, Jared M Novick, Maria L Chalmers, and Rhonda B Friedman. 2000. A parametric approach to orthographic processing in the brain: an fmri study. *Cognitive Neuroscience, Journal of*, 12(2):281–297.

Scott Thede and Mary Harper. 1999. A second-order hidden markov model for part-of-speech tagging. In *ACL*, pages 175–182.

Lorraine Tyler, Peter Bright, Paul Fletcher, and Emmanuel Stamatakis. 2004. Neural processing of nouns and verbs: The role of inflectional morphology. *Neuropsychologia*, 42(4):512–523.

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

Leila Wehbe, Brian Murphy, Partha Talukdar, Alona Fyshe, Aaditya Ramdas, and Tom Mitchell. 2014. Simultaneously uncovering the patterns of brain regions involved in different story reading subprocesses. *PloS one*, 9(11):e112575.

Bidirectional Recurrent Convolutional Neural Network for Relation Classification

Rui Cai, Xiaodong Zhang and **Houfeng Wang***
Key Laboratory of Computational Linguistics (Ministry of Education),
School of EECS, Peking University, Beijing, 100871, China
Collaborative Innovation Center for Lanuage Ability, Xuzhou, Jiangsu, 221009, China
{cairui, zxdcs, wanghf}@pku.edu.cn

Abstract

Relation classification is an important semantic processing task in the field of natural language processing (NLP). In this paper, we present a novel model BRCNN to classify the relation of two entities in a sentence. Some state-of-the-art systems concentrate on modeling the shortest dependency path (SDP) between two entities leveraging convolutional or recurrent neural networks. We further explore how to make full use of the dependency relations information in the SDP, by combining convolutional neural networks and two-channel recurrent neural networks with long short term memory (LSTM) units. We propose a bidirectional architecture to learn relation representations with directional information along the SDP forwards and backwards at the same time, which benefits classifying the direction of relations. Experimental results show that our method outperforms the state-of-the-art approaches on the SemEval-2010 Task 8 dataset.

1 Introduction

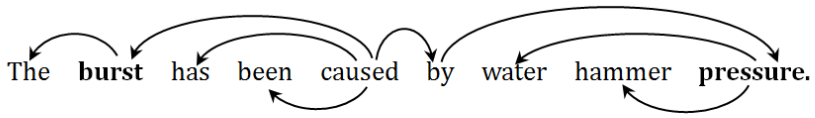
Relation classification aims to classify the semantic relations between two entities in a sentence. For instance, in the sentence "The $[\text{burst}]_{e_1}$ has been caused by water hammer $[\text{pressure}]_{e_2}$", entities *burst* and *pressure* are of relation Cause-Effect(e_2, e_1). Relation classification plays a key role in robust knowledge extraction, and has become a hot research topic in recent years.

Nowadays, deep learning techniques have made significant improvement in relation classification, compared with traditional relation classification approaches focusing on designing effective features (Rink and Harabagiu, 2010) or kernels (Zelenko et al., 2003; Bunescu and Mooney, 2005) Although traditional approaches are able to exploit the symbolic structures in sentences, they still suffer from the difficulty to generalize over the unseen words. Some recent works learn features automatically based on neural networks (NN), employing continuous representations of words (word embeddings). The NN research for relation classification has centered around two main network architectures: convolutional neural networks and recursive/recurrent neural networks. Convolutional neural network aims to generalize the local and consecutive context of the relation mentions, while recurrent neural networks adaptively accumulate the context information in the whole sentence via memory units, thereby encoding the global and possibly unconsecutive patterns for relation classification. Socher et al. (2012) learned compositional vector representations of sentences with a recursive neural network. Kazuma et al. (2013) proposed a simple customizaition of recursive neural networks. Zeng et al. (2014) proposed a convolutional neural network with position embeddings.

Recently, more attentions have been paid to modeling the shortest dependency path (SDP) of sentences. Liu et al. (2015) developed a dependency-based neural network, in which a convolutional neural network has been used to capture features on the shortest path and a recursive neural network is designed to model subtrees. Xu et al. (2015b) applied long short term memory (LSTM) based recurrent neural networks (RNNs) along the shortest dependency path. However, SDP is a special structure in which every two neighbor words are separated by a dependency relations. Previous works treated dependency relations in the same

*Corresponding author

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 756–765,
Berlin, Germany, August 7-12, 2016.



Realtion	Shortest Dependency Path
Cause-Effect(e_2,e_1)	burst ←nsubjpass— caused —prep→ by —pobj→ pressure
Cause-Effect(e_1,e_2)	pressure ←pobj— by ←prep— caused —nsubjpass→ burst

Figure 1: The shortest dependency path representation for an example sentence from SemEval-08.

way as words or some syntactic features like part-of-speech (POS) tags, because of the limitations of convolutional neural networks and recurrent neural networks. Our first contribution is that we propose a recurrent convolutional neural network (RCNN) to encode the global pattern in SDP utilizing a two-channel LSTM based recurrent neural network and capture local features of every two neighbor words linked by a dependency relation utilizing a convolution layer.

We further observe that the relationship between two entities are directed. For instance, Figure 1 shows that the shortest path of the sentence "The [burst]$_{e_1}$ has been caused by water hammer [pressure]$_{e_2}$." corresponds to relation Cause-Effect(e_2, e_1). The SDP of the sentence also corresponds to relation Cause-Effect(e_2, e_1), where e_1 refers to the entity at front end of SDP and e_2 refers to the entity at back end of SDP, and the inverse SDP corresponds to relation Cause-Effect(e_1, e_2). Previous work (Xu et al., 2015b) simply transforms a (K+1)-relation task into a $(2K + 1)$ classification task, where 1 is the *Other* relation and K is the number of directed relations. Besides, the recurrent neural network is a biased model, where later inputs are more dominant than earlier inputs. It could reduce the effectiveness when it is used to capture the semantics of a whole shortest dependency path, because key components could appear anywhere in a SDP rather than the end.

Our second contribution is that we propose a bidirectional recurrent convolutional neural networks (BRCNN) to learn representations with bidirectional information along the SDP forwards and backwards at the same time, which also strengthen the ability to classifying directions of relationships between entities. Experimental results show that the bidirectional mechanism significantly improves the performance.

We evaluate our method on the SemEval-2010 relation classification task, and achieve a state-of-the-art F_1-score of 86.3%.

2 The Proposed Method

In this section, we describe our method in detail. Subsection 2.1 provides an overall picture of our BCRNN model. Subsection 2.2 presents the rationale of using SDPs and some characteristics of SDP. Subsection 2.3 describes the two-channel recurrent neural network, and bidirectional recurrent convolutional neural network is introduced in Subsection 2.4. Finally, we present our training objective in Subsection 2.5.

2.1 Framework

Our BCRNN model is used to learn representations with bidirectional information along the SDP forwards and backwards at the same time. Figure 2 depicts the overall architecture of the BRCNN model.

Given a sentence and its dependency tree, we build our neural network on its SDP extracted from the tree. Along the SDP, two recurrent neural networks with long short term memory units are applied to learn hidden representations of words and dependency relations respectively. A convolution layer is applied to capture local features from hidden representations of every two neighbor words and the dependency relations between them. A max pooling layer thereafter gathers information from local features of the SDP or the inverse SDP. We have a $softmax$ output layer after pooling layer for classification in the unidirectional model RCNN.

On the basis of RCNN model, we build a bidirectional architecture BRCNN taking the SDP and the inverse SDP of a sentence as input. During the training stage of a (K+1)-relation task,

two fine-grained $softmax$ classifiers of RCNNs do a $(2K + 1)$-class classification respectively. The pooling layers of two RCNNs are concatenated and a coarse-grained $softmax$ output layer is followed to do a $(K + 1)$-class classification. The final (2K+1)-class distribution is the combination of two (2K+1)-class distributions provided by fine-grained classifiers respectively during the testing stage.

2.2 The Shortest Dependency Path

If e_1 and e_2 are two entities mentioned in the same sentence such that they are observed to be in a relationship R, the shortest path between e_1 and e_2 condenses most illuminating information for the relationship $R(e_1, e_2)$. It is because (1) if entities e_1 and e_2 are arguments of the same predicate, the shortest path between them will pass through the predicate; (2) if e_1 and e_2 belong to different predicate-argument structures that share a common argument, the shortest path will pass through this argument.

Bunescu and Mooney (2005) first used shortest dependency paths between two entities to capture the predicate-argument sequences, which provided strong evidence for relation classification. Xu et al. (2015b) captured information from the sub-paths separated by the common ancestor node of two entities in the shortest paths. However, the shortest dependency path between two entities is usually short (~4 on average) , and the common ancestor of some SDPs is e_1 or e_2, which leads to imbalance of two sub-paths.

We observe that, in the shortest dependency path, each two neighbor words w_a and w_b are linked by a dependency relation r_{ab}. The dependency relations between a governing word and its children make a difference in meaning. Besides, if we inverse the shortest dependency path, it corresponds to the same relationship with an opposite direction. For example , in Figure 1, the shortest path is composed of some sub-structure like "$burst \xrightarrow{nsubjpass} caused$". Following the above intuition, we design a bidirectional recurrent convolutional neural network, which can capture features from the local substructures and inversely at the same time.

2.3 Two-Channel Recurrent Neural Network with Long Short Term Memory Units

The recurrent neural network is suitable for modeling sequential data, as it keeps hidden state vector h, which changes with input data at each step accordingly. We make use of words and dependency relations along the SDP for relations classification (Figure 2). We call them *channels* as these information sources do not interact during recurrent propagation. Each word and dependency relation in a given sentence is mapped to a real-valued vector by looking up in a embedding table. The embeddings of words are trained on a large corpus unsupervisedly and are thought to be able to capture their syntactic and semantic information, and the embeddings of dependency relations are initialized randomly.

The hidden state h_t, for the t-th input is a function of its previous state h_{t-1} and the embedding x_t of current input. Traditional recurrent networks have a basic interaction, that is, the input is linearly transformed by a weight matrix and nonlinearly squashed by an activation function. Formally, we have

$$h_t = f(W_{in} \cdot x_t + W_{rec} \cdot h_{t-1} + b_h) \qquad (1)$$

where W_{in} and W_{rec} are weight matrices for the input and recurrent connections, respectively. b_h is a bias term for the hidden state vector, and f a non-linear activation function.

It was difficult to train RNNs to capture long-term dependencies because the gradients tend to either vanish or explode. Therefore, some more sophisticated activation function with gating units were designed. Long short term memory units are proposed in Hochreiter and Schmidhuber (1997) to overcome this problem. The main idea is to introduce an adaptive gating mechanism, which decides the degree to which LSTM units keep the previous state and memorize the extracted features of the current data input. Many LSTM variants have been proposed. We adopt in our method a variant introduced by Zaremba and Sutskever (2014). Concretely, the LSTM-based recurrent neural network comprises four components: an input gate i_t, a forget gate f_t, an output gate o_t, and a memory cell c_t.

First, we compute the values for i_t, the input gate, and g_t the candidate value for the states of

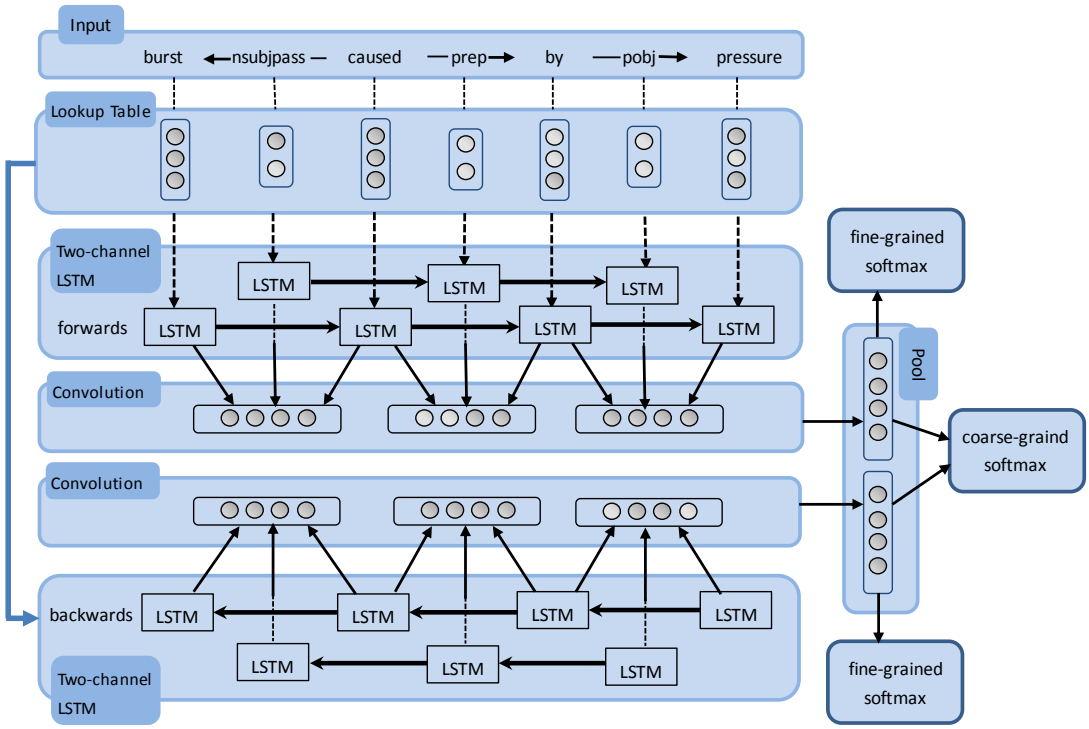


Figure 2: The overall architecture of BRCNN. Two-Channel recurrent neural networks with LSTM units pick up information along the shortest dependency path, and inversely at the same time. Convolution layers are applied to extract local features from the dependency units.

the memory cells at time t:

$$i_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \quad (2)$$

$$g_t = tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \quad (3)$$

Second, we compute the value for f_t, the activations of the memory cells' forget gates at time t:

$$f_t = \sigma(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f) \quad (4)$$

Given the value of the input gate activations i_t, the forget gate activation f_t and the candidate state value g_t, we can compute c_t the memory cells' new state at time t:

$$c_t = i_t \otimes g_t + f_t \otimes c_{t-1} \quad (5)$$

With the new state of the memory cells, we can compute the value of their output gates and, subsequently, their outputs:

$$o_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \quad (6)$$

$$h_t = o_t \otimes tanh(c_t) \quad (7)$$

In the above equations, σ denotes a sigmoid function; $\otimes$ denotes element-wise multiplication.

2.4 Bidirectional Recurrent Convolutional Neural Network

We observe that a governing word w_a and its children w_b are linked by a dependency relation r_{ab}, which makes a difference in meaning. For example, "*kills* $\xrightarrow{nsubj}$ *it*" is distinct from "*kills* $\xrightarrow{dobj}$ *it*". The shortest dependency path is composed of many substructures like "$w_a \xrightarrow{r_{ab}} w_b$", which are hereinafter referred to as "*dependency unit*". Hidden states of words and dependency relations in the SDP are obtained, utilizing two-channel recurrent neural network. The hidden states of w_a, w_b and r_{ab} are h_a, h_b and h'_{ab}, and the hidden state of the dependency unit d_{ab} is $[h_a \oplus h'_{ab} \oplus h_b]$, where $\oplus$ denotes concatenate operation. Local features L_{ab} for the dependency unit d_{ab} can be extracted, utilizing a convolution layer upon the two-channel recurrent neural network . Formally, we have

$$L_{ab} = f(W_{con} \cdot [h_a \oplus h'_{ab} \oplus h_b] + b_{con}) \quad (8)$$

where W_{con} is the weight matrix for the convolution layer and b_{con} is a bias term for the hidden state vector. f is a non-linear activation function(*tanh* is used in our model). A pooling layer thereafter gather global information G from

local features of dependency units, which is defined as

$$G = \max_{d=1}^{D} L_d \tag{9}$$

where the max function is an element-wise function, and D is the number of dependency units in the SDP.

The advantage of two-channel recurrent neural network is the ability to better capture the contextual information, adaptively accumulating the context information the whole path via memory units. However, the recurrent neural network is a biased model, where later inputs are more dominant than earlier inputs. It could reduce the effectiveness when it is used to capture features for relation classification, for the entities are located at both ends of SDP and key components could appear anywhere in a SDP rather than at the end. We tackle the problem with Bidirectional Convolutional Recurrent Neural Network.

On the basis of observation, we make a hypothesis that SDP is a symmetrical structure. For example, if there is a forward shortest path $\overrightarrow{S}$ which corresponds to relation $R_x(e_1, e_2)$, the backward shortest path $\overleftarrow{S}$ can be obtained by inversing $\overrightarrow{S}$, and $\overleftarrow{S}$ corresponds to $R_x(e_2, e_1)$, and both $\overrightarrow{S}$ and $\overleftarrow{S}$ correspond to relation R_x.

As shown in Figure 2, two RCNNs pick up information along $\overrightarrow{S}$ and $\overleftarrow{S}$, obtaining global representations $\overrightarrow{G}$ and $\overleftarrow{G}$. A representation with bidirectional information is obtained by concatenating $\overrightarrow{G}$ and $\overleftarrow{G}$. A coarse-grained $softmax$ classifier is used to predict a (K+1)-class distribution y. Formally,

$$y = softmax(W_c \cdot [\overleftarrow{G} \oplus \overrightarrow{G}] + b_c) \tag{10}$$

Where W_c is the transformation matrix and b_c is the bias vector. Coarse-grained classifier makes use of representation with bidirectional information ignoring the direction of relations, which learns the inherent correlation between the same directed relations with opposite directions, such as $R_x(e_1, e_2)$ and $R_x(e_2, e_1)$.

Two fine-grained $softmax$ classifiers are applied to $\overrightarrow{G}$ and $\overleftarrow{G}$ with linear transformation to give the (2K+1)-class distribution $\overrightarrow{y}$ and $\overleftarrow{y}$ respectively. Formally,

$$\overrightarrow{y} = softmax(W_f \cdot \overrightarrow{G} + b_f) \tag{11}$$

$$\overleftarrow{y} = softmax(W_f \cdot \overleftarrow{G} + b_f) \tag{12}$$

where W_f is the transformation matrix and b_f is the bias vector. Classifying $\overrightarrow{S}$ and $\overleftarrow{S}$ respecitvely at the same time can strengthen the model ability to judge the direction of relations.

2.5 Training Objective

The $(K + 1)$-class $softmax$ classifier is used to estimate probability that $\overrightarrow{S}$ and $\overleftarrow{S}$ are of relation R . The two $(2K + 1)$-class $softmax$ classifiers are used to estimate the probability that $\overrightarrow{S}$ and $\overleftarrow{S}$ are of relation $\overrightarrow{R}$ and $\overleftarrow{R}$ respectively. For a single data sample, the training objective is the penalized cross-entropy of three classifiers, given by

$$\begin{aligned} J = &\sum_{i=1}^{2K+1} \overrightarrow{t}_i \log \overrightarrow{y}_i + \sum_{i=1}^{2K+1} \overleftarrow{t}_i \log \overleftarrow{y}_i \\ &+ \sum_{i=1}^{K} t_i \log y_i + \lambda \cdot \|\theta\|^2 \end{aligned} \tag{13}$$

where $t \in \mathbb{R}^{K+1}$, $\overrightarrow{t}$ and $\overleftarrow{t} \in \mathbb{R}^{2K+1}$, indicating the one-hot represented ground truth. y, $\overrightarrow{y}$ and $\overleftarrow{y}$ are the estimated probabilities for each class described in section 2.4. θ is the set of model parameters to be learned, and λ is a regularization coefficient.

For decoding (predicting the relation of an unseen sample), the bidirectional model provides the (2K+1)-class distribution $\overrightarrow{y}$ and $\overleftarrow{y}$. The final (2K+1)-class distribution y_{test} becomes the combination of $\overrightarrow{y}$ and $\overleftarrow{y}$. Formally,

$$y_{test} = \alpha \cdot \overrightarrow{y} + (1 - \alpha) \cdot z(\overleftarrow{y}) \tag{14}$$

where α is the fraction of the composition of distributions, which is set to the value 0.65 according to the performance on validation dataset. During the implementation of BRCNN, elements in two class distributions at the same position are not corresponding, e.g. Cause-Effect(e_1, e_2) in $\overrightarrow{y}$ should correspond to Cause-Effect(e_2, e_1) in $\overleftarrow{y}$. We apply a function z to transform $\overleftarrow{y}$ to a corresponding forward distribution like $\overrightarrow{y}$.

3 Experiments

3.1 Dataset

We evaluated our BRCNN model on the SemEval-2010 Task 8 dataset, which is an established benchmark for relation classification (Hendrickx et al., 2010). The dataset contains 8000 sentences for training, and 2717 for testing. We split 800 samples out of the training set for validation.

Classifier	Additional Information	F_1
SVM (Rink and Harabagiu, 2010)	POS, WordNet, Prefixes and other morphological features, dependency parse, Levin classed, PropBank, FanmeNet, NomLex-Plus, Google *n*-gram, paraphrases, TextRunner	82.2
RNN (Socher et al., 2011)	Word embeddings	74.8
	+ POS, NER, WordNet	77.6
MVRNN (Socher et al., 2012)	Word embeddings	79.1
	+ POS, NER, WordNet	82.4
CNN (Zeng et al., 2014)	Word embeddings	69.7
	+ word position embeddings, WordNet	82.7
FCM (Yu et al., 2014)	Word embeddings	80.6
	+ dependency parsing, NER	83.0
CR-CNN (dos Santos et al., 2015)	Word embeddings	82.8
	+ word position embeddings	84.1
SDP-LSTM (Xu et al., 2015b)	Word embeddings	82.4
	+ POS + GR + WordNet embeddings	83.7
DepNN (Liu et al., 2015)	Word embeddings, WordNet	83.0
	Word embeddings, NER	83.6
depLCNN (Xu et al., 2015a)	Word embeddings, WordNet, word around nominals	83.7
	+ negative sampling from NYT dataset	85.6
BRCNN (Our Model)	Word embeddings	85.4
	+ POS, NER, WordNet embeddings	**86.3**

Table 1: Comparison of relation classification systems.

The dataset has (K+1)=10 distinguished relations, as follows.

- Cause-Effect
- Component-Whole
- Content-Container
- Entity-Destination
- Entity-Origin
- Message-Topic
- Member-Collection
- Instrument-Agency
- Product-Agency
- Other

The former K=9 relations are directed, whereas the Other class is undirected, we have (2K+1)=19 different classes for 10 relations. All baseline systems and our model use the official macro-averaged F_1-score to evaluate model performance. This official measurement excludes the Other relation.

3.2 Hyperparameter Settings

In our experiment, word embeddings were 200-dimensional as used in (Yu et al., 2014), trained on Gigaword with word2vec (Mikolov et al., 2013). Embeddings of relation are 50-dimensional and initialized randomly. The hidden layers in each channel had the same number of units as their embeddings (200 or 50). The convolution layer was 200-dimensional. The above values were chosen according to the performance on the validation dataset.

As we can see in Figure 1, dependency relation r "$\xrightarrow{prep}$" in $\overrightarrow{S}$ becomes r^{-1} "$\xleftarrow{prep}$" in $\overleftarrow{S}$. Experiment results show that, the performance of BRCNN is improved if r and r^{-1} correspond to different relations embeddings rather than a same embedding. We notice that dependency relations contain much fewer symbols than the words contained in the vocabulary, and we initialize the embeddings of dependency relations randomly for they can be adequately tuned during supervised training.

We add l_2 penalty for weights with coefficient 10^{-5}, and dropout of embeddings with rate 0.5. We applied AdaDelta for optimization (Zeiler, 2012), where gradients are computed with an adaptive learning rate.

3.3 Results

Table 1 compares our BRCNN model with other state-of-the-art methods. The first entry in the table presents the highest performance achieved by traditional feature-based methods. Rink and Harabagiu. (2010) fed a variety of handcrafted features to the SVM classifier and achieve an F_1-score of 82.2%.

Recent performance improvements on this dataset are mostly achieved with the help of neural networks. Socher et al. (2012) built a recursive neural network on the constituency tree and achieved a comparable performance with Rink and Harabagiu. (2010). Further, they extended their recursive network with matrix-vector interaction and elevated the F_1 to 82.4%. Xu et al. (2015b) first introduced a type of gated recurrent neural network (LSTM) into this task and raised the F_1-score to 83.7%.

From the perspective of convolution, Zeng et al. (2014) constructed a CNN on the word sequence; they also integrated word position embeddings, which helped a lot on the CNN architecture. dos Santos et al. (2015) proposed a similar CNN model, named CR-CNN, by replacing the common $softmax$ cost function with a ranking-based cost function. By diminishing the impact of the Other class, they have achieved an F_1-score of 84.1%. Along the line of CNNs, Xu et al. (2015a) designed a simple negative sampling method, which introduced additional samples from other corpora like the NYT dataset. Doing so greatly improved the performance to a high F_1-score of 85.6%. Liu et al. (2015) proposed a convolutional neural network with a recursive neural network designed to model the subtrees, and achieve an F_1-score of 83.6%.

Without the use of neural networks, Yu et al. (2014) proposed a Feature-based Compositional Embedding Model (FCM), which combined unlexicalized linguistic contexts and word embeddings. They achieved an F_1-score of 83.0%.

We make use of three types of information to improve the performance of BRCNN: POS tags, NER features and WordNet hypernyms. Our proposed BRCNN model yields an F_1-score of 86.3%, outperforming existing competing approaches. Without using any human-designed features, our model still achieve an F_1-score of 85.4%, while the best performance of state-of-the-art methods is 84.1% (dos Santos et al., 2015).

3.4 Analysis

Table 2 compares our RCNN model with CNNs and RNNs.

Model	F_1
CNN	81.8
LSTM	76.6
Two-channel LSTM	81.5
RCNN	**82.4**

Table 2: Comparing RCNN with CNNs and RNNS.

For a fair comparison, hyperparameters are set according to the performance on validation dataset as BRCNN . CNN with embeddings of words, positions and dependency relations as input achieves an F_1-score of 81.8%. LSTM with word embeddings as input only achieves an F_1-score of 76.6%, which proves that dependency relations in SDPs play an important role in relation classification. Two-channel LSTM concatenates the pooling layers of words and dependency relations along the shortest dependency path, achieves an F_1-score of 81.5% which is still lower than CNN. RCNN captures features from dependency units by combining the advantages of CNN and RNN, and achieves an F_1-score of 82.4%.

Model	**Input**	F_1
RCNN	$\overrightarrow{S}$ of all relations	82.4
Bi-RCNN	$\overrightarrow{S}$ and $\overleftarrow{S}$ of all relations	81.2
Bi-RCNN	$\overrightarrow{S}$ and $\overleftarrow{S}$ of directed relations , $\overrightarrow{S}$ of Other	84.9
BRCNN	$\overrightarrow{S}$ and $\overleftarrow{S}$ of directed relations, $\overrightarrow{S}$ of Other	**85.4**

Table 3: Comparing different variants of our model.

Bi-RCNN is a variant of BRCNN, which doesn't have the coarse-grained classifier. $\overrightarrow{S}$ and $\overleftarrow{S}$ are shortest dependency paths described in section 2.4. As shown in Table 3, if we inverted the SDP of all relations as input, we observe a performance degradation of 1.2% compared with RCNN. As mentioned in section 3.1, the SemEval-2010 task 8 dataset contains an undirected class Other in addition to 9 directed relations(18 classes). For bidirectional model, it is natural that the inversed Other relation is also in

the Other class itself. However, the class Other is used to indicate that relation between two nominals dose not belong to any of the 9 directed classes. Therefore, the class Other is very noisy since it groups many different types of relations with different directions.

On the basis of the analysis above, we only inverse the SDP of directed relations. A significant improvement is observed and Bi-RCNN achieves an F_1-score of 84.9%. This proves bidirectional representations provide more useful information to classify directed relations. We can see that our model still benefits from the coarse-grained classification, which can help our model learn inherent correlation between directed relations with opposite directions. Compared with Bi-RCNN classifying $\overrightarrow{S}$ and $\overleftarrow{S}$ into 19 classes separately, BRCNN also conducts a 10 classes (9 directed relations and Other) classification and improves 0.5% in F_1-score. Beyond the relation classification task, we believe that our bidirectional method is general technique, which is not restricted in a specific dataset and has the potential to benefit other NLP tasks.

4 Related Work

Relation classification is an important topic in NLP. Traditional Methods for relation classification mainly fall into three classes: feature-based, kernel-based and neural network-based.

In feature-based approaches, different types of features are extracted and fed into a classifier. Generally, three types of features are often used. Lexical features concentrate on the entities of interest, e.g., POS. Syntactic features include chunking, parse trees, etc. Semantic features are exemplified by the concept hierarchy, entity class. Kambhatla (2004) used a maximum entropy model for feature combination. Rink and Harabagiu (2010) collected various features, including lexical, syntactic as well as semantic features.

In kernel based methods, similarity between two data samples is measured without explicit feature representation. Bunescu and Mooney (2005) designed a kernel along the shortest dependency path between two entities by observing that the relation strongly relies on SDPs. Wang (2008) provided a systematic analysis of several kernels and showed that relation extraction can benefit from combining convolution kernel and syntactic features. Plank and Moschitti (2013) combined structural information and semantic information in a tree kernel. One potential difficulty of kernel methods is that all data information is completely summarized by the kernel function, and thus designing an effective kernel becomes crucial.

Recently, deep neural networks are playing an important role in this task. Socher et al. (2012) introduced a recursive neural network model that assigns a matrix-vector representation to every node in a parse tree, in order to learn compositional vector representations for sentences of arbitrary syntactic type and length.

Convolutional neural works are widely used in relation classification. Zeng et al. (2014) proposed an approach for relation classification where sentence-level features are learned through a CNN, which has word embedding and position features as its input. In parallel, lexical features were extracted according to given nouns. dos Santos et al. (2015) tackled the relation classification task using a convolutional neural network and proposed a new pairwise ranking loss function, which achieved the state-of-the-art result in SemEval-2010 Task 8.

Yu et al. (2014) proposed a Factor-based Compositional Embedding Model (FCM) by deriving sentence-level and substructure embeddings from word embeddings, utilizing dependency trees and named entities. It achieved slightly higher accuracy on the same dataset than Zeng et al. (2014), but only when syntactic information is used.

Nowadays, many works concentrate on extracting features from the SDP based on neural networks. Xu et al. (2015a) learned robust relation representations from SDP through a CNN, and proposed a straightforward negative sampling strategy to improve the assignment of subjects and objects. Liu et al. (2015) proposed a recursive neural network designed to model the subtrees, and CNN to capture the most important features on the shortest dependency path. Xu et al. (2015b) picked up heterogeneous information along the left and right sub-path of the SDP respectively, leveraging recurrent neural networks with long short term memory units. We propose BRCNN to model the SDP, which can pick up bidirectional information with a combination of LSTM and CNN.

5 Conclusion

In this paper, we proposed a novel bidirectional neural network BRCNN, to improve the performance of relation classification. The BRCNN model, consisting of two RCNNs, learns features along SDP and inversely at the same time. Information of words and dependency relations are used utilizing a two-channel recurrent neural network with LSTM units. The features of dependency units in SDP are extracted by a convolution layer.

We demonstrate the effectiveness of our model by evaluating the model on SemEval-2010 relation classification task. RCNN achieves a better performance at learning features along the shortest dependency path, compared with some common neural networks. A significant improvement is observed when BRCNN is used, outperforming state-of-the-art methods.

6 Acknowledgements

Our work is supported by National Natural Science Foundation of China (No.61370117 & No. 61433015) and Major National Social Science Fund of China (No. 12&ZD227).

References

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language*, pages 724–731.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference*, pages 626–634.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions. Association for Computational Linguistics*, pages pages 94–99.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. In *Neural computation*, pages 1735–1780.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 22. Association for Computational Linguistics.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Joint Conference on Natural Language Processing and the 7th International Joint Conference on Natural Language Processing*, pages 285–290.

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

Barbara Plank and Alessandro Moschitti. 2013. Embeddings semantic similarity in tree kernels for domain adaption of relation extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1498–1507.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 151–161.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Mengqiu Wang. 2008. A re-examination of dependency path kernels for relation extraction. In *Proceedings of the Third International Joint Conference on Natural Language Processing*, pages 841–846.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing,*, pages 1785–1794.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*, pages 95–101.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *arXiv preprint arXiv:1410.4615*.

Mathew D. Zeiler. 2012. An adaptive learning rate method. In *arXiv preprint at arXiv:1212.5701*.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2335–2344.

Sentence Rewriting for Semantic Parsing

Bo Chen Le Sun Xianpei Han Bo An
State Key Laboratory of Computer Sciences
Institute of Software, Chinese Academy of Sciences, China.
{chenbo, sunle, xianpei, anbo}@nfs.iscas.ac.cn

Abstract

A major challenge of semantic parsing is the vocabulary mismatch problem between natural language and target ontology. In this paper, we propose a sentence rewriting based semantic parsing method, which can effectively resolve the mismatch problem by rewriting a sentence into a new form which has the same structure with its target logical form. Specifically, we propose two sentence-rewriting methods for two common types of mismatch: a dictionary-based method for 1-N mismatch and a template-based method for N-1 mismatch. We evaluate our sentence rewriting based semantic parser on the benchmark semantic parsing dataset – WEBQUESTIONS. Experimental results show that our system outperforms the base system with a 3.4% gain in F1, and generates logical forms more accurately and parses sentences more robustly.

1 Introduction

Semantic parsing is the task of mapping natural language sentences into logical forms which can be executed on a knowledge base (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Kate and Mooney, 2006; Wong and Mooney, 2007; Lu et al., 2008; Kwiatkowksi et al., 2010). Figure 1 shows an example of semantic parsing. Semantic parsing is a fundamental technique of natural language understanding, and has been used in many applications, such as question answering (Liang et al., 2011; He et al., 2014; Zhang et al., 2016) and information extraction (Krishnamurthy and Mitchell, 2012; Choi et al., 2015; Parikh et al., 2015).

Semantic parsing, however, is a challenging task. Due to the variety of natural language expressions, the same meaning can be expressed using different sentences. Furthermore, because logical forms depend on the vocabulary of target-ontology, a sentence will be parsed into different logical forms when using different ontologies. For example, in below the two sentences s_1 and s_2 express the same meaning, and they both can be parsed into the two different logical forms lf_1 and lf_2 using different ontologies.

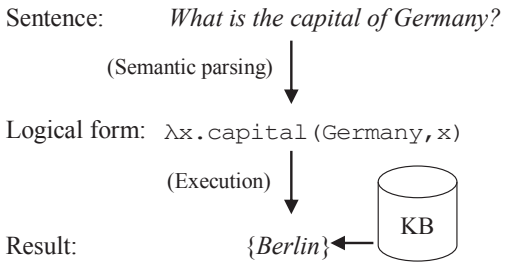


Figure 1: An example of semantic parsing.

s_1	*What is the population of Berlin?*
s_2	*How many people live in Berlin?*
lf_1	λx.population(Berlin,x)
lf_2	count(λx.person(x)∧live(x,Berlin))

Based on the above observations, one major challenge of semantic parsing is the structural mismatch between a natural language sentence and its target logical form, which are mainly raised by the vocabulary mismatch between natural language and ontologies. Intuitively, if a sentence has the same structure with its target logical form, it is easy to get the correct parse, e.g., a semantic parser can easily parse s_1 into lf_1 and s_2 into lf_2. On the contrary, it is difficult to parse a sentence into its logic form when they have different structures, e.g., $s_1 \rightarrow lf_2$ or $s_2 \rightarrow lf_1$.

To resolve the vocabulary mismatch problem,

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 766–777,
Berlin, Germany, August 7-12, 2016.

(a) An example using traditional method

s_0 : *What is the name of Sonia Gandhis daughter?*
l_0 : λ`x.child(S.G.,x)`
r_0 : {*Rahul Gandhi (Wrong answer), Priyanka Vadra*}

(b) An example using our method

s_0 : *What is the name of Sonia Gandhis daughter?*
s_1 : *What is the name of Sonia Gandhis female child?*
l_1 : λ`x.child(S.G.,x)`$\wedge$`gender(x,female)`
r_1 : {*Priyanka Vadra*}

Table 1: Examples of (a) sentences s_0, possible logical form l_0 from traditional semantic parser, result r_0 for the logical form l_0; (b) possible sentence s_1 from rewriting for the original sentence s_0, possible logical form l_1 for sentence s_1, result r_1 for l_1. *Rahul Gandhi* is a wrong answer, as he is the son of *Sonia Gandhi*.

this paper proposes a sentence rewriting approach for semantic parsing, which can rewrite a sentence into a form which will have the same structure with its target logical form. Table 1 gives an example of our rewriting-based semantic parsing method. In this example, instead of parsing the sentence "*What is the name of Sonia Gandhis daughter?*" into its structurally different logical form `childOf.S.G.`$\wedge$`gender.female` directly, our method will first rewrite the sentence into the form "*What is the name of Sonia Gandhis female child?*", which has the same structure with its logical form, then our method will get the logical form by parsing this new form. In this way, the semantic parser can get the correct parse more easily. For example, the parse obtained through traditional method will result in the wrong answer "*Rahul Gandhi*", because it cannot identify the vocabulary mismatch between "*daughter*" and `child`$\wedge$`female`[1]. By contrast, by rewriting "*daughter*" into "*female child*", our method can resolve this vocabulary mismatch.

Specifically, we identify two common types of vocabulary mismatch in semantic parsing:

1. 1-N mismatch: a simple word may correspond to a compound formula. For example, the word "*daughter*" may correspond to the compound formula `child`$\wedge$`female`.

2. N-1 mismatch: a logical constant may correspond to a complicated natural language expression, e.g., the formula `population` can be expressed using many phrases such as "*how many people*" and "*live in*".

[1] In this paper, we may simplify logical forms for readability, e.g., `female` for `gender.female`.

To resolve the above two vocabulary mismatch problems, this paper proposes two sentence rewriting algorithms: One is a dictionary-based sentence rewriting algorithm, which can resolve the 1-N mismatch problem by rewriting a word using its explanation in a dictionary. The other is a template-based sentence rewriting algorithm, which can resolve the N-1 mismatch problem by rewriting complicated expressions using paraphrase template pairs.

Given the generated rewritings of a sentence, we propose a ranking function to jointly choose the optimal rewriting and the correct logical form, by taking both the rewriting features and the semantic parsing features into consideration.

We conduct experiments on the benchmark WEBQUESTIONS dataset (Berant et al., 2013). Experimental results show that our method can effectively resolve the vocabulary mismatch problem and achieve accurate and robust performance.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 describes our sentence rewriting method for semantic parsing. Section 4 presents the scoring function which can jointly ranks rewritings and logical forms. Section 5 discusses experimental results. Section 6 concludes this paper.

2 Related Work

Semantic parsing has attracted considerable research attention in recent years. Generally, semantic parsing methods can be categorized into synchronous context free grammars (SCFG) based methods (Wong and Mooney, 2007; Arthur et al., 2015; Li et al., 2015), syntactic structure based methods (Ge and Mooney, 2009; Reddy et al., 2014; Reddy et al., 2016), combinatory categorical grammars (CCG) based methods (Zettlemoyer and Collins, 2007; Kwiatkowksi et al., 2010; Kwiatkowski et al., 2011; Krishnamurthy and Mitchell, 2014; Wang et al., 2014; Artzi et al., 2015), and dependency-based compositional semantics (DCS) based methods (Liang et al., 2011; Berant et al., 2013; Berant and Liang, 2014; Berant and Liang, 2015; Pasupat and Liang, 2015; Wang et al., 2015).

One major challenge of semantic parsing is how to scale to open-domain situation like Freebase and Web. A possible solution is to learn lexicons from large amount of web text and a knowledge base using a distant supervised method (Krishna-

murthy and Mitchell, 2012; Cai and Yates, 2013a; Berant et al., 2013). Another challenge is how to alleviate the burden of annotation. A possible solution is to employ distant-supervised techniques (Clarke et al., 2010; Liang et al., 2011; Cai and Yates, 2013b; Artzi and Zettlemoyer, 2013), or unsupervised techniques (Poon and Domingos, 2009; Goldwasser et al., 2011; Poon, 2013).

There were also several approaches focused on the mismatch problem. Kwiatkowski et al. (2013) addressed the ontology mismatch problem (i.e., two ontologies using different vocabularies) by first parsing a sentence into a domain-independent underspecified logical form, and then using an ontology matching model to transform this underspecified logical form to the target ontology. However, their method is still hard to deal with the 1-N and the N-1 mismatch problems between natural language and target ontologies. Berant and Liang (2014) addressed the structure mismatch problem between natural language and ontology by generating a set of canonical utterances for each candidate logical form, and then using a paraphrasing model to rerank the candidate logical forms. Their method addresses mismatch problem in the reranking stage, cannot resolve the mismatch problem when constructing candidate logical forms. Compared with these two methods, we approach the mismatch problem in the parsing stage, which can greatly reduce the difficulty of constructing the correct logical form, through rewriting sentences into the forms which will be structurally consistent with their target logic forms.

Sentence rewriting (or paraphrase generation) is the task of generating new sentences that have the same meaning as the original one. Sentence rewriting has been used in many different tasks, e.g., used in statistical machine translation to resolve the word order mismatch problem (Collins et al., 2005; He et al., 2015). To our best knowledge, this paper is the first work to apply sentence rewriting for vocabulary mismatch problem in semantic parsing.

3 Sentence Rewriting for Semantic Parsing

As discussed before, the vocabulary mismatch between natural language and target ontology is a big challenge in semantic parsing. In this section, we describe our sentence rewriting algorithm for solving the mismatch problem. Specifically, we solve the 1-N mismatch problem by dictionary-based rewriting and solve the N-1 mismatch problem by template-based rewriting. The details are as follows.

Word	Logical Form	Wiktionary Explanation
son	`child⋀male`	*male child*
actress	`actor⋀female`	*female actor*
father	`parent⋀male`	*male parent*
grandaprent	`parent⋀parent`	*parent of one's parent*
brother	`sibling⋀male`	*male sibling*

Table 2: Several examples of words, their logical forms and their explanations in Wiktionary.

3.1 Dictionary-based Rewriting

In the 1-N mismatch case, a word will correspond to a compound formula, e.g., the target logical form of the word "*daughter*" is `child⋀female` (Table 2 has more examples).

To resolve the 1-N mismatch problem, we rewrite the original word ("*daughter*") into an expression ("*female child*") which will have the same structure with its target logical form (`child⋀female`). In this paper, we rewrite words using their explanations in a dictionary. This is because each word in a dictionary will be defined by a detailed explanation using simple words, which often will have the same structure with its target formula. Table 2 shows how the vocabulary mismatch between a word and its logical form can be resolved using its dictionary explanation. For instance, the word "*daughter*" is explained as "*female child*" in Wiktionary, which has the same structure as `child⋀female`.

In most cases, only common nouns will result in the 1-N mismatch problem. Therefore, in order to control the size of rewritings, this paper only rewrite the common nouns in a sentence by replacing them with their dictionary explanations. Because a sentence usually will not contain too many common nouns, the size of candidate rewritings is thus controllable. Given the generated rewritings of a sentence, we propose a sentence selection model to choose the best rewriting using multiple features (See details in Section 4).

Table 3 shows an example of the dictionary-based rewriting. In Table 3, the example sentence s contains two common nouns ("*name*" and "*daughter*"), therefore we will generate three rewritings r_1, r_2 and r_3. Among these rewritings,

s : *What is the name of Sonia Gandhis daughter?*
r_1: *What is the reputation of Sonia Gandhis daughter?*
r_2: *What is the name of Sonia Gandhis female child?*
r_3: *What is the reputation of Sonia Gandhis female child?*

Table 3: An example of the dictionary-based sentence rewriting.

the candidate rewriting r_2 is what we expected, as it has the same structure with the target logical form and doesn't bring extra noise (i.e., replacing "*name*" with its explanation "*reputation*").

For the dictionary used in rewriting, this paper uses Wiktionary. Specifically, given a word, we use its "Translations" part in the Wiktionary as its explanation. Because most of the 1-N mismatch are caused by common nouns, we only collect the explanations of common nouns. Furthermore, for polysomic words which have several explanations, we only use their most common explanations. Besides, we ignore explanations whose length are longer than 5.

3.2 Template-based Rewriting

In the N-1 mismatch case, a complicated natural language expression will be mapped to a single logical constant. For example, considering the following mapping from the natural language sentence s to its logical form lf based on Freebase ontology:

s: *How many people live in Berlin?*
lf: λx.population(Berlin,x)

where the three words: "*how many*" (count), "*people*" (people) and "*live in*" (live) will map to the predicate population together. Table 4 shows more N-1 examples.

Expression	Logical constant
how many, people, live in	population
how many, people, visit, annually	annual-visit
what money, use	currency
what school, go to	education
what language, speak, officially	official-language

Table 4: Several N-1 mismatch examples.

To resolve the N-1 mismatch problem, we propose a template rewriting algorithm, which can rewrite a complicated expression into its simpler form. Specifically, we rewrite sentences based on a set of paraphrase template pairs $P = \{(t_{i1}, t_{i2})|i = 1, 2, ..., n\}$, where each template t is a sentence with an argument slot \$y, and t_{i1} and t_{i2} are paraphrases. In this paper, we only consider single-slot templates. Table 5 shows several paraphrase template pairs.

Template 1	Template 2
How many people live in \$y	*What is the population of \$y*
What money in \$y is used	*What is the currency of \$y*
What school did \$y go to	*What is the education of \$y*
What language does \$y speak officially	*What is the official language of \$y*

Table 5: Several examples of paraphrase template pairs.

Given the template pair database and a sentence, our template-based rewriting algorithm works as follows:

1. Firstly, we generate a set of candidate templates $ST = \{st_1, st_2, ..., st_n\}$ of the sentence by replacing each named entity within it by "\$y". For example, we will generate template "*How many people live in \$y*" from the sentence "*How many people live in Berlin*".

2. Secondly, using the paraphrase template pair database, we retrieve all possible rewriting template pairs (t_1, t_2) with $t_1 \in ST$, e.g., we can retrieve template pair ("*How many people live there in \$y*", "*What is the population of \$y*" for t_2) using the above ST.

3. Finally, we get the rewritings by replacing the argument slot "\$y" in template t_2 with the corresponding named entity. For example, we get a new candidate sentence "*What is the population of Berlin*" by replacing "\$y" in t_2 with Berlin. In this way we can get the rewriting we expected, since this rewriting will match its target logical form population(Berlin).

To control the size and measure the quality of rewritings using a specific template pair, we also define several features and the similarity between template pairs (See Section 4 for details).

To build the paraphrase template pair database, we employ the method described in Fader et al. (2014) to automatically collect paraphrase template pairs. Specifically, we use the WikiAnswers paraphrase corpus (Fader et al., 2013), which contains 23 million question-clusters, and all ques-

How many people live in chembakolli?
How many people is in chembakolli?
How many people live in chembakolli india?
How many people live there chembakolli?
How many people live there in chembakolli?
What is the population of Chembakolli india?
What currency is used on St Lucia?
What is st lucia money?
What is the money used in st lucia?
What kind of money did st lucia have?
What money do st Lucia use?
Which money is used in St Lucia?

Table 6: Two paraphrase clusters from the WikiAnswers corpus.

tions in the same cluster express the same meaning. Table 6 shows two paraphrase clusters from the WikiAnswers corpus. To build paraphrase template pairs, we first replace the shared noun words in each cluster with the placeholder "$y", then each two templates in a cluster will form a paraphrase template pair. To filter out noisy template pairs, we only retain salient paraphrase template pairs whose co-occurrence count is larger than 3.

4 Sentence Rewriting based Semantic Parsing

In this section we describe our semantic rewriting based semantic parsing system. Figure 2 presents the framework of our system. Given a sentence, we first rewrite it into a set of new sentences, then we generate candidate logical forms for each new sentence using a base semantic parser, finally we score all logical forms using a scoring function and output the best logical form as the final result. In following, we first introduce the used base semantic parser, then we describe the proposed scoring function.

4.1 Base Semantic Parser

In this paper, we produce logical forms for each sentence rewritings using an agenda-based semantic parser (Berant and Liang, 2015), which is based on the lambda-DCS proposed by Liang (2013). For parsing, we use the lexicons and the grammars released by Berant et al. (2013), where lexicons are used to trigger unary and binary predicates, and grammars are used to conduct logical forms. The only difference is that we also use the composition rule to make the parser can handle complicated questions involving two binary predicates, e.g., `child.obama⊓gender.female`.

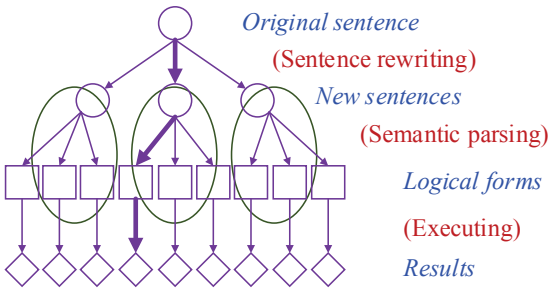


Figure 2: The framework of our sentence rewriting based semantic parsing.

For model learning and sentence parsing, the base semantic parser learned a scoring function by modeling the policy as a log-linear distribution over (partial) agenda derivations Q:

$$p_\theta(a|s) = \frac{\exp\{\phi(a)^T\theta)\}}{\sum_{a' \in A} \exp\{\phi(a')^T\theta)\}} \quad (1)$$

The policy parameters are updated as follows:

$$\theta \leftarrow \theta + \eta R(h_{target}) \sum_{t=1}^{T} \delta(h_{target}) \quad (2)$$

$$\begin{aligned} \delta_t(h) &= \nabla_\theta \log p_\theta(a_t|s_t) \\ &= \phi(a_t) - E_{p_\theta(a'_t|s_t)}[\phi(a'_t)] \end{aligned} \quad (3)$$

The reward function $R(h)$ measures the compatibility of the resulting derivation, and η is the learning rate which is set using the AdaGrad algorithm (Duchi et al., 2011). The target history h_{target} is generated from the root derivation d^* with highest reward out of the K (beam size) root derivations, using local reweighting and history compression.

4.2 Scoring Function

To select the best semantic parse, we propose a scoring function which can take both sentence rewriting features and semantic parsing features into consideration. Given a sentence x, a generated rewriting x' and the derivation d of x', we score them using follow function:

$$\begin{aligned} score(x, x', d) &= \theta \cdot \phi(x, x', d) \\ &= \theta_1 \cdot \phi(x, x') + \theta_2 \cdot \phi(x', d) \end{aligned}$$

This scoring function is decomposed into two parts: one for sentence rewriting – $\theta_1 \cdot \phi(x, x')$ and the other for semantic parsing – $\theta_2 \cdot \phi(x', d)$. Following Berant and Liang (2015), we update the parameters θ_2 of semantic parsing features as the

Input: Q/A pairs $\{(x_i, y_i) : i = 1...n\}$; Knowledge base $\mathcal{K}$; Number of sentences N; Number of iterations T.

Definitions: The function $REWRITING(x_i)$ returns a set of candidate sentences by applying sentence rewriting on sentence x; $PARSE(p_\theta, x)$ parses the sentence x based on current parameters θ, using agenda-based parsing; $CHOOSEORACLE(h_0)$ chooses the derivation with highest reward from the root of h_0; $CHOOSEORACLE(H_{target})$ chooses the derivation with highest reward from a set of derivations. $CHOOSEORACLE(h^*_{target})$ chooses the new sentence that results in derivation with highest reward.

Algorithm:

$\theta_1 \leftarrow \mathbf{0}, \theta_2 \leftarrow \mathbf{0}$
for $t = 1...T, i = 1...N$:
  $X = REWRITING(x_i)$
  for each $x'_i \in X$:
    $h_0 \leftarrow PARSE(p_\theta, x'_i)$
    $d^* \leftarrow CHOOSEORACLE(h_0)$
    $h_{target} \leftarrow PARSE(p_\theta^{+cw}, x'_i)$
  $h^*_{target} \leftarrow CHOOSEORACLE(H_{target})$
  $x'^*_i \leftarrow CHOOSEORACLE(h^*_{target})$
  $\theta_2 \leftarrow \theta_2 + \eta R(h^*_{target}) \sum_{t=1}^{T} \delta(h^*_{target})$
  $\theta_1 \leftarrow \theta_1 + \eta R(h^*_{target}) \delta(x_i, x'^*_i)$

Output: Estimated parameters θ_1 and θ_2.

Table 7: Our learning algorithm for parameter estimation from question-answer pairs.

same as (2). Similarly, the parameters θ_1 of sentence rewriting features are updated as follows:

$$\theta_1 \leftarrow \theta_1 + \eta R(h^*_{target}) \delta(x, x'^*)$$

$$\begin{aligned}\delta(x, x'^*) &= \nabla \log p_{\theta_1}(x'^* | x) \\ &= \phi(x, x'^*) - E_{p_{\theta_1}(x'|x)}[\phi(x, x')]\end{aligned}$$

where the learning rate η is set using the same algorithm in Formula (2).

4.3 Parameter Learning Algorithm

To estimate the parameters θ_1 and θ_2, our learning algorithm uses a set of question-answer pairs (x_i, y_i). Following Berant and Liang (2015), our updates for θ_1 and θ_2 do not maximize reward nor the log-likelihood. However, the reward provides a way to modulate the magnitude of the updates. Specifically, after each update, our model results in making the derivation, which has the highest reward, to get a bigger score. Table 7 presents our learning algorithm.

4.4 Features

As described in Section 4.3, our model uses two kinds of features. One for the semantic parsing module which are simply the same features described in Berant and Liang (2015). One for the sentence rewriting module these features are defined over the original sentence, the generated sentence rewritings and the final derivations:

Features for dictionary-based rewriting. Given a sentence s_0, when the new sentence s_1 is generated by replacing a word to its explanation $w \rightarrow ex$, we will generate four features: The first feature indicates the word replaced. The second feature indicates the replacement $w \rightarrow ex$ we used. The final two features are the POS tags of the left word and the right word of w in s_0.

Features for template-based rewriting. Given a sentence s_0, when the new sentence s_1 is generated through a template based rewriting $t_1 \rightarrow t_2$, we generate four features: The first feature indicates the template pair (t_1, t_2) we used. The second feature is the similarity between the sentence s_0 and the template t_1, which is calculated using the word overlap between s_0 and t_1. The third feature is the compatibility of the template pair, which is the pointwise mutual information (PMI) between t_1 and t_2 in the WikiAnswers corpus. The final feature is triggered when the target logical form only contains an atomic formula (or predicate), and this feature indicates the mapping from template t_2 to the predicate p.

5 Experiments

In this section, we assess our method and compare it with other methods.

5.1 Experimental Settings

Dataset: We evaluate all systems on the benchmark WEBQUESTIONS dataset (Berant et al., 2013), which contains 5,810 question-answer pairs. All questions are collected by crawling the Google Suggest API, and their answers are obtained using Amazon Mechanical Turk. This dataset covers several popular topics and its questions are commonly asked on the web. According to Yao (2015), 85% of questions can be answered by predicting a single binary relation. In our experiments, we use the standard train-test split (Berant et al., 2013), i.e., 3,778 questions (65%) for training and 2,032 questions (35%) for testing, and divide the training set into 3 random 80%-20% splits for development.

Furthermore, to verify the effectiveness of our method on solving the vocabulary mismatch problem, we manually select 50 mismatch test examples from the WEBQUESTIONS dataset, where

all sentences have different structure with their target logical forms, e.g., "*Who is keyshia cole dad?*" and "*What countries have german as the official language?*".

System Settings: In our experiments, we use the Freebase Search API for entity lookup. We load Freebase using Virtuoso, and execute logical forms by converting them to SPARQL and querying using Virtuoso. We learn the parameters of our system by making three passes over the training dataset, with the beam size $K = 200$, the dictionary rewriting size $K_D = 100$, and the template rewriting size $K_T = 100$.

Baselines: We compare our method with several traditional systems, including semantic parsing based systems (Berant et al., 2013; Berant and Liang, 2014; Berant and Liang, 2015; Yih et al., 2015), information extraction based systems (Yao and Van Durme, 2014; Yao, 2015), machine translation based systems (Bao et al., 2014), embedding based systems (Bordes et al., 2014; Yang et al., 2014), and QA based system (Bast and Haussmann, 2015).

Evaluation: Following previous work (Berant et al., 2013), we evaluate different systems using the fraction of correctly answered questions. Because golden answers may have multiple values, we use the average F1 score as the main evaluation metric.

5.2 Experimental Results

Table 8 provides the performance of all base-lines and our method. We can see that:

1. Our method achieved competitive performance: Our system outperforms all baselines and get the best F1-measure of 53.1 on WEBQUESTIONS dataset.

2. Sentence rewriting is a promising technique for semantic parsing: By employing sentence rewriting, our system gains a 3.4% F1 improvement over the base system we used (Berant and Liang, 2015).

3. Compared to all baselines, our system gets the highest precision. This result indicates that our parser can generate more-accurate logical forms by sentence rewriting. Our system also achieves the second highest recall, which is a competitive performance. Interestingly, both the two systems with the highest recall (Bast and Haussmann, 2015; Yih et al., 2015) rely on extra-techniques such as entity linking and relation matching.

System	Prec.	Rec.	F1 (avg)
Berant et al., 2013	48.0	41.3	35.7
Yao and Van-Durme, 2014	51.7	45.8	33.0
Berant and Liang, 2014	40.5	46.6	39.9
Bao et al., 2014	–	–	37.5
Bordes et al., 2014a	–	–	39.2
Yang et al., 2014	–	–	41.3
Bast and Haussmann, 2015	49.8	60.4	49.4
Yao, 2015	52.6	54.5	44.3
Berant and Liang, 2015	50.5	55.7	49.7
Yih et al., 2015	52.8	**60.7**	52.5
Our approach	**53.7**	60.0	**53.1**

Table 8: The results of our system and recently published systems. The results of other systems are from either original papers or the standard evaluation web.

The effectiveness on mismatch problem. To analyze the commonness of mismatch problem in semantic parsing, we randomly sample 500 questions from the training data and do manually analysis, we found that 12.2% out of the sampled questions have mismatch problems: 3.8% out of them have 1-N mismatch problem and 8.4% out of them have N-1 mismatch problem.

To verify the effectiveness of our method on solving the mismatch problem, we conduct experiments on the 50 mismatch test examples and Table 9 shows the performance. We can see that our system can effectively resolve the mismatch between natural language and target ontology: compared to the base system, our system achieves a significant 54.5% F1 im-provement.

System	Prec.	Rec.	F1 (avg)
Base system	31.4	43.9	29.4
Our system	**83.3**	**92.3**	**83.9**

Table 9: The results on the 50 mismatch test dataset.

When scaling a semantic parser to open-domain situation or web situation, the mismatch problem will be more common as the ontology and language complexity increases (Kwiatkowski et al., 2013). Therefore we believe the sentence rewriting method proposed in this paper is an important technique for the scalability of semantic parser.

The effect of different rewriting algorithms. To analyze the contribution of different rewriting methods, we perform experiments using different sentence rewriting methods and the results are presented in Table 10. We can see that:

Method	Prec.	Rec.	F1 (avg)
base	49.8	55.3	49.1
+ dictionary SR (only)	51.6	57.5	50.9
+ template SR (only)	52.9	59.0	52.3
+ both	**53.7**	**60.0**	**53.1**

Table 10: The results of the base system and our systems on the 2032 test questions.

1. Both sentence rewriting methods improved the parsing performance, they resulted in 1.8% and 3.2% F1 improvements respectively[2].

2. Compared with the dictionary-based rewriting method, the template-based rewriting method can achieve higher performance improvement. We believe this is because N-1 mismatch problem is more common in the WEBQUESTIONS dataset.

3. The two rewriting methods are good complementary of each other. The semantic parser can achieve a higher performance improvement when using these two rewriting methods together.

The effect on improving robustness. We found that the template-based rewriting method can greatly improve the robustness of the base semantic parser. Specially, the template-based method can rewrite similar sentences into a uniform template, and the (template, predicate) feature can provide additional information to reduce the uncertainty during parsing. For example, using only the uncertain alignments from the words "*people*" and "*speak*" to the two predicates `official_language` and `language_spoken`, the base parser will parse the sentence "*What does jamaican people speak?*" into the incorrect logical form `official_language.jamaican` in our experiments, rather than into the correct form `language_spoken.jamaican` (See the final example in Table 11). By exploiting the alignment from the template "*what language does $y people speak*" to the predicate , our system can parse the above sentence correctly.

The effect on OOV problem. We found that the sentence rewriting method can also provide extra profit for solving the OOV problem. Traditionally, if a sentence contains a word which is not covered by the lexicon, it will cannot be correctly parsed. However, with the help of sentence rewriting, we may rewrite the OOV words into the words which are covered by our lexicons. For example, in Table 11 the 3rd question "*What are some of the traditions of islam?*" cannot be correctly parsed as the lexicons dont cover the word "*tradition*". Through sentence rewriting, we can generate a new sentence "*What is of the religion of islam?*", where all words are covered by the lexicons, in this way the sentence can be correctly parsed.

[2]Our base system yields a slight drop in accuracy compared to the original system (Berant and Liang, 2015), as we parallelize the learning algorithm, and the order of the data for updating the parameter is different to theirs.

O	*Who is willow smith mom name?*
R	*Who is willow smith female parent name?*
LF	`parentOf.willow_smith⊓gender.female`
O	*Who was king henry viii son?*
R	*Who was king henry viii male child?*
LF	`childOf.king_henry⊓gender.male`
O	*What are some of the traditions of islam?*
R	*What is of the religion of islam?*
LF	`religionOf.islam`
O	*What does jamaican people speak?*
R	*What language does jamaican people speak?*
LF	`language_spoken.jamaica`

Table 11: Examples which our system generates more accurate logical form than the base semantic parser. **O** is the original sentence; **R** is the generated sentence from sentence rewriting (with the highest score for the model, including rewriting part and parsing part); **LF** is the target logical form.

5.3 Error Analysis

To better understand our system, we conduct error analysis on the parse results. Specifically, we randomly choose 100 questions which are not correctly answered by our system. We found that the errors are mainly raised by following four reasons (See Table 12 for detail):

Reason	#(Ratio)	Sample Example
Label issue	38	*What band was george clinton in?*
N-ary predicate($n > 2$)	31	*What year did the seahawks win the superbowl?*
Temporal clause	15	*Who was the leader of the us during wwii?*
Superlative	8	*Who was the first governor of colonial south carolina?*
Others	8	*What is arkansas state capitol?*

Table 12: The main reasons of parsing errors, the ratio and an example for each reason are also provided.

The first reason is the label issue. The main label issue is incompleteness, i.e., the answers of a question may not be labeled completely. For example, for the question "*Who does nolan ryan play for?*", our system returns 4 correct teams but the golden answer only contain 2 teams. One another label issue is the error labels. For example, the gold answer of the question "*What state is barack obama from?*" is labeled as "*Illinois*", however, the correct answer is "*Hawaii*".

The second reason is the n-ary predicate problem ($n > 2$). Currently, it is hard for a parser to conduct the correct logical form of n-ary predicates. For example, the question "*What year did the seahawks win the superbowl?*" describes an n-ary championship event, which gives the championship and the champion of the event, and expects the season. We believe that more research attentions should be given on complicated cases, such as the n-ary predicates parsing.

The third reason is temporal clause. For example, the question "*Who did nasri play for before arsenal?*" contains a temporal clause "*before*". We found temporal clause is complicated and makes it strenuous for the parser to understand the sentence.

The fourth reason is superlative case, which is a hard problem in semantic parsing. For example, to answer "*What was the name of henry viii first wife?*", we should choose the first one from a list ordering by time. Unfortunately, it is difficult for the current parser to decide what to be ordered and how to order.

There are also many other miscellaneous error cases, such as spelling error in the question, e.g., "*capitol*" for "*capital*", "*mary*" for "*marry*".

6 Conclusions

In this paper, we present a novel semantic parsing method, which can effectively deal with the mismatch between natural language and target ontology using sentence rewriting. We resolve two common types of mismatch (i) one word in natural language sentence vs one compound formula in target ontology (1-N), (ii) one complicated expression in natural language sentence vs one formula in target ontology (N-1). Then we present two sentence rewriting methods, dictionary-based method for 1-N mismatch and template-based method for N-1 mismatch. The resulting system significantly outperforms the base system on the WEBQUESTIONS dataset.

Currently, our approach only leverages simple sentence rewriting methods. In future work, we will explore more advanced sentence rewriting methods. Furthermore, we also want to employ sentence rewriting techniques for other challenges in semantic parsing, such as the spontaneous, unedited natural language input, etc.

Acknowledgments

We sincerely thank the reviewers for their valuable comments and suggestions. This work is supported by the National High Technology Development 863 Program of China under Grants no. 2015AA015405, and the National Natural Science Foundation of China under Grants no. 61433015, 612722324 and 61572477.

References

Philip Arthur, Graham Neubig, Sakriani Sakti, Tomoki Toda, and Satoshi Nakamura. 2015. Semantic parsing of ambiguous input through paraphrasing and verification. *Transactions of the Association for Computational Linguistics*, 3:571–584.

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

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1699–1710, Lisbon, Portugal, September. Association for Computational Linguistics.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976, Baltimore, Maryland, June. Association for Computational Linguistics.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*, pages 1431–1440.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland, June. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

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

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar, October. Association for Computational Linguistics.

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

Qingqing Cai and Alexander Yates. 2013b. Semantic parsing freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1311–1320, Beijing, China, July. Association for Computational Linguistics.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 18–27, Uppsala, Sweden, July. Association for Computational Linguistics.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan, June. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12:2121–2159, July.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria, August. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, pages 1156–1165, New York, NY, USA. ACM.

Ruifang Ge and Raymond Mooney. 2009. Learning a compositional semantic parser using an existing syntactic parser. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 611–619, Suntec, Singapore, August. Association for Computational Linguistics.

Dan Goldwasser, Roi Reichart, James Clarke, and Dan Roth. 2011. Confidence driven unsupervised semantic parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1486–1495, Portland, Oregon, USA, June. Association for Computational Linguistics.

Shizhu He, Kang Liu, Yuanzhe Zhang, Liheng Xu, and Jun Zhao. 2014. Question answering over linked data using first-order logic. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1092–1103, Doha, Qatar, October. Association for Computational Linguistics.

He He, Alvin Grissom II, John Morgan, Jordan Boyd-Graber, and Hal Daumé III. 2015. Syntax-based rewriting for simultaneous machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 55–64, Lisbon, Portugal, September. Association for Computational Linguistics.

Rohit J. Kate and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 913–920, Sydney, Australia, July. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 754–765, Jeju Island, Korea, July. Association for Computational Linguistics.

Jayant Krishnamurthy and Tom M. Mitchell. 2014. Joint syntactic and semantic parsing with combinatory categorial grammar. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1188–1198, Baltimore, Maryland, June. Association for Computational Linguistics.

Tom Kwiatkowksi, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1223–1233, Cambridge, MA, October. Association for Computational Linguistics.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1512–1523, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556, Seattle, Washington, USA, October. Association for Computational Linguistics.

Junhui Li, Muhua Zhu, Wei Lu, and Guodong Zhou. 2015. Improving semantic parsing with enriched synchronous context-free grammar. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1465, Lisbon, Portugal, September. Association for Computational Linguistics.

Percy Liang, Michael Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 590–599, Portland, Oregon, USA, June. Association for Computational Linguistics.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv preprint arXiv:1309.4408*.

Wei Lu, Hwee Tou Ng, Wee Sun Lee, and Luke S. Zettlemoyer. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 783–792, Honolulu, Hawaii, October. Association for Computational Linguistics.

Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 756–766, Denver, Colorado, May–June. Association for Computational Linguistics.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1–10, Singapore, August. Association for Computational Linguistics.

Hoifung Poon. 2013. Grounded unsupervised semantic parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 933–943, Sofia, Bulgaria, August. Association for Computational Linguistics.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4:127–140.

Adrienne Wang, Tom Kwiatkowski, and Luke Zettlemoyer. 2014. Morpho-syntactic lexical generalization for ccg semantic parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1284–1295, Doha, Qatar, October. Association for Computational Linguistics.

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

Yuk Wah Wong and Raymond Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 960–967, Prague, Czech Republic, June. Association for Computational Linguistics.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 645–650, Doha, Qatar, October. Association for Computational Linguistics.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 956–966, Baltimore, Maryland, June. Association for Computational Linguistics.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 66–70, Denver, Colorado, June. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July. Association for Computational Linguistics.

John M. Zelle and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1050–1055, Portland, OR, August. AAAI Press/MIT Press.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, Edinburgh, Scotland, July 26-29, 2005*, pages 658–666.

Luke Zettlemoyer and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic, June. Association for Computational Linguistics.

Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint model for question answering over multiple knowledge bases. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 3094–3100.

Chinese Zero Pronoun Resolution with Deep Neural Networks

Chen Chen and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{yzcchen,vince}@hlt.utdallas.edu

Abstract

While unsupervised anaphoric zero pronoun (AZP) resolvers have recently been shown to rival their supervised counterparts in performance, it is relatively difficult to scale them up to reach the next level of performance due to the large amount of feature engineering efforts involved and their ineffectiveness in exploiting lexical features. To address these weaknesses, we propose a supervised approach to AZP resolution based on deep neural networks, taking advantage of their ability to learn useful task-specific representations and effectively exploit lexical features via word embeddings. Our approach achieves state-of-the-art performance when resolving the Chinese AZPs in the OntoNotes corpus.

1 Introduction

A zero pronoun (ZP) is a gap in a sentence that is found when a phonetically null form is used to refer to a real-world entity. An anaphoric zero pronoun (AZP) is a ZP that corefers with one or more preceding mentions in the associated text. Below is an example taken from the Chinese Treebank (CTB), where the ZP (denoted as *pro*) refers to 俄罗斯 (Russia).

[俄罗斯] 作为米洛舍夫维奇一贯的支持者，*pro* 曾经提出调停这场政治危机。

([Russia] is a consistent supporter of Milošević, *pro* has proposed to mediate the political crisis.)

As we can see, ZPs lack grammatical attributes that are useful for overt pronoun resolution such as number and gender. This makes ZP resolution more challenging than overt pronoun resolution.

Automatic ZP resolution is typically composed of two steps. The first step, AZP identification, involves extracting ZPs that are anaphoric. The second step, AZP resolution, aims to identify an antecedent of an AZP. State-of-the-art ZP resolvers have tackled both of these steps in a supervised manner, training one classifier for AZP identification and another for AZP resolution (e.g., Zhao and Ng (2007), Kong and Zhou (2010)).

More recently, Chen and Ng (2014b; 2015) have proposed unsupervised probabilistic AZP resolution models (henceforth the CN14 model and the CN15 model, respectively) that rival their supervised counterparts in performance. An appealing aspect of these unsupervised models is that their language-independent generative process enables them to be applied to languages where data annotated with ZP links are not readily available. Though achieving state-of-the-art performance, these models have several weaknesses.

First, a lot of manual efforts need to be spent on engineering the features for generative probabilistic models, as these models are sensitive to the choice of features. For instance, having features that are (partially) dependent on each other could harm model performance. Second, in the absence of labeled data, it is difficult, though not impossible, for these models to profitably employ lexical features (e.g., word pairs, syntactic patterns involving words), as determining which lexical features are useful and how to combine the potentially large number of lexical features in an unsupervised manner is a very challenging task. In fact, the unsupervised models proposed by Chen and Ng (2014b; 2015) are unlexicalized, presumably owing to the aforementioned reasons. Unfortunately, as shown in previous work (e.g, Zhao and Ng (2007), Chen and Ng (2013)), the use of lexical features contributed significantly to the performance of state-of-the-art supervised AZP resolvers. Finally, owing to the lack of labeled data, the model parameters are learned to maximize data

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 778–788,
Berlin, Germany, August 7-12, 2016.

likelihood, which may not correlate well with the desired evaluation measure (i.e., F-score). Hence, while unsupervised resolvers have achieved state-of-the-art performance, these weaknesses together suggest that it is very challenging to scale these models up so that they can achieve the next level of performance.

Our goal in this paper is to improve the state of the art in AZP resolution. Motivated by the aforementioned weaknesses, we propose a novel approach to AZP resolution using deep neural networks, which we believe has three key advantages over competing unsupervised counterparts.

First, deep neural networks are particularly good at discovering hidden structures from the input data and learning task-specific representations via successive transformations of the input vectors, where different layers of a network correspond to different levels of abstractions that are useful for the target task. For the task of AZP resolution, this is desirable. Traditionally, it is difficult to correctly resolve an AZP if its context is lexically different from its antecedent's context. This is especially the case for unsupervised resolvers. In contrast, a deep network can handle difficult cases like this via learning representations that make lexically different contexts look similar.

Second, we train our deep network in a supervised manner.[1] In particular, motivated by recent successes of applying the mention-ranking model (Denis and Baldridge, 2008) to entity coreference resolution (e.g., Chang et al. (2013), Durrett and Klein (2013), Clark and Manning (2015), Martschat and Strube (2015), Wiseman et al. (2015)), we propose to employ a ranking-based deep network, which is trained to assign the highest probability to the correct antecedent of an AZP given a set of candidate antecedents. This contrasts with existing supervised AZP resolvers, all of which are classification-based. Optimizing this objective function is better than maximizing data likelihood, as the former is more tightly coupled with the desired evaluation metric (F-score) than the latter.

Finally, given that our network is trained in a supervised manner, we can extensively employ lexical features and use them in combination with other types of features that have been shown to be useful for AZP resolution. However, rather than employing words directly as features, we employ word embeddings trained in an unsupervised manner. The goal of the deep network will then be to take these task-independent word embeddings as input and convert them into embeddings that would work best for AZP resolution via supervised learning. We call our approach an *embedding matching* approach because the underlying deep network attempts to compare the embedding learned for an AZP with the embedding learned for each of its antecedents.

To our knowledge, this is the first approach to AZP resolution based on deep networks. When evaluated on the Chinese portion of the OntoNotes 5.0 corpus, our embedding matching approach to AZP resolution outperforms the CN15 model, achieving state-of-the-art results.

The rest of the paper is organized as follows. Section 2 overviews related work on zero pronoun resolution for Chinese and other languages. Section 3 describes our embedding matching approach, specifically the network architecture and the way we train and apply the network. We present our evaluation results in Section 4 and our conclusions in Section 5.

2 Related Work

Chinese ZP resolution. Early approaches to Chinese ZP resolution are *rule-based*. Converse (2006) applied Hobbs' algorithm (Hobbs, 1978) to resolve the ZPs in the CTB documents. Yeh and Chen (2007) hand-engineered a set of rules for ZP resolution based on Centering Theory (Grosz et al., 1995).

In contrast, virtually all recent approaches to this task are *learning-based*. Zhao and Ng (2007) are the first to employ a supervised learning approach to Chinese ZP resolution. They trained an AZP resolver by employing syntactic and positional features in combination with a decision tree learner. Unlike Zhao and Ng, Kong and Zhou (2010) employed context-sensitive convolution tree kernels (Zhou et al., 2008) in their resolver to model syntactic information. Chen and Ng (2013) extended Zhao and Ng's feature set with novel features that encode the context surrounding a ZP and its candidate antecedents, and exploited the coreference links between ZPs as bridges to

[1] Note that deep neural networks do *not* necessarily have to be trained in a supervised manner. In fact, in early research on extending semantic modeling using auto-encoders (Salakhutdinov and Hinton, 2007), the networks were trained in an unsupervised manner, where the model parameters were optimized for the reconstruction of the input vectors.

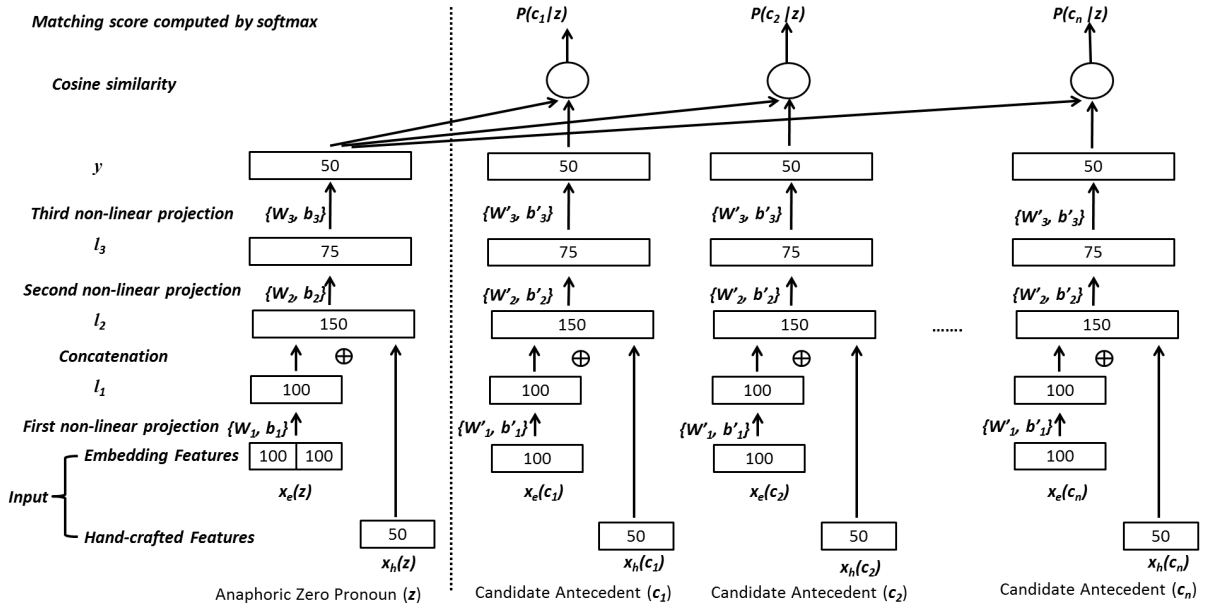


Figure 1: The architecture of our embedding matching model. The number in each box indicates the size of the corresponding vector.

find textually distant antecedents for ZPs. As mentioned above, there have been attempts to perform unsupervised AZP resolution. For instance, using only data containing manually resolved *overt* pronouns, Chen and Ng (2014a) trained a supervised overt pronoun resolver and applied it to resolve AZPs. More recently, Chen and Ng (2014b; 2015) have proposed unsupervised probabilistic AZP resolution models that rivaled their supervised counterparts in performance. While we aim to resolve *anaphoric* ZPs, Rao et al. (2015) resolved *deictic non-anaphoric* ZPs, which "refer to salient entities in the environment such as the speaker, hearer or pragmatically accessible referent without requiring any introduction in the preceding text".

ZP resolution for other languages. There have been rule-based and supervised machine learning approaches for resolving ZPs in other languages. For example, to resolve ZPs in Spanish texts, Ferrández and Peral (2000) proposed a set of hand-crafted rules that encode preferences for candidate antecedents. In addition, supervised approaches have been extensively employed to resolve ZPs in Korean (e.g., Han (2006)), Japanese (e.g., Seki et al. (2002), Isozaki and Hirao (2003), Iida et al. (2006; 2007), Sasano et al. (2008), Taira et al. (2008), Imamura et al. (2009), Sasano et al. (2009), Watanabe et al. (2010), Hayashibe et al. (2011), Iida and Poesio (2011), Sasano and Kurohashi (2011), Yoshikawa et al. (2011), Hangyo et al. (2013), Yoshino et al. (2013), Iida et al. (2015)), and Italian (e.g., Iida and Poesio (2011)).

3 Model

In this section, we first introduce our network architecture (Section 3.1), and then describe how we train it (Section 3.2) and apply it (Section 3.3).

3.1 Network Architecture

The network architecture is shown in Figure 1. Since we employ a ranking model to rank the candidate antecedents of an AZP z, the inputs to the network are (1) a feature vector representing the AZP, and (2) n feature vectors representing its n candidate antecedents, c_1, c_2, ..., c_n. As will be explained in detail in Section 3.2.2, the features in each feature vector can be divided into two types: word embedding features and hand-crafted features. Each input feature vector will then be passed through three hidden layers in the network, which will successively map it into a low-dimensional feature space. The resulting vector can be viewed as the low-dimensional semantic embedding of the corresponding input vector. Finally, the model computes a *matching* score between z and each of its candidate antecedents based on their low-dimensional representations. These scores are then normalized into probabilities using a softmax.

More formally, let $x_e(z)$ and $x_h(z)$ be the vec-

tors of embedding and hand-crafted features representing AZP z respectively, and let $x_e(c_i)$ and $x_h(c_i)$ be the vectors of embedding and hand-crafted features representing candidate antecedent c_i respectively. In addition, let $y(z)$ and $y(c_i)$ be the (low-dimensional) output vectors for z and c_i respectively, l_1, l_2, and l_3 be the intermediate hidden layers, W_i and W_i' be the weight matrices associated with z and the c_i's in hidden layer i, b_i and b_i' be the bias terms associated with z and the c_i's.[2] We then have:

$$\begin{aligned} l_1(z) &= f(W_1 x_e(z) + b_1) \\ l_2(z) &= l_1(z) \oplus x_h(z) \\ l_3(z) &= f(W_2 l_2(z) + b_2) \\ y(z) &= f(W_3 l_3(z) + b_3) \end{aligned} \quad (1)$$

$$\begin{aligned} l_1(c_i) &= f(W_1' x_e(c_i) + b_1') \\ l_2(c_i) &= l_1(c_i) \oplus x_h(c_i) \\ l_3(c_i) &= f(W_2' l_2(c_i) + b_2') \\ y(c_i) &= f(W_3' l_3(z) + b_3') \end{aligned} \quad (2)$$

where f is the activation function at output layer y and hidden layers l_1 and l_3. In this network, we employ tanh as the activation function. Hence,

$$f(x) = \tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (3)$$

The matching score between an AZP z and a candidate antecedent c_i is then measured as:

$$R(z, c_i) = \cos(y(z), y(c_i)) = \frac{y(z)^T y(c_i)}{||y(z)||||y(c_i)||} \quad (4)$$

3.2 Training

3.2.1 Training Instance Creation

We create one training instance from each AZP in each training document. Since our model is ranking-based, each training instance corresponds to an AZP z and all of its candidate antecedents C_i. In principle, we can follow previous work and assume that the set of candidate antecedents C contains all and only those maximal or modifier noun phrases (NPs) that precede z in the associated text and are at most two sentences away from it. However, to improve *training* efficiency, we select exactly four candidate antecedents for each AZP z as follows. First, we take the closest correct antecedent z to be one of the four candidate antecedents. Next, we compute a salience score for each of its *non-coreferent* candidate antecedents and select the three with the highest salience scores as the remaining three candidate antecedents.

We compute salience as follows. For each AZP z, we compute the salience score for each (partial) entity preceding z.[3] To reduce the size of the list of preceding entities, we only consider a partial entity *active* if at least one of its mentions appears within two sentences of the active AZP z. We compute the salience score of each active entity w.r.t. z using the following equation:

$$\sum_{m \in E} g(m) * decay(m) \quad (5)$$

where m is a mention belonging to active entity E, $g(m)$ is a grammatical score which is set to 4, 2, or 1 depending on whether m's grammatical role is SUBJECT, OBJECT, or OTHER respectively, and $decay(m)$ is a decay factor that is set to 0.5^{dis} (where dis is the sentence distance between m and z).

Finally, we assign the correct label (i.e., the *matching score*) to each candidate antecedent. The score is 1 for the correct antecedent and 0 otherwise.

3.2.2 Features

As we can see from Figure 1, each input feature vector, regardless of whether it is representing an AZP or one of its candidate antecedents, is composed of two types of features, embedding features and hand-crafted features, as described below.

Embedding features. To encode the lexical contexts of the AZP and its candidate antecedents, one could employ one-hot vectors. However, the resulting lexical features may suffer from sparsity. To see the reason, assuming that the vocabulary size is V and the number of neurons in the first hidden layer l_1 is L_1, the size of the weight matrices W_1 and W_1' is $V * L_1$, which in our dataset is around two million while the number of training examples is much smaller.

Therefore, instead of using one-hot vectors, we employ embedding features. Specifically, we employ the pre-trained word embeddings (of size 100)

[2] Note that the target AZP and its candidate antecedents use different weight matrices and biases within each layer. This is needed because the features of the AZP and those of the candidate antecedents come from two different feature spaces.

[3] We compute the list of preceding entities automatically using SinoCoreferencer (Chen and Ng, 2014c), a Chinese entity coreference resolver downloadable from `http://www.hlt.utdallas.edu/~yzcchen/coreference/`.

Syntactic features (13)	whether z is the first gap in an IP clause; whether z is the first gap in a subject-less IP clause, and if so, POS(w_1); whether POS(w_1) is NT; whether w_1 is a verb that appears in a NP or VP; whether P_l is a NP node; whether P_r is a VP node; the phrasal label of the parent of the node containing POS(w_1); whether V has a NP, VP or CP ancestor; whether C is a VP node; whether there is a VP node whose parent is an IP node in the path from w_1 to C.
Other features (6)	whether z is the first gap in a sentence; whether z is in the headline of the text; the type of the clause in which z appears; the grammatical role of z (SUBJECT, OBJECT, or OTHER); whether w_{-1} is a punctuation; whether w_{-1} is a comma.

Table 1: Hand-crafted features associated with an AZP. z is a zero pronoun. V is the VP node following z. w_i is the ith word to the right of z (if i is positive) or the ith word to the left of z (if i is negative). C is lowest common ancestor of w_{-1} and w_1. P_l and P_r are the child nodes of C that are the ancestors of w_{-1} and w_1 respectively.

Syntactic features (12)	whether c has an ancestor NP, and if so, whether this NP is a descendent of c's lowest ancestor IP; whether c has an ancestor VP, and if so, whether this VP is a descendent of c's lowest ancestor IP; whether c has an ancestor CP; the grammatical role of c (SUBJECT, OBJECT, or OTHER); the clause type in which c appears; whether c is an adverbial NP, a temporal NP, a pronoun or a named entity.
Distance features (4)	the sentence distance between c and z; the segment distance between c and z, where segments are separated by punctuations; whether c is the closest NP to z; whether c and z are siblings in the associated parse tree.
Other features (2)	whether c is in the headline of the text; whether c is a subject whose governing verb is lexically identical to the verb governing of z.

Table 2: Hand-crafted features associated with a candidate antecedent. z is a zero pronoun. c is a candidate antecedent of z. V is the VP node following z in the parse tree.

obtained by training *word2vec*[4] on the Chinese portion of the training data from the OntoNotes 5.0 corpus. For an AZP z, we first find the word preceding it and its governing verb, and then concatenate the embeddings of these two words to form the AZP's embedding features. (If z happens to begin a sentence, we use a special embedding to represent the word preceding it.) For a candidate antecedent, we employ the word embedding of its head word as its embedding features.

Hand-crafted features. The hand-crafted features are (low-dimensional) features that capture the syntactic, positional and other relationships between an AZP and its candidate antecedents. These features are similar to the ones employed in previous work on AZP resolution (e.g., Zhao and Ng (2007), Kong and Zhou (2010), Chen and Ng (2013)).

We split these hand-crafted features into two disjoint sets: those associated with an AZP and those associated with a candidate antecedent. If a feature is computed based on the AZP, then we regard it as a feature associated with the AZP; otherwise, we put it in the other feature set. A brief description of the hand-crafted features associated with an AZP and those associated with a candidate antecedent are shown in Table 1 and Table 2 respectively. Note that we convert each multi-valued feature into a corresponding set of binary-valued features (i.e., if a feature has N different values, we will create N binary indicators to represent it). To ensure that the number of hand-crafted features representing an AZP is equal to the number of hand-crafted features representing a candidate antecedent[5], we append to the end of a feature vector as many dummy zeroes as needed.[6]

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

3.2.3 Parameter Estimation

We employ online learning to train the network, with one training example in a mini-batch. In other words, we update the weights after processing each training example based on the *correct* matching scores of the training example (which is 1 for the correct antecedent and 0 otherwise) and the network's *predicted* matching scores.

To compute the predicted matching score between AZP z and one of its candidate antecedents c_i, we apply the following softmax function:

$$P(c_i|z,\Lambda) = \frac{exp(\gamma R(z,c_i))}{\sum_{c' \in \mathcal{C}} exp(\gamma R(z,c'))} \quad (6)$$

where (1) γ is a smoothing factor that is empirically set on a held-out data set, (2) $R(z,c_i)$ is the cosine similarity between vector $y(z)$ and vector $y(c_i)$ (see Section 3.1), (3) $\mathcal{C}$ denotes the set of candidate antecedents of z, and (4) Λ denotes the set of parameters of our neural network:

[5] As seen in Figure 1, we set the length of the vector to 50.

[6] Appending dummy 0s is solely for the convenience of the network implementation: doing so does not have any effect on any computation.

$$\Lambda = \{W_1, W_2, W_3, b_1, b_2, b_3, W_1', W_2', W_3', b_1', b_2', b_3'\} \quad (7)$$

To maximize the matching score of the correct antecedent, we estimate the model parameters to minimize the following loss function:

$$\mathcal{J}_z(\Lambda) = -\sum_{c_i \in \mathcal{C}} \delta(z, c_i) P(c_i|z, \Lambda) \quad (8)$$

where $\delta(z, c_i)$ is an indicator function indicating whether AZP z and candidate antecedent c_i are coreferent:

$$\delta(z, c_i) = \begin{cases} 1, & \text{if } z \text{ and } c_i \text{ are coreferent} \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

Since $\mathcal{J}_z(\Lambda)$ is differentiable w.r.t. to Λ, we train the model using stochastic gradient descent. Specifically, the model parameters Λ are updated according to the following update rule:

$$\Lambda_t = \Lambda_{t-1} - \alpha \frac{\partial \mathcal{J}(\Lambda_{t-1})}{\partial \Lambda_{t-1}} \quad (10)$$

where α is the learning rate, and Λ_t and Λ_{t-1} are model parameters at the tth iteration and the $(t-1)$th iteration respectively. To avoid overfitting, we determine the hyperparameters of the network using a held-out development set.

3.3 Inference

After training, we can apply the resulting network to find an antecedent for each AZP. Each test instance corresponds to an AZP z and four of its candidate antecedents. Specifically, the four candidate antecedents with the highest salience scores will be chosen. Importantly, unlike in training, where we guarantee that the correct antecedents is among the set of candidate antecedents, in testing, we *don't*. We use the network to rank the candidate antecedents by computing the posterior probability of each of them being a correct antecedent of z, and select the one with the highest probability to be its antecedent.

The aforementioned resolution procedure can be improved, however. The improvement is motivated by a problem we observed previously (Chen and Ng, 2013): an AZP and its closest antecedent can sometimes be far away from each other, thus making it difficult to correctly resolve the AZP. To address this problem, we employ the following resolution procedure in our experiments. Given a test document, we process its AZPs in a left-to-right manner. As soon as we resolve an AZP to a preceding NP c, we fill the corresponding AZP's gap with c. Hence, when we process an AZP z, all of its preceding AZPs in the associated text have been resolved, with their gaps filled by the NPs they are resolved to. To resolve z, we create test instances between z and its four most salient candidate antecedents in the same way as described before. The only difference is that the set of candidate antecedents of z may now include those NPs that are used to fill the gaps of the AZPs resolved so far. Some of these additional candidate antecedents are closer to z than the original candidate antecedents, thereby facilitating the resolution of z. If the model resolves z to the additional candidate antecedent that fills the gap left behind by, say, AZP z', we postprocess the output by resolving z to the NP that z' is resolved to.[7]

	Training	Test
Documents	1,391	172
Sentences	36,487	6,083
Words	756,063	110,034
AZPs	12,111	1,713

Table 3: Statistics on the training and test sets.

4 Evaluation

4.1 Experimental Setup

Datasets. We employ the Chinese portion of the OntoNotes 5.0 corpus that was used in the official CoNLL-2012 shared task (Pradhan et al., 2012). In the CoNLL-2012 data, the training set and the development set contain ZP coreference annotations, but the test set does not. Therefore, we train our models on the training set and perform evaluation on the development set. Statistics on the datasets are shown in Table 3. The documents in these datasets come from six sources, namely Broadcast News (BN), Newswire (NW), Broadcast Conversation (BC), Telephone Conversation (TC), Web Blog (WB) and Magazine (MZ).

Evaluation measures. Following previous work on AZP resolution (e.g., Zhao and Ng (2007), Chen and Ng (2013)), we express the results of AZP resolution in terms of recall (R), precision (P) and F-score (F). We report the scores for each source in addition to the overall score.

[7] This postprocessing step is needed because the additional candidate antecedents are only gap fillers.

Number of embedding features for a word	100
Number of hand-crafted features	50
Number of neurons in l_1	100
Number of neurons in l_3	75
Number of neurons in y	50
Number of epochs over the training data	100
Smoothing factor γ	20
Learning rate α	0.01

Table 4: Hyperparameter values.

Hyperparameter tuning. We reserve 20% of the training set for tuning hyperparameters. The tuned hyperparameter values are shown in Table 4.

Evaluation settings. Following Chen and Ng (2013), we evaluate our model in three settings. In Setting 1, we assume the availability of gold syntactic parse trees and gold AZPs. In Setting 2, we employ gold syntactic parse trees and system (i.e., automatically identified) AZPs. Finally, in Setting 3, we employ system syntactic parse trees and system AZPs. The gold and system syntactic parse trees, as well as the gold AZPs, are obtained from the CoNLL-2012 shared task dataset, while the system AZPs are identified by a learning-based AZP identifier described in the Appendix.

Baseline system. As our baseline, we employ Chen and Ng's (2015) system, which has achieved the best result on our test set.

4.2 Results and Discussion

Results of the baseline system and our model on entire test set are shown in row 1 of Table 5. The three major columns in the table show the results obtained in the three settings. As we can see, our model outperforms the baseline significantly by 2.0%, 1.8%, and 1.1% in F-score under Settings 1, 2, and 3, respectively.[8]

Rows 2−7 of Table 5 show the resolution results on each of the six sources. As we can see, in Setting 1, our model beats the baseline on all six sources in F-score: by 2.4% (NW), 2.5% (MZ), 4.5% (WB), 1.6% (BN), 1.4% (BC), and 0.4% (TC). All the improvements are significant except for TC. These results suggest that our approach works well across different sources. In Setting 2, our model outperforms the baseline on all sources except NW and BC, where the F-scores drop insignificantly by 0.1% for both sources. Finally, in Setting 3, our model outperforms the baseline on all sources except NW and TC, where F-scores drop significantly by 0.7% for NW and 1.1% for TC.

Given the challenges in applying supervised learning (in particular, the difficulty and time involved in training the deep neural network as well as the time and effort involved in manually annotating the data needed to train the network), one may wonder whether the small though statistically significant improvements in these results provide sufficient justification for going back to supervised learning from the previous state-of-the-art unsupervised model. We believe that this is the beginning, not the end, of applying deep neural networks for AZP resolution. In particular, there is a lot of room for improvements, which may involve incorporating more sophisticated features and improving the design of the network (e.g., the dimensionality of the intermediate representations, the number of hidden layers, the objective function), for instance.

4.3 Ablation Results

Recall that the input of our model is composed of two groups of features, embedding features and hand-crafted features. To investigate the contribution of each of these two feature groups, we conduct ablation experiments. Specifically, in each ablation experiment, we retrain the network using only one group of features.

Ablation results under the three settings are shown in Table 6. In Setting 1, when the hand-crafted features are ablated, F-score drops significantly by 12.2%. We attribute the drop to the fact that the syntactic, positional, and other relationships encoded in the hand-crafted features play an important role in resolving AZPs. When the embedding features are ablated, F-score drops significantly by 3.7%. This result suggest the effectiveness of the embedding features.

Similar trends can be observed w.r.t. the other two settings: in Setting 2, F-score drops significantly by 6.8% and 2.2% when the hand-crafted features and the embedding features are ablated respectively, while in Setting 3, F-score drops significantly by 4.6% and 1.1% when the hand-crafted features and the embedding features are ablated.

4.4 Learning Curve

We show in Figure 2 the learning curve of the our model obtained under Setting 1. As we can see, after the first epoch, the F-score on the entire test set is around 46%, and it gradually increases to

[8] All significance tests are paired t-tests, with $p < 0.05$.

Source	Setting 1: Gold Parses, Gold AZPs						Setting 2: Gold Parses, System AZPs						Setting 3: System Parses, System AZPs					
	Baseline			Our Model			Baseline			Our Model			Baseline			Our Model		
	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F	R	P	F
Overall	50.0	50.4	50.2	51.8	52.5	52.2	35.7	26.2	30.3	39.6	27.0	32.1	19.6	15.5	17.3	21.9	15.8	18.4
NW	46.4	46.4	46.4	48.8	48.8	48.8	32.1	28.1	30.0	34.5	26.4	29.9	11.9	14.3	13.0	11.9	12.8	12.3
MZ	38.9	39.1	39.0	41.4	41.6	41.5	29.6	19.6	23.6	34.0	22.4	27.0	4.9	4.7	4.8	9.3	7.3	8.2
WB	51.8	51.8	51.8	56.3	56.3	56.3	39.1	22.9	28.9	44.7	25.1	32.2	20.1	14.3	16.7	23.9	16.1	19.2
BN	53.8	53.8	53.8	55.4	55.4	55.4	30.8	30.7	30.7	36.9	31.9	34.2	18.2	22.3	20.0	22.1	23.2	22.6
BC	49.2	49.6	49.4	50.4	51.3	50.8	35.9	26.6	30.6	37.6	25.6	30.5	19.4	14.6	16.7	21.2	14.6	17.3
TC	51.9	53.5	52.7	51.9	54.2	53.1	43.5	28.7	34.6	46.3	29.0	35.6	31.8	17.0	22.2	31.4	15.9	21.1

Table 5: AZP resolution results of the baseline and our model on the test set.

System	Setting 1: Gold Parses Gold AZPs			Setting 2: Gold Parses System AZPs			Setting 3: System Parses System AZPs		
	R	P	F	R	P	F	R	P	F
Full system	51.8	52.5	52.2	39.6	27.0	32.1	21.9	15.8	18.4
Embedding features only	39.2	40.8	40.0	30.9	21.5	25.3	16.3	12.0	13.8
Hand-crafted features only	48.2	48.7	48.5	37.0	25.1	29.9	20.6	14.9	17.3

Table 6: Ablation results of AZP resolution on the whole test set.

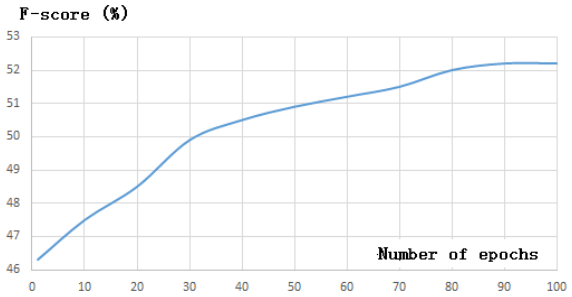


Figure 2: The learning curve of our model on the entire test set under Setting 1.

52% in the 80th epoch when performance starts to plateau. These results provide suggestive evidence for our earlier hypothesis that our objective function (Equation (8)) is tightly coupled with the desired evaluation metric (F-score).

4.5 Analysis of Results

To gain additional insights into our approach, we examine the outputs of our model obtained under Setting 1.

We first analyze the cases where the AZP was correctly resolved by our model but incorrectly resolved by the baseline. Consider the following representative example with the corresponding English translation.

[陈水扁] 在登机前发表简短谈话时表示，[台湾] 要站起来走出去。... ∗*pro*∗ 也希望此行能把国际友谊带回来。
[Chen Shui-bian] delivered a short speech before boarding, saying that [Taiwan] should stand up and go out. ... ∗*pro*∗ also hopes that this trip can bring back international friendship.

In this example, the correct antecedent of the AZP is 陈水扁 (Chen Shui-bian). However, the baseline incorrectly resolves it to 台湾 (Taiwan). The baseline's mistake can be attributed to the facts that (1) 台湾 is the most salient candidate antecedent in the discourse, and (2) 台湾 is closer to the AZP than the correct antecedent 陈水扁. Nevertheless, our model still correctly identifies 陈水扁 as the AZP's antecedent because of the embedding features. A closer inspection of the training data reveals that although the word 陈水扁 never appeared as the antecedent of an AZP whose governing verb is 希望 (hope) in the training data, many AZPs that are governed by 希望 are coreferent with other person names. Because the word 陈水扁 has a similar word embedding as those person names, our approach successfully generalizes such lexical context and makes the right resolution decision.

Next, we examine the errors made by our model and find that the majority of the mistakes result from insufficient lexical contexts. Currently, to encode the lexical contexts, we only consider the word preceding the AZP and its governing verb, as well as the head word of the candidate antecedent. However, this encoding ignored a lot of potentially useful context information, such as the clause following the AZP, the modifier of the candidate antecedent and the clause containing the candidate antecedent. Consider the following example:

[我] 前一会精神上太紧张。...∗pro∗ 现在比较平静了。
[I] was too nervous a while ago. ... ∗pro∗ am now calmer.

To resolve the AZP to its correct antecedent 我 (I), one needs to compare the two clauses containing the AZP and 我. However, since our model does not encode a candidate antecedent's context, it does not resolve the AZP correctly. One way to address this problem would be to employ *sentence embeddings* to represent the clauses containing the AZP and its candidate antecedents, and then perform sentence embedding matching to resolve the AZP. The primary challenge concerns how to train the model to match two clauses with probably no overlapping words and with a limited number of training examples.

5 Conclusions

We proposed an embedding matching approach to zero pronoun resolution based on deep networks. To our knowledge, this is the first neural network-based approach to zero pronoun resolution. When evaluated on the Chinese portion of the OntoNotes corpus, our approach achieved state-of-the-art results.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported in part by NSF Grants IIS-1219142 and IIS-1528037. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

References

Kai-Wei Chang, Rajhans Samdani, and Dan Roth. 2013. A constrained latent variable model for coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 601--612.

Chen Chen and Vincent Ng. 2013. Chinese zero pronoun resolution: Some recent advances. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1360--1365.

Chen Chen and Vincent Ng. 2014a. Chinese zero pronoun resolution: An unsupervised approach combining ranking and integer linear programming. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1622--1628.

Chen Chen and Vincent Ng. 2014b. Chinese zero pronoun resolution: An unsupervised probabilistic model rivaling supervised resolvers. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 763--774.

Chen Chen and Vincent Ng. 2014c. SinoCoreferencer: An end-to-end Chinese event coreference resolver. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation*, pages 4532--4538.

Chen Chen and Vincent Ng. 2015. Chinese zero pronoun resolution: A joint unsupervised discourse-aware model rivaling state-of-the-art resolvers. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 320--326.

Kevin Clark and Christopher D. Manning. 2015. Entity-centric coreference resolution with model stacking. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1405--1415.

Susan Converse. 2006. *Pronominal Anaphora Resolution in Chinese*. Ph.D. thesis, University of Pennsylvania.

Pascal Denis and Jason Baldridge. 2008. Specialized models and ranking for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 660--669.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1971--1982.

Antonio Ferrández and Jesús Peral. 2000. A computational approach to zero-pronouns in Spanish. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 166--172.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203--226.

Na-Rae Han. 2006. *Korean zero pronouns: Analysis and resolution*. Ph.D. thesis, University of Pennsylvania.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 924--934.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 201--209.

Jerry Hobbs. 1978. Resolving pronoun references. *Lingua*, 44:311--338.

Ryu Iida and Massimo Poesio. 2011. A cross-lingual ILP solution to zero anaphora resolution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, pages 804--813.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploting syntactic patterns as clues in zero-anaphora resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 625--632.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2007. Zero-anaphora resolution by learning rich syntactic pattern features. *ACM Transactions on Asian Language Information Processing*, 6(4).

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2179--2189.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85--88.

Hideki Isozaki and Tsutomu Hirao. 2003. Japanese zero pronoun resolution based on ranking rules and machine learning. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 184--191.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In Bernhard Scholkopf and Alexander Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 44--56. MIT Press.

Fang Kong and GuoDong Zhou. 2010. A tree kernel-based unified framework for Chinese zero anaphora resolution. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 882--891.

Sebastian Martschat and Michael Strube. 2015. Latent structures for coreference resolution. In *Transactions of the Association for Computational Linguistics*, 3:405--418.

Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Olga Uryupina, and Yuchen Zhang. 2012. CoNLL-2012 shared task: Modeling multilingual unrestricted coreference in OntoNotes. In *Proceedings of 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning: Shared Task*, pages 1--40.

Sudha Rao, Allyson Ettinger, Hal Daumé III, and Philip Resnik. 2015. Dialogue focus tracking for zero pronoun resolution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 494--503.

Ruslan Salakhutdinov and Geoffrey Hinton. 2007. Semantic hashing. In *Proceedings of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of the 5th International Joint Conference on Natural Language Processing*, pages 758--766.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2008. A fully-lexicalized probabilistic model for Japanese zero anaphora resolution. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 769--776.

Ryohei Sasano, Daisuke Kawahara, and Sadao Kurohashi. 2009. The effect of corpus size on case frame acquisition for discourse analysis. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 521--529.

Kazuhiro Seki, Atsushi Fujii, and Tetsuya Ishikawa. 2002. A probabilistic method for analyzing Japanese anaphora integrating zero pronoun detection and resolution. In *Proceedings of the 19th International Conference on Computational Linguistics - Volume 1*.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 523--532.

Yotaro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2010. A structured model for joint learning of argument roles and predicate senses. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 98--102.

Sam Wiseman, Alexander M. Rush, Stuart Shieber, and Jason Weston. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1416--1426.

Yaqin Yang and Nianwen Xue. 2010. Chasing the ghost: recovering empty categories in the Chinese Treebank. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1382--1390.

Ching-Long Yeh and Yi-Chun Chen. 2007. Zero anaphora resolution in Chinese with shallow parsing. *Journal of Chinese Language and Computing*, 17(1):41--56.

Katsumasa Yoshikawa, Masayuki Asahara, and Yuji Matsumoto. 2011. Jointly extracting Japanese predicate-argument relation with Markov Logic. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1125--1133.

Koichiro Yoshino, Shinsuke Mori, and Tatsuya Kawahara. 2013. Predicate argument structure analysis using partially annotated corpora. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 957--961.

Shanheng Zhao and Hwee Tou Ng. 2007. Identification and resolution of Chinese zero pronouns: A machine learning approach. In *Proceedings of the 2007 Joint Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning*, pages 541--550.

GuoDong Zhou, Fang Kong, and Qiaoming Zhu. 2008. Context-sensitive convolution tree kernel for pronoun resolution. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing*, pages 25--31.

Appendix: Anaphoric Zero Pronoun Identification

Recall that Settings 2 and 3 in our evaluation involve the use of system AZPs. Our supervised AZP identification procedure is composed of two steps. First, in the *extraction* step, we heuristically extract ZPs. Then, in the *classification* step, we train a classifier to determine which of the ZPs extracted in the first step are AZPs.

To implement the extraction step, we use Zhao and Ng's (2007) observation: ZPs can only occur before a VP node in a syntactic parse tree. However, according to Kong and Zhou (2010), ZPs do not need to be extracted from every VP: if a VP node occurs in a coordinate structure or is modified by an adverbial node, then only its parent VP node needs to be considered. We extract ZPs from all VPs that satisfy the above constraints.

To implement the classification step, we train a binary classifier using SVM^{light} (Joachims, 1999) on the CoNLL-2012 training set to distinguish AZPs from non-AZPs. Each instance corresponds to a ZP extracted in the first step and is represented by 32 features, 13 of which were proposed by Zhao and Ng (2007) and 19 of which were proposed by Yang and Xue (2010). A brief description of these features can be found in Table 7.

Syntactic features (13)	whether z is the first gap in an IP clause; whether z is the first gap in a subject-less IP clause, and if so, POS(w_1); whether POS(w_1) is NT; whether t_1 is a verb that appears in a NP or VP; whether P_l is a NP, QP, IP or ICP node; whether P_r is a VP node; the phrasal label of the parent of the node containing POS(t_1); whether V has a NP, VP, QP or CP ancestor; whether C is a VP node; whether the parent of V is an IP node; whether V's lowest IP ancestor has (1) a VP node as its parent and (2) a VV node as its left sibling; whether there is a VP node whose parent is an IP node in the path from t_1 to C.
Lexical features (13)	the words surrounding z and/or their POS tags, including w_1, w_{-1}, POS(w_1), POS(w_{-1}) + POS(w_1), POS(w_1) + POS(w_2), POS(w_{-2}) + POS(w_{-1}), POS(w_1) + POS(w_2) + POS(w_3), POS(w_{-1}) + w_1, and w_{-1} + POS(w_1); whether w_1 is a transitive verb, an intransitive verb or a preposition; whether w_{-1} is a transitive verb without an object.
Other features (6)	whether z is the first gap in a sentence; whether z is in the headline of the text; the type of the clause in which z appears; the grammatical role of z; whether w_{-1} is a punctuation; whether w_{-1} is a comma.

Table 7: Features for AZP identification. z is a zero pronoun. V is the VP node following z. w_i is the ith word to the right of z (if i is positive) or the ith word to the left of z (if i is negative). C is lowest common ancestor of w_{-1} and w_1. P_l and P_r are the child nodes of C that are the ancestors of w_{-1} and w_1 respectively.

When gold parse trees are employed, the recall, precision and F-score of the AZP identifier on our test set are 75.1%, 50.1% and 60.1% respectively. Using automatic parse trees, the performance of the AZP identifier drops to 43.7% (R), 30.7% (P) and 36.1% (F).

Constrained Multi-Task Learning for Automated Essay Scoring

Ronan Cummins
ALTA Institute
Computer Lab
University of Cambridge

Meng Zhang
ALTA Institute
Computer Lab
University of Cambridge

Ted Briscoe
ALTA Institute
Computer Lab
University of Cambridge

{rc635,mz342,ejb}@cl.cam.ac.uk

Abstract

Supervised machine learning models for automated essay scoring (AES) usually require substantial task-specific training data in order to make accurate predictions for a particular writing task. This limitation hinders their utility, and consequently their deployment in real-world settings. In this paper, we overcome this shortcoming using a constrained multi-task pairwise-preference learning approach that enables the data from multiple tasks to be combined effectively.

Furthermore, contrary to some recent research, we show that high performance AES systems can be built with little or no task-specific training data. We perform a detailed study of our approach on a publicly available dataset in scenarios where we have varying amounts of task-specific training data and in scenarios where the number of tasks increases.

1 Introduction

Automated essay scoring (AES) involves the prediction of a score (or scores) relating to the *quality* of an extended piece of written text (Page, 1966). With the burden involved in manually grading student texts and the increase in the number of ESL (English as a second language) learners worldwide, research into AES is increasingly seen as playing a viable role in assessment. Automating the assessment process is not only useful for educators but also for learners, as it can provide instant feedback and encourage iterative refinement of their writing.

The AES task has usually been addressed using machine learning. Given a set of texts and associated gold scores, machine learning approaches aim to build models that can generalise to unseen instances. Regression (Page, 1994; Persing and Ng, 2014; Phandi et al., 2015), classification (Larkey, 1998; Rudner and Liang, 2002), and preference-ranking[1] approaches (Yannakoudakis et al., 2011) have all been applied to the task. In general, machine learning models only perform well when the training and test instances are from similar distributions. However, it is usually the case that essays are written in response to prompts which are carefully designed to elicit answers according to a number of dimensions (e.g. register, topic, and genre). For example, Table 1 shows extracts from two prompts from a publicly available dataset[2] that aim to elicit different genres of persuasive/argumentative responses on different topics.

Most previous work on AES has either ignored the differences between essays written in response to different prompts (Yannakoudakis et al., 2011) with the aim of building general AES systems, or has built prompt-specific models for each prompt independently (Chen and He, 2013; Persing and Ng, 2014). One of the problems hindering the wide-scale adoption and deployment of AES systems is the dependence on prompt-specific training data, i.e. substantial model retraining is often needed when a new prompt is released. Therefore, systems that can adapt to new writing tasks (i.e. prompts) with relatively few new task-specific training examples are particularly appealing. For example, a system that is trained using only responses from prompt #1 in Table 1 may not generalise well to essays written in response to prompt #2, and vice versa. Even more complications arise when the scoring scale, marking criteria, and/or grade level (i.e. educational stage) vary from task

[1] also known as pairwise learning-to-rank

[2] available at https://www.kaggle.com/c/asap-aes

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 789–799,
Berlin, Germany, August 7-12, 2016.

#1	*Some experts are concerned that people are spending too much time on their computers and less time exercising, enjoying nature, and interacting with family and friends. Write a letter to your local newspaper in which you state your opinion on the effects computers have on people.*
#2	*Do you believe that certain materials, such as books, music, movies, magazines, etc., should be removed from the shelves if they are found offensive? Support your position with convincing arguments from your own experience, observations, and/or reading.*

Table 1: Two sample writing tasks from the ASAP (Automated Student Assessment Prize) dataset.

to task. If essays written in response to different tasks are marked on different scoring scales, then the actual scores assigned to essays across tasks are not directly comparable. This effect becomes even more pronounced when prompts are aimed at students in different educational stages.

In this paper, we address this problem of prompt adaptation using multi-task learning. In particular, we treat each prompt as a different task and introduce a constrained preference-ranking approach that can learn from multiple tasks even when the scoring scale and marking criteria are different across tasks. Our constrained preference-ranking approach significantly increases performance over a strong baseline system when there is limited prompt-specific training data available. Furthermore, we perform a detailed study using varying amounts of task-specific training data and varying numbers of tasks. First, we review some related work.

2 Related Work

A number of commercially available systems for AES, have been developed using machine learning techniques. These include PEG (Project Essay Grade) (Page, 2003), e-Rater (Attali and Burstein, 2006), and Intelligent Essay Assessor (IEA) (Landauer et al., 1998). Beyond commercial systems, there has been much research into varying aspects involved in automated assessment, including coherence (Higgins et al., 2004; Yannakoudakis and Briscoe, 2012), prompt-relevance (Persing and Ng, 2014; Higgins et al., 2006), argumentation (Labeke et al., 2013; Somasundaran et al., 2014; Persing and Ng, 2015), grammatical error detection and correction (Rozovskaya and Roth, 2011; Felice et al., 2014), and the development of publicly available resources (Yannakoudakis et al., 2011; Dahlmeier et al., 2013; Persing and Ng, 2014; Ng et al., 2014).

While most of the early commercially available systems use linear-regression models to map essay features to a score, a number of more sophisticated approaches have been developed. Preference-ranking (or pairwise learning-to-rank) has been shown to outperform regression for the AES problem (Yannakoudakis et al., 2011). However, they did not study prompt-specific models, as their models used training data originating from different prompts. We also adopt a preference-ranking approach but explicitly model prompt effects during learning. Algorithms that aim to directly maximise an evaluation metric have also been attempted. A listwise learning-to-rank approach (Chen and He, 2013) that directly optimises quadratic-weighted Kappa, a commonly used evaluation measure in AES, has also shown promising results.

Using training data from natural language tasks to boost performance of related tasks, for which there is limited training data, has received much attention of late (Collobert and Weston, 2008; Duh et al., 2010; Cheng et al., 2015). However, there have been relatively few attempts to apply transfer learning to automated assessment tasks. Notwithstanding, Napoles and Callison-Burch (2015) use a multi-task approach to model differences in assessors, while Heilman and Madnani (2013) specifically focus on domain-adaptation for short answer scoring over common scales. Most relevant is the work of Phandi et al. (2015), who applied domain-adaptation to the AES task using EasyAdapt (EA) (Daume III, 2007). They showed that supplementing a Bayesian linear ridge regression model (BLRR) with data from one other source domain is beneficial when there is limited target domain data. However, it was shown that simply using the source domain data as extra training data outperformed the EA domain adaptation approach in three out of four cases. One major limitation to their approach was that in many instances the source domain and target domain pairs were from different grade levels. This means that any attempt to resolve scores to a common scale is undermined by the fact that the gold scores are not comparable across domains, as the essays were written by students of different educational levels. A further limitation is that multi-domain adapta-

tion (whereby one has access to multiple source domains) was not considered.

The main difference between our work and previous work is that our model incorporates *multiple* source tasks and introduces a learning mechanism that enables us to combine these tasks even when the scores across tasks are not directly comparable. This has not been achieved before. This is non-trivial as it is difficult to see how this can be accomplished using a standard linear-regression approach. Furthermore, we perform the first comprehensive study of multi-task learning for AES using different training set sizes for a number of different learning scenarios.

3 Preference Ranking Model

In this section, we describe our baseline AES model which is somewhat similar to that developed by Yannakoudakis et al. (2011).

3.1 Perceptron Ranking (TAP$_{rank}$)

We use a preference-ranking model based on a binary margin-based linear classifier (the Timed Aggregate Perceptron or TAP) (Briscoe et al., 2010). In its simplest form this Perceptron uses batch learning to learn a decision boundary for classifying an input vector $\boldsymbol{x_i}$ as belonging to one of two categories. A timing-variable τ (set to 1.0 by default) controls both the learning rate and the number of epochs during training. A preference-ranking model is then built by learning to classify pairwise *difference vectors*, i.e. learning a weight vector $\boldsymbol{w}$ such that $\boldsymbol{w}(\boldsymbol{x_i} - \boldsymbol{x_j}) > \delta$, when essay i has a higher gold score than essay j, where δ is the one-sided margin[3] (Joachims, 2002; Chapelle and Keerthi, 2010). Therefore, instead of directly learning to predict the gold score of an essay vector, the model learns a weight vector $\boldsymbol{w}$ that minimizes the misclassification of *difference vectors*. Given that the number of pairwise difference vectors in a moderately sized dataset can be extremely large, the training set is reduced by randomly sampling difference vectors according to a user-defined probability (Briscoe et al., 2010). In all experiments in our paper we choose this probability such that $5n$ difference vectors are sampled, where n is the number of training instances (essays) used. We did not tune any of the hyper-parameters of the model.

[3]This margin is set to $\delta = 2.0$ by default.

3.2 From Rankings to Predicted Scores

As the weight vector $\boldsymbol{w}$ is optimized for pairwise ranking, a further step is needed to use the ranking model for predicting a score. In particular, for each of the n vectors in our training set, a real-scalar value is assigned according to the dot-product of the weight vector and the training instance (i.e. $\boldsymbol{w} \cdot \boldsymbol{x_i}$), essentially giving its distance (or margin) from the zero vector. Then using the training data, we train a one-dimensional linear regression model $\beta + \epsilon$ to map these assignments to the gold score of each instance.

Finally, to make a prediction $\hat{y}$ for a test vector, we first calculate its distance from the zero vector using $\boldsymbol{w} \cdot \boldsymbol{x_i}$ and map it to the scoring scale using the linear regression model $\hat{y} = \beta(\boldsymbol{w} \cdot \boldsymbol{x_i}) + \epsilon$. For brevity we denote this entire approach (a ranking and a linear regression step) to predicting the final score as **TAP**.

3.3 Features

The set of features used for our ranking model is similar to those identified in previous work (Yannakoudakis et al., 2011; Phandi et al., 2015) and is as follows:

1. word unigrams, bigrams, and trigrams
2. POS (part-of-speech) counts
3. essay length (as the number of unique words)
4. GRs (grammatical relations)
5. max-word length and min-sentence length
6. the presence of cohesive devices
7. an estimated error rate

Each essay is processed by the RASP system (Briscoe et al., 2006) with the standard tokenisation and sentence boundary detection modules. All n-grams are extracted from the tokenised sentences. The grammatical relations (GRs) are extracted from the top parse of each sentence in the essay. The presence of cohesive devices are used as features. In particular, we use four categories (i.e. addition, comparison, contrast and conclusion) which are hypothesised to measure the cohesion of a text.

The error rate is estimated based on a language model using ukWaC (Ferraresi et al., 2008) which contains more than 2 billion English tokens. A trigram in an essay will be treated as an error if it

Details						System Performance (QW-κ)		
Task	**# essays**	**Grade Level**	**Original Scale**	**Mean Score Resolved (0-60)**	**Human Agreement**	**BLRR Phandi**	**SVM Phandi**	**TAP**
1	1783	8	2-12	39	0.721	0.761	0.781	**0.815**
2	1800	10	1-6	29	0.814	0.606	0.621	**0.674**
3	1726	10	0-3	37	0.769	0.621	0.630	**0.642**
4	1772	10	0-3	29	0.851	0.742	0.749	**0.789**
5	1805	8	0-4	36	0.753	0.784	0.782	**0.801**
6	1800	10	0-4	41	0.776	0.775	0.771	**0.793**
7	1569	7	0-30	32	0.721	0.730	0.727	**0.772**
8	723	10	0-60	37	0.629	0.617	0.534	**0.688**

Table 2: Details of ASAP dataset and a preliminary evaluation of the performance of our **TAP** baseline against previous work (Phandi et al., 2015). All models used only task-specific data and 5-fold cross-validation. Best result is in **bold**.

is not found in the language model. Spelling errors are detected using a dictionary lookup, while a rule-based error module (Andersen et al., 2013) with rules generated from the Cambridge Learner Corpus (CLC) (Nicholls, 2003) is used to detect further errors. Finally, the unigrams, bigrams and trigrams are weighted by *tf-idf* (Sparck Jones, 1972), while all other features are weighted by their actual frequency in the essay.

4 Data and Preliminary Evaluation

In order to compare our baseline with previous work, we use the ASAP (Automated Student Assessment Prize) public dataset. Some details of the essays for the eight tasks in the dataset are described in the Table 2. The prompts elicit responses of different genres and of different lengths. In particular, it is important to note that the prompts have different scoring scales and are associated with different grade levels (7-10). Furthermore, the gold scores are distributed differently even if resolved to a common 0-60 scale. In order to benchmark our baseline system against previously developed approaches (BLRR and SVM regression (Phandi et al., 2015)) which use this data, we learned task-specific models using 5-fold cross-validation within each of the eight ASAP sets and aim to predict the unresolved *original* score as per previous work. We present the quadratic weighted kappa (QW-κ) of the systems in Table 2.[4] Our baseline preference-ranking model (TAP) outperforms previous approaches on task-specific data. It is worth noting that we did not tune either of the hyperparameters of TAP.

[4]The results for BLRR and SVM regression are taken directly from the original work and it is unlikely that we have used the exact same fold split. Regardless, the consistent increases mean that TAP represents a strong baseline system upon which we develop our constrained multi-task approach.

5 Multi-Task Learning

For multi-task learning we use EA encoding (Daume III, 2007) extended over k tasks $\mathcal{T}_{j=1..k}$ where each essay x_i is associated with one task $x_i \in \mathcal{T}_j$. The transfer-learning algorithm takes a set of input vectors associated with the essays, and for each vector $\boldsymbol{x}_i \in \mathbb{R}^F$ maps it via $\Phi(\boldsymbol{x}_i)$ to a higher dimensional space $\Phi(\boldsymbol{x}_i) \in \mathbb{R}^{(1+k)\cdot F}$. The encoding function $\Phi(\boldsymbol{x}_i)$ is as follows:

$$\Phi(\boldsymbol{x}) = \bigoplus_{j=0}^{k} f(\boldsymbol{x}, j) \tag{1}$$

where $\bigoplus$ is vector concatenation and $f(\boldsymbol{x}, j)$ is as follows:

$$f(\boldsymbol{x}, j) = \begin{cases} \boldsymbol{x}, & \text{if } j = 0 \\ \boldsymbol{x}, & \text{if } x \in \mathcal{T}_j \\ \mathbf{0}^F, & \text{otherwise} \end{cases} \tag{2}$$

Essentially, the encoding makes a task-specific copy of the original feature space of dimensionality F to ensure that there is one shared-representation and one task-specific representation for each input vector (with a zero vector for all other tasks). This approach can be seen as a re-encoding of the input vectors and can be used with any vector-based learning algorithm. Fig. 1 (left) shows an example of the extended feature vectors for three tasks $\mathcal{T}_j$ on different scoring scales. Using only the shared-representation (in blue) as input vectors to a learning algorithm results in a standard approach which does not learn task-specific characteristics. However, using the full representation allows the learning algorithm to capture both general and task-specific characteristics jointly. This simple encoding technique is easy to implement and has been shown to be useful for a number of NLP tasks (Daume III, 2007).

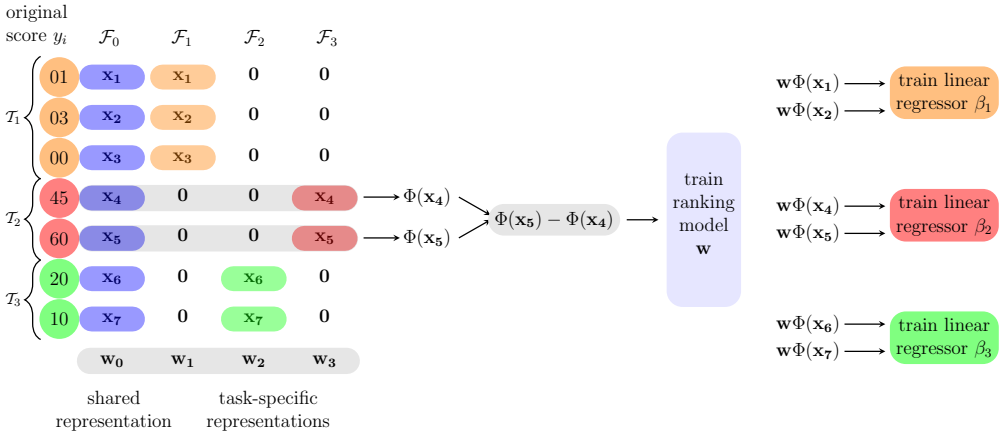


Figure 1: Example of the constrained multi-task learning approach for three tasks where the shared representation is in blue and the task-specific representations are in orange, red, and green. The original gold scores for each task $\mathcal{T}_j$ are on different scoring scales. The preference-ranking weight vector $\boldsymbol{w}$ to be learned is shown at the bottom. A one-dimensional linear regression model is learned for each task.

5.1 Constrained Preference-Ranking

Given essays from multiple tasks, it is often the case that the gold scores have different distributions, are not on the same scale, and have been marked using different criteria. Therefore, we introduce a modification to TAP (called cTAP$_{rank}$) that constrains the creation of pairwise difference vectors when training the weight vector $\boldsymbol{w}$. In particular, during training we ensure that pairwise difference vectors are not created from pairs of essays originating from different tasks.[5] We ensure that the same number of difference vectors are sampled during training for both TAP$_{rank}$ and our constrained version (i.e. both models use the same number of training instances). Figure 1 shows an example of the creation of a valid pairwise-difference vector in the multi-task framework.

Furthermore, for cTAP$_{rank}$ we train a final linear regression step on each of the task-specific training data separately. Therefore, we predict a score y for essay $\boldsymbol{x_i}$ for task $\mathcal{T}_j$ as $\hat{y} = \beta_j(\boldsymbol{w} \cdot \boldsymbol{x_i}) + \epsilon_j$. This is because for cTAP$_{rank}$ we assume that scores across tasks are not necessarily comparable. Therefore, although we utilise information originating from different tasks, the approach *never* mixes or directly compares instances originating from different tasks. This approach to predicting the final score is denoted **cTAP**.

[5]The same effect can be achieved in SVM^{rank} by encoding the prompt/task using the query id (*qid*). This constraint is analogous to the way SVM^{rank} is used in information retrieval where document relevance scores returned from different queries are not comparable.

6 Experimental Set-up

In this section, we outline the different learning scenarios, data folds, and evaluation metrics used in our main experiments.

6.1 Learning Approaches

We use the same features outlined in Section 3.3 to encode feature vectors for our learning approaches. In particular we study three learning approaches denoted and summarised as follows:

TAP: which uses the TAP$_{rank}$ algorithm with input vectors $\boldsymbol{x_i}$ of dimensionality F.

MTL-TAP: which uses the TAP$_{rank}$ algorithm with MTL extended input vectors $\Phi(\boldsymbol{x}_i)$.

MTL-cTAP: which uses the cTAP$_{rank}$ algorithm with MTL extended input vectors $\Phi(\boldsymbol{x}_i)$.[6]

For TAP and MTL-TAP, we attempt to resolve the essay score to a common scale (0-60) and subsequently train and test using this resolved scale. We then convert the score back to the original prompt-specific scale for evaluation. This is the approach used by the work most similar to ours (Phandi et al., 2015). It is worth noting that the resolution of scores to a common scale prior to training is *necessary* for both TAP and MTL-TAP when using data from multiple ASAP prompts. However, this step is *not* required for MTL-cTAP as this algorithm learns a ranking function $\boldsymbol{w}$ without directly comparing essays from different sets during training. Furthermore, the final regres-

[6]In the standard learning scenario when only target task data is available, MTL-TAP and MTL-cTAP are identical.

System	Target Task/Prompts 1	2	3	4	5	6	7	8
Tgt-TAP	0.830	0.728	0.717	0.842	0.851	0.811	0.790	0.730
Src-TAP	0.779	0.663	0.703	0.735	0.789	0.688	0.616	**0.625**
Src-MTL-TAP	0.824‡	0.683†	0.728‡	0.771‡	**0.829**‡	0.699	0.737‡	0.575
Src-MTL-cTAP	**0.826**‡	**0.698**$^{\star}_{\star}$‡	**0.729**‡	**0.773**$_{\star}$‡	0.827‡	**0.702**†$_{\star}$	**0.744**$^{\star}_{\star}$‡	0.589$^{\star}_{\star}$
All-TAP	0.806	0.652	0.702	0.805	0.814	0.802	0.728	0.629
All-MTL-TAP	0.831‡	0.722‡	0.728‡	0.823‡	0.849‡	0.808	0.783‡	0.680‡
All-MTL-cTAP	**0.832**‡	**0.731**‡$_{\star}$	**0.729**‡$_{\star}$	**0.840**$^{\star}_{\star}$‡	**0.852**‡$_{\star}$	**0.810**†	**0.802**$^{\star}_{\star}$‡	**0.717**$^{\star}_{\star}$‡

Table 3: Average Spearman ρ of systems over two-folds on the ASAP dataset. The best approach per prompt is in **bold**. ‡ (†) means that ρ is statistically greater than **Src-TAP** (top half) and **All-TAP** (bottom half) using the Steiger test at the 0.05 level (‡ means significant for both folds, † means for one of the folds), while $^{\star}_{\star}$ means statistically greater than **All-MTL-TAP** on both folds ($_{\star}$ for one fold).

System	Target Tasks/Prompts 1	2	3	4	5	6	7	8
Tgt-TAP	0.813	0.667	0.626	0.779	0.789	0.763	0.758	0.665
All-TAP	0.803	0.598	0.583	0.648	0.747	0.741	0.674	0.462
All-MTL-TAP	**0.825**‡	0.658‡	0.643‡	0.702‡	0.784‡	0.759‡	0.778‡	0.692‡
All-MTL-cTAP	0.816‡	**0.667**‡$_{\star}$	**0.654**$^{\star}_{\star}$‡	**0.783**$^{\star}_{\star}$‡	**0.801**$^{\star}_{\star}$‡	**0.778**$^{\star}_{\star}$‡	**0.787**‡$_{\star}$	**0.692**‡

Table 4: Average QW-κ of systems over two-folds on the ASAP dataset. The best approach per prompt is in **bold**. ‡ (†) means that κ is statistically ($p < 0.05$) greater than **All-TAP** using an approximate randomisation test (Yeh, 2000) using 50,000 samples. $^{\star}_{\star}$ means statistically greater than **All-MTL-TAP** on both folds ($_{\star}$ for one fold).

sion step in cTAP only uses original target task scores and therefore predicts scores on the correct scoring scale for the task. We study the three different learning approaches, TAP, MTL-TAP, and MTL-cTAP, in the following scenarios:

All: where the approach uses data from both the target task and the available source tasks.

Tgt: where the approach uses data from the target task only.

Src: where the approach uses data from only the available source tasks.

6.2 Data Folds

For our main experiments we divide the essays associated with each of the eight tasks into two folds. For *all* subsequent experiments, we train using data in one fold (often associated with multiple tasks) and test on data in the remaining fold of the specific target task. We report results for each task separately. These splits allow us to perform studies of all three learning approaches (TAP, MTL-TAP, and MTL-cTAP) using varying amounts of source and target task training data.

6.3 Evaluation Metrics

We use both Spearman's ρ correlation and Quadratic-weighted κ (QW-κ) to evaluate the performance of all approaches. Spearman's ρ measures the quality of the ranking of predicted scores produced by the system (i.e. the output from the ranking-preference model). We calculate Spearman's ρ using the ordinal gold score and the real-valued prediction on the original prompt-specific scoring scale of each prompt. Statistical significant differences between two correlations sharing one dependent variable (i.e. the gold scores) can be determined using Steiger's (1980) test.

QW-κ measures the chance corrected agreement between the predicted scores and the gold scores. QW-κ can be viewed as a measure of accuracy as it is lower when the predicted scores are further away from the gold scores. This metric measures both the quality of the ranking of scores and the quality of the linear regression step of our approach. These metrics are complementary as they measure different aspects of performance. We calculate QW-κ using the ordinal gold score and the real-valued prediction rounded to the nearest score on the original prompt-specific scale (see Table 2).

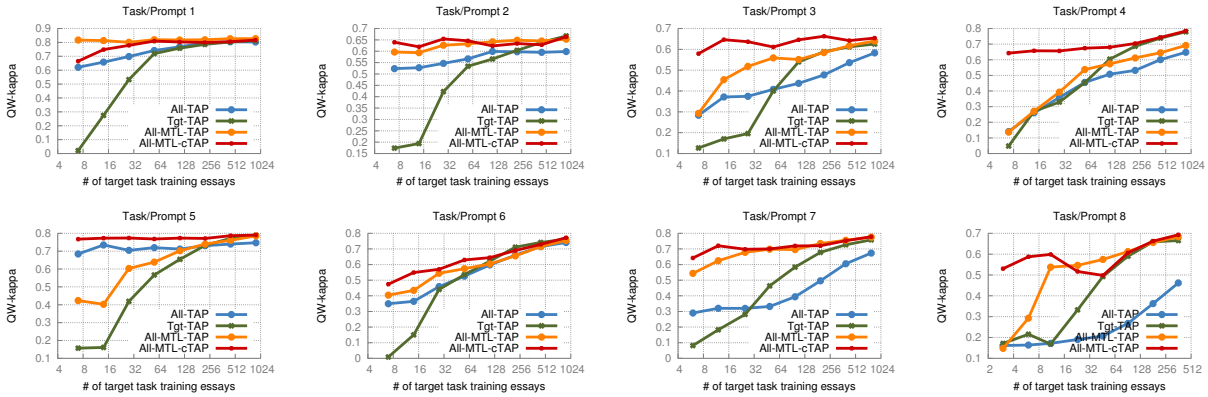


Figure 2: Average QW-κ over two folds for all tasks as size of target task training data increases

7 Results and Discussion

Table 3 and Table 4 show the performance of a number of models for both ρ and κ respectively. In general, we see that the **MTL** versions nearly always outperform the baseline TAP when using the same training data. This shows that multi-task learning is superior to simply using the source tasks as extra training data for the AES task. Interestingly this has not been shown before. Furthermore, the MTL-cTAP approach tends to be significantly better than the other for many prompts under varying scenarios for both Spearman's ρ and QW-κ. This shows that models that attempt to directly compare essays scores across certain writing-tasks lead to poorer performance.

When looking at Spearman's ρ in Table 3 we see that the models that do not use any target task data during training (Src) can achieve a performance which is close to the baseline that only uses all of the available target data (Tgt-TAP). This indicates that our system can *rank* essays well without any target task data. However, it is worth noting that *without* any target task training data and lacking any prior information as to the distribution of gold scores for the target task, achieving a consistently high accuracy (i.e. QW-κ) is extremely difficult (if not impossible). Therefore, Table 4 only shows results for models that make use of target task data.

For the models trained with data from all eight tasks, we can see that All-MTL-cTAP outperforms both All-TAP and All-MTL-TAP on most of the tasks for both evaluation metrics (ρ and κ). Interestingly, All-MTL-cTAP also outperforms Tgt-TAP on most of the prompts for both evaluation metrics. This indicates that All-MTL-cTAP manages to successfully incorporate useful information from the source tasks even when there is ample target-task data. We next look at scenarios when target-task training data is lacking.

7.1 Study of Target-Task Training Size

In real-world scenarios, it is often the case that we lack training data for a new writing task. We now report the results of an experiment that uses varying amounts of target-task training data. In particular, we use all source tasks and initially a small sample of task-specific data for each task (every 128^{th} target essay) and measure the performance of Tgt-TAP and the **All-*** models. We then double the amount of target-task training data used (by using every 64^{th} essay) and again measure performance, repeating this process until all target-task data is used. Figure 2 shows the performance of Tgt-TAP and the **All-*** models as target-task data increases.

In particular, Figure 2 shows that All-MTL-cTAP consistently outperforms all approaches in terms of agreement (QW-κ) and is particularly superior when there is very little target-task training data. It is worth remembering that All-MTL-cTAP only uses the target-task training instances for the final linear regression step. These results indicate that because the preference-ranking model performs so well, only a few target-task training instances are needed for the linear-regression step of All-MTL-cTAP. On the other hand, All-MTL-TAP uses all of the training instances in its final linear regression step, and performs significantly worse on a number of prompts. Again this shows the strengths of the constrained multi-task approach.

7.2 Study of Number of Source-tasks

All previous experiments that used source task data used the entire seven additional tasks. We

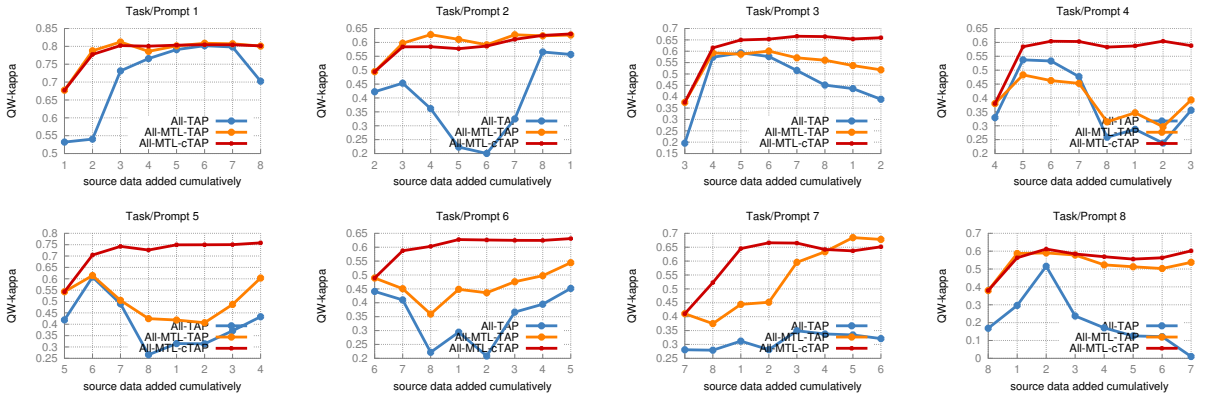


Figure 3: Average QW-κ over two folds as number of source tasks increases (using 25 target task instances)

now study the performance of the approaches as the number of source tasks changes. In particular, we limit the number of target task training instances to 25 and cumulatively add entire source task data in the order in which they occur in Table 2, starting with the source task appearing directly after the target task. We then measure performance at each stage. At the end of the process, each approach has access to all source tasks and the limited target task data.

Figure 3 shows the QW-κ for each prompt as the number of source tasks increases. We can see that All-TAP is the worst performing approach and often decreases as certain tasks are added as training data. All-MTL-cTAP is the best performing approach for nearly all prompts. Furthermore, All-MTL-cTAP is more robust than other approaches, as it rarely decreases in performance as the number of tasks increases.

8 Qualitative Analysis

As an indication of the type of interpretable information contained in the task-specific representations of the All-MTL-cTAP model, we examined the shared representation and two task-specific representations that relate to the example tasks outlined in Table 1. Table 5 shows the top weighted lexical features (i.e. unigrams, bigrams, or trigrams) (and their respective weights) in different parts of the All-MTL-cTAP model.

In general, we can see that the task-specific lexical components of the model capture topical aspects of the tasks and enable domain adaptation to occur. For example, we can see that *books*, *materials*, and *censorship* are highly discriminative lexical features for ranking essays written in response to task #2. The shared representation contains highly weighted lexical features across all tasks and captures vocabulary items useful for ranking in general. While this analysis gives us some insight into our model, it is more difficult to interpret the weights of other feature types (e.g. POS, GRs) across different parts of the model. We leave further analysis of our approach to future work.

9 Discussion and Conclusion

Unlike previous work (Phandi et al., 2015) we have shown, for the first time, that MTL outperforms an approach of simply using source task data as extra training data. This is because our approach uses information from multiple tasks without directly relying on the comparability of gold scores across tasks. Furthermore, it was concluded in previous work that at least some target-task training data is necessary to build high performing AES systems. However, as seen in Table 3, high performance rankers (ρ) can be built *without any* target-task data. Nevertheless, it is worth noting that without any target-data, accurately predicting the actual score (high κ) is extremely difficult. Therefore, although some extra information (i.e. the expected distribution of gold scores) would need to be used to produce accurate scores with a high quality ranker, the ranking is still useful for assessment in a number of scenarios (e.g. grading on a curve where the distribution of student scores is predefined).

The main approach adopted in this paper is quite similar to using SVM^{rank} (Joachims, 2002) while encoding the prompt id as the qid. When combined with a multi-task learning technique this allows the preference-ranking algorithm to learn

Shared		Task #1		Task #2	
2.024	offensive	1.146	this	2.027	offensive
1.852	hydrogen	0.985	less	1.229	books
1.641	hibiscus	0.980	computers	0.764	do_n't
1.602	shows	0.673	very	0.720	materials
1.357	strong	0.661	would	0.680	censorship
1.326	problem	0.647	could	0.679	person
1.288	grateful	0.624	,_and	0.676	read
1.286	dirigibles	0.599	family	0.666	children
1.234	books	0.599	less_time	0.661	offensive_.
1.216	her_new	0.579	spend	0.659	those
...	...	...	...	...	...
1.068	urban_areas	0.343	benefit_our_society	0.480	should_be_able
1.007	airships	0.341	believe_that_computers	0.475	able_to

Table 5: Highest weighted lexical features (i.e. unigrams, bigrams, or trigrams) and their weights in both shared and task-specific representations of the **All-MTL-cTAP** model (associated with results in Table 4) for the two example tasks referred to in Table 1.

both task-specific and shared-representations in a theoretically sound manner (i.e. without making any speculative assumptions about the relative orderings of essays that were graded on different scales using different marking criteria), and is general enough to be used in many situations.

Ultimately these complementary techniques (multi-task learning and constrained pairwise preference-ranking) allow essay scoring data from *any* source to be included during training. As shown in Section 7.2, our approach is robust to increases in the number of tasks, meaning that one can freely add extra data when available and expect the approach to use this data appropriately. This constrained multi-task preference-ranking approach is likely to be useful for many applications of multi-task learning, when the gold-scores across tasks are not directly comparable.

Future work will aim to study different dimensions of the prompt (e.g. genre, topic) using multi-task learning at a finer level. We also aim to further study the characteristics of the multi-task model in order to determine which features transfer well across tasks. Another avenue of potential research is to use multi-task learning to predict scores for different aspects of text quality (e.g. coherence, grammaticality, topicality).

Acknowledgements

We would like to thank Cambridge English Language Assessment for supporting this research, and the anonymous reviewers for their useful feedback. We would also like to thank Ekaterina Kochmar, Helen Yannakoudakis, Marek Rei, and Tamara Polajnar for feedback on early drafts of this paper.

References

Øistein E Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications, BEA*, pages 32–41.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with e-rater® v. 2. *The Journal of Technology, Learning and Assessment*, 4(3).

Ted Briscoe, John Carroll, and Rebecca Watson. 2006. The second release of the RASP system. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 77–80, Sydney, Australia, July. Association for Computational Linguistics.

Ted Briscoe, Ben Medlock, and Øistein Andersen. 2010. Automated assessment of esol free text examinations. Technical Report 790, The Computer Lab, University of Cambridge, February.

Olivier Chapelle and S Sathiya Keerthi. 2010. Efficient algorithms for ranking with svms. *Information Retrieval*, 13(3):201–215.

Hongbo Chen and Ben He. 2013. Automated essay scoring by maximizing human-machine agreement. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1752, Seattle, Washington, USA, October. Association for Computational Linguistics.

Hao Cheng, Hao Fang, and Mari Ostendorf. 2015. Open-domain name error detection using a multi-task RNN. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 737–746.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner english: The nus corpus of learner english. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31, Atlanta, Georgia, June. Association for Computational Linguistics.

Hal Daume III. 2007. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June. Association for Computational Linguistics.

Kevin Duh, Katsuhito Sudoh, Hajime Tsukada, Hideki Isozaki, and Masaaki Nagata, 2010. *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, chapter N-Best Reranking by Multitask Learning, pages 375–383. Association for Computational Linguistics.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, CoNLL 2014, Baltimore, Maryland, USA, June 26-27, 2014*, pages 15–24.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukwac, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

Michael Heilman and Nitin Madnani. 2013. Ets: domain adaptation and stacking for short answer scoring. In *Proceedings of the 2nd joint conference on lexical and computational semantics*, volume 2, pages 275–279.

Derrick Higgins, Jill Burstein, Daniel Marcu, and Claudia Gentile. 2004. Evaluating multiple aspects of coherence in student essays. In *HLT-NAACL*, pages 185–192.

D. Higgins, J. Burstein, and Y. Attali. 2006. Identifying off-topic student essays without topic-specific training data. *Natural Language Engineering*, 12(2):145–159.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

N Van Labeke, D Whitelock, D Field, S Pulman, and JTE Richardson. 2013. Openessayist: extractive summarisation and formative assessment of free-text essays. In *Proceedings of the 1st International Workshop on Discourse-Centric Learning Analytics*, Leuven, Belgium, April.

Thomas K Landauer, Peter W Foltz, and Darrell Laham. 1998. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284.

Leah S Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 90–95. ACM.

Courtney Napoles and Chris Callison-Burch. 2015. Automatically scoring freshman writing: A preliminary investigation. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 254–263, Denver, Colorado, June. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The conll-2014 shared task on grammatical error correction. In *In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2013 Shared Task)*. Association for Computational Linguistics.

Diane Nicholls. 2003. The cambridge learner corpus: Error coding and analysis for lexicography and elt. In *Proceedings of the Corpus Linguistics 2003 conference*, volume 16, pages 572–581.

Ellis B Page. 1966. The imminence of grading essays by computer. *Phi Delta Kappan*, 47:238–243.

Ellis Batten Page. 1994. Computer grading of student prose, using modern concepts and software. *The Journal of experimental education*, 62(2):127–142.

Ellis Batten Page. 2003. Project essay grade: Peg. *Automated essay scoring: A cross-disciplinary perspective*, pages 43–54.

Isaac Persing and Vincent Ng. 2014. Modeling prompt adherence in student essays. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1534–1543, Baltimore, Maryland, June. ACL.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552.

Peter Phandi, Kian Ming A. Chai, and Hwee Tou Ng. 2015. Flexible domain adaptation for automated essay scoring using correlated linear regression. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 431–439, Lisbon, Portugal, September. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for esl correction tasks. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 924–933, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lawrence M Rudner and Tahung Liang. 2002. Automated essay scoring using bayes' theorem. *The Journal of Technology, Learning and Assessment*, 1(2).

Swapna Somasundaran, Jill Burstein, and Martin Chodorow. 2014. Lexical chaining for measuring discourse coherence quality in test-taker essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 950–961, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21.

James H Steiger. 1980. Tests for comparing elements of a correlation matrix. *Psychological bulletin*, 87(2):245.

Helen Yannakoudakis and Ted Briscoe. 2012. Modeling coherence in esol learner texts. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 33–43, Montréal, Canada, June. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA, June. Association for Computational Linguistics.

Alexander Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics - Volume 2*, COLING '00, pages 947–953, Stroudsburg, PA, USA. Association for Computational Linguistics.

CFO: Conditional Focused Neural Question Answering with Large-scale Knowledge Bases

Zihang Dai* Carnegie Mellon University dzihang@andrew.cmu.edu

Lei Li* Toutiao.com lileicc@gmail.com

Wei Xu Baidu Research xuwei06@baidu.com

Abstract

How can we enable computers to automatically answer questions like "*Who created the character Harry Potter*"? Carefully built knowledge bases provide rich sources of facts. However, it remains a challenge to answer factoid questions raised in natural language due to numerous expressions of one question. In particular, we focus on the most common questions — ones that can be answered with a single fact in the knowledge base. We propose CFO, a **C**onditional **Fo**cused neural-network-based approach to answering factoid questions with knowledge bases. Our approach first zooms in a question to find more probable candidate subject mentions, and infers the final answers with a unified conditional probabilistic framework. Powered by deep recurrent neural networks and neural embeddings, our proposed CFO achieves an accuracy of 75.7% on a dataset of 108k questions – the largest public one to date. It outperforms the current state of the art by an absolute margin of 11.8%.

1 Introduction

Community-driven question answering (QA) websites such as Quora, Yahoo-Answers, and Answers.com are accumulating millions of users and hundreds of millions of questions. A large portion of the questions are about facts or trivia. It has been a long pursuit to enable machines to answer such questions automatically.

In recent years, several efforts have been made on utilizing open-domain knowledge bases to answer factoid questions. A knowledge base (KB) consists of structured representation of facts in the form of subject-relation-object triples. Lately, several large-scale general-purpose KBs have been constructed, including YAGO (Suchanek et al., 2007), Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010), and DBpedia (Lehmann et al., 2014). Typically, structured queries with predefined semantics (e.g. SPARQL) can be issued to retrieve specified facts from such KBs. Thus, answering factoid questions will be straightforward once they are converted into the corresponding structured form. However, due to complexity of language, converting natural language questions to structure forms remains an open challenge.

Among all sorts of questions, there is one category that only requires a single fact (triple) in KB as the supporting evidence. As a typical example, the question "*Who created the character Harry Potter*" can be answered with the single fact `(HarryPotter, CharacterCreatedBy, J.K.Rowling)`. In this work, we refer to such questions as *single-fact questions*. Previously, it has been observed that *single-fact questions* constitute the majority of factoid questions in community QA sites (Fader et al., 2013). Despite the simplicity, automatically answering such questions remains far from solved — the latest best result on a dataset of 108k single-fact questions is only 63.9% in terms of accuracy (Bordes et al., 2015).

To find the answer to a single-fact question, it suffices to identify the subject entity and relation (implicitly) mentioned by the question, and then forms a corresponding structured query. The problem can be formulated into a probabilistic form. Given a single-fact question q, finding the subject-relation pair $\hat{s}, \hat{r}$ from the KB $\mathcal{K}$ which maximizes the conditional probability $p(s, r|q)$, i.e.

$$\hat{s}, \hat{r} = \arg\max_{s,r \in \mathcal{K}} p(s, r|q) \quad (1)$$

*Part of the work was done while at Baidu.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 800–810, Berlin, Germany, August 7-12, 2016.

Based on the formulation (1), the central problem is to estimate the conditional distribution $p(s, r|q)$. It is very challenging because of *a*) the vast amount of facts — a large-scale KB such as Freebase contains billions of triples, *b*) the huge variety of language — there are multiple aliases for an entity, and numerous ways to compose a question, *c*) the severe sparsity of supervision — most combinations of s, r, q are not expressed in training data. Faced with these challenges, existing methods have exploited to incorporate prior knowledge into semantic parsers, to design models and representations with better generalization property, to utilize large-margin ranking objective to estimate the model parameters, and to prune the search space during inference. Noticeably, models based on neural networks and distributed representations have largely contributed to the recent progress (see section 2).

In this paper, we propose CFO, a novel method to answer *single-fact questions* with large-scale knowledge bases. The contributions of this paper are,

- we employ a fully probabilistic treatment of the problem with a novel conditional parameterization using neural networks,
- we propose the focused pruning method to reduce the search space during inference, and
- we investigate two variations to improve the generalization of representations for millions of entities under highly sparse supervision.

In experiments, CFO achieves 75.7% in terms of top-1 accuracy on the largest dataset to date, outperforming the current best record by an absolute margin of 11.8%.

2 Related Work

The research of KB supported QA has evolved from earlier domain-specific QA (Zelle and Mooney, 1996; Tang and Mooney, 2001; Liang et al., 2013) to open-domain QA based on large-scale KBs. An important line of research has been trying to tackle the problem by semantic parsing, which directly parses natural language questions into structured queries (Liang et al., 2011; Cai and Yates, 2013; Kwiatkowski et al., 2013; Yao and Van Durme, 2014). Recent progresses include designing KB specific logical representation and parsing grammar (Berant et al., 2013), using distant supervision (Berant et al., 2013), utilizing paraphrase information (Fader et al., 2013; Berant and Liang, 2014), requiring little question-answer pairs (Reddy et al., 2014), and exploiting ideas from agenda-based parsing (Berant and Liang, 2015).

In contrast, another line of research tackles the problem by deep learning powered similarity matching. The core idea is to learn semantic representations of both the question and the knowledge from observed data, such that the correct supporting evidence will be the nearest neighbor of the question in the learned vector space. Thus, a main difference among several approaches lies in the neural networks proposed to represent questions and KB elements. While (Bordes et al., 2014b; Bordes et al., 2014a; Bordes et al., 2015; Yang et al., 2014) use relatively shallow embedding models to represent the question and knowledge, (Yih et al., 2014; Yih et al., 2015) employ a convolutional neural network (CNN) to produce the representation. In the latter case, both the question and the relation are treated as a sequence of letter-trigram patterns, and fed into two parameter shared CNNs to get their embeddings. What's more, instead of measuring the similarity between a question and an evidence triple with a single model as in (Bordes et al., 2015), (Yih et al., 2014; Yih et al., 2015) adopt a multi-stage approach. In each stage, one element of the triple is compared with the question to produce a partial similarity score by a dedicated model. Then, these partial scores are combined to generate the overall measurement.

Our proposed method is closely related to the second line of research, since neural models are employed to learn semantic representations. As in (Bordes et al., 2015; Yih et al., 2014), we focus on *single-fact questions*. However, we propose to use recurrent neural networks (RNN) to produce the question representation. More importantly, our method follows a probabilistic formulation, and our parameterization relies on factors other than similarity measurement.

Besides KB-based QA, our work is also loosely related to work using deep learning systems in QA tasks with free text evidences. For example, (Iyyer et al., 2014) focuses questions from the quiz bowl competition with recursive neural network. New architectures including memory networks (Weston et al., 2015), dynamic memory networks (Kumar et al., 2015), and more (Peng et al., 2015; Lee et al., 2015) have been explored under the `bAbI` syn-

thetic QA task (Weston et al., 2016). In addition, (Hermann et al., 2015) seeks to answer Cloze style questions based on news articles.

3 Overview

In this section, we formally formulate the problem of *single-fact question answering* with knowledge bases. A knowledge base $\mathcal{K}$ contains three components: a set of entities $\mathcal{E}$, a set of relations $\mathcal{R}$, and a set of facts $\mathcal{F} = \{\langle s, r, o\rangle\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$, where $s, o \in \mathcal{E}$ are the subject and object entities, and $r \in \mathcal{R}$ is a binary relation. $\mathrm{E}(r), \mathrm{E}(s)$ are the vector representations of a relation and an entity, respectively. $s \rightarrow r$ indicates that there exists some entity o such that $\langle s, r, o\rangle \in \mathcal{F}$. For single-fact questions, a common assumption is that the answer entity o and some triple $\langle s_i, r_k, o\rangle \in \mathcal{F}$ reside in the given knowledge base. The goal of our model is to find such subject s_i and relation r_k mentioned or implied in the question. Once found, a structured query (e.g. in SPARQL) can be constructed to retrieve the result entity.

3.1 Conditional Factoid Factorization

Given a question q, the joint conditional probability of subject-relation pairs $p(s, r|q)$ can be used to retrieve the answer using the exact inference defined by Eq. (1). However, since there can be millions of entities and thousands of relations in a knowledge base, it is less effective to model $p(s, r|q)$ directly. Instead, we propose a conditional factoid factorization,

$$p(s, r|q) = p(r|q) \cdot p(s|q, r) \tag{2}$$

and utilize two neural networks to parameterize each component, $p(r|q)$ and $p(s|q, r)$, respectively. Hence, our proposed method contains two phases: inferring the implied relation r from the question q, and inferring the mentioned subject entity s given the relation r and the question q.

There is an alternative factorization $p(s, r|q) = p(s|q) \cdot p(r|s, q)$. However, it is rather challenging to estimate $p(s|q)$ directly due to the vast amount of entities ($> 10^6$) in a KB. In comparison, our proposed factorization takes advantage of the relatively limited number of relations (on the order of thousands). What's more, by exploiting additional information from the candidate relation r, it's more feasible to model $p(s|q, r)$ than $p(s|q)$, leading to more robust estimation.

A key difference from prior multi-step approach is that our method do not assume any independence between the target subject and relation given a question, as does in the prior method (Yih et al., 2014). It proves effective in our experiments.

3.2 Inference via Focused Pruning

As defined by the Eq. (1), a solution needs to consider all available subject-relation pairs in the KB as candidates. With a large-scale KB, the number of candidates can be notoriously large, resulting in a extremely noisy candidate pool. We propose a method to prune the candidate space. The pruning is equivalent to a function that takes a KB $\mathcal{K}$ and a question q as input, and outputs a much limited set $\mathcal{C}$ of candidate subject-relation pairs.

$$\mathcal{H}(\mathcal{K}, q) \rightarrow \mathcal{C} \tag{3}$$

$\mathcal{C}_s$ and $\mathcal{C}_r$ are used to represent the subject and relation candidates, respectively.

The fundamental intuition for pruning is that the subject entity must be mentioned by some textual substring (*subject mention*) in the question. Thus, the candidate space can be restricted to entities whose name/alias matches an n-gram of the question, as in (Yih et al., 2014; Yih et al., 2015; Bordes et al., 2015). We refer to this straight-forward method as *N-Gram pruning*. By considering all n-grams, this approach usually achieves a high recall rate. However, the candidate pool is still noisy due to many non-subject-mention n-grams.

Our key idea is to reduce the noise by guiding the pruning method's attention to more probable parts of a question. An observation is that certain parts of a sentence are more likely to be the subject mention than others. For example, "*Harry Potter*" in "*Who created the character Harry Potter*" is more likely than "*the character*", "*character Harry*", etc. Specifically, our method employs a deep network to identify such focus segments in a question. This way, the candidate pool can be not only more compact, but also significantly less noisy.

Finally, combing the ideas of Eq.(2) and (3), we propose an approximate solution to the problem defined by Eq. (1)

$$\hat{s}, \hat{r} \approx \arg\max_{s,r \in \mathcal{C}} p(s|q, r)p(r|q) \tag{4}$$

4 Proposed CFO

In this section, we first review the gated recurrent unit (GRU), an RNN variant extensively used in

this work. Then, we describe the model parameterization of $p(r|q)$ and $p(s|q,r)$, and the focused pruning method in inference.

4.1 Review: Gated Recurrent Units

In this work we employ GRU (Cho et al., 2014) as the RNN structure. At time step t, a GRU computes its hidden state h_t using the following compound functions

$$z = \text{sigmoid}\left(W_{xz}x_t + W_{hz}h_{t-1} + b_z\right) \quad (5)$$

$$r = \text{sigmoid}\left(W_{xr}x_t + W_{hr}h_{t-1} + b_r\right) \quad (6)$$

$$\tilde{h} = \tanh\left(W_{xh}x_t + r \otimes W_{hh}h_{t-1} + b_h\right) \quad (7)$$

$$h_t = z \otimes h_{t-1} + (1 - z) \otimes \tilde{h} \quad (8)$$

where $W_{\{.\}}$, and $b_{\{.\}}$ are all trainable parameters. To better capture the context information on both sides, two GRUs with opposite directions can be combined to form a bidirectional GRU (BiGRU).

4.2 Model Parameterization

Relation network In this work, the probability of relations given a question, $p(r|q)$, is modeled by the following network

$$p_{\theta_r}(r|q) = \frac{\exp\big(v(r,q)\big)}{\sum_{r'}\exp\big(v(r',q)\big)} \quad (9)$$

where the *relation scoring function* $v(r,q)$ measures the similarity between the question and the relation

$$v(r,q) = f(q)^\top \mathrm{E}(r) \quad (10)$$

$\mathrm{E}(r)$ is the trainable embedding of the relation (randomly initialized in this work) and $f(q)$ computes the semantic question embedding. Specifically, the question q is represented as a sequence of tokens (potentially with unknown ones). Then, the question embedding model f consists of a word embedding layer to transform tokens into distributed representations, a two-layer BiGRU to capture the question semantics, and a linear layer to project the final hidden states of the BiGRU into the same vector space as $\mathrm{E}(r)$.

Subject network As introduced in section 3, the factor $p(s|q,r)$ models the fitness of a subject s appearing in the question q, given the main topic is about the relation r. Thus, two forces *a*) the raw context expressed by q, and *b*) the candidate topic described by r, jointly impact the fitness of the subject s. For simplicity, we use two additive terms to model the joint effect

$$p_{\theta s}(s|q,r) = \frac{\exp\big(u(s,r,q)\big)}{\sum_{s'}\exp\big(u(s',r,q)\big)} \quad (11)$$

where $u(s,r,q)$ is the *subject scoring function*,

$$u(s,r,q) = g(q)^\top \mathrm{E}(s) + \alpha h(r,s) \quad (12)$$

$g(q)$ is another semantic question embedding, $\mathrm{E}(s)$ is a vector representation of a subject, $h(r,s)$ is the subject-relation score, and α is the weight parameter used to trade off the two sources.

Firstly, the *context score* $g(q)^\top \mathrm{E}(s)$ models the intrinsic plausibility that the subject s appears in the question q using vector space similarity. As $g(q)^\top \mathrm{E}(s)$ has the same form as equation (10), we let g adpot the same model structure as f. However, initializing $\mathrm{E}(s)$ randomly and training it with supervised signal, just like training $\mathrm{E}(r)$, is insufficient in practice — while a large-scale KB has millions of subjects, only thousands of question-triple pairs are available for training. To alleviate the problem, we seek two potential solutions: *a) pretrained* embeddings, and *b) type vector* representation.

The pretrained embedding approach utilizes unsupervised method to train entity embedings. In particular, we employ the `TransE` (Bordes et al., 2013), which trains the embedings of entities and relations by enforcing $\mathrm{E}(s) + \mathrm{E}(r) = \mathrm{E}(o)$ for every observed triple $(s,r,o) \in \mathcal{K}$. As there exists other improved variants (Gu et al., 2015), `TransE` scales the best when KB size grows.

Alternatively, type vector is a fixed (not trainable) vector representation of entities using type information. Since each entity in the KB has one or more predefined types, we can encode the entity as a vector (bag) of types. Each dimension of a type vector is either 1 or 0, indicating whether the entity is associated with a specific type or not. Thus, the dimensionality of a type vector is equal to the number of types in KB. Under this setting, with $\mathrm{E}(s)$ being a binary vector, let $g(q)$ be a continuous vector with arbitrary value range can be problematic. Therefore, when type vector is used as $\mathrm{E}(s)$, we add a sigmoid layer upon the final linear projection of g, squashing each element of $g(q)$ to the range $[0,1]$.

Compared to the first solution, type vector is fully based on the type profile of an entity, and requires no training. As a benefit, considerably

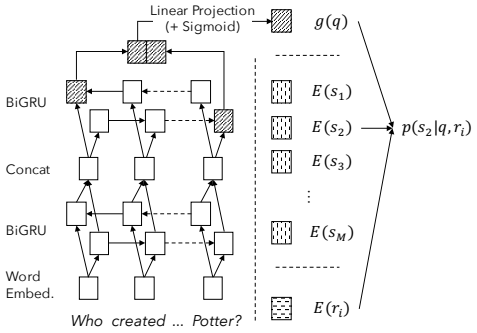


Figure 1: Overall structure of the subject network. Sigmoid layer is added only when type vector is used as $\mathrm{E}(s)$.

fewer parameters are needed. Also, given the type information is discriminative enough, using type vector will lead to easier generalization. However, containing only type information can be very restrictive.

In addition to the context score, we use the *subject-relation score* $h(r, s)$ to capture the compatibility that s and r show up together. Intuitively, for an entity to appear in a topic characterized by a relation, a necessary condition will be that the entity has the relation connected to it. Inspired by this structural regularity, in the simplest manner, we instantiate the idea with an indicator function,

$$h(r, s) = \mathbb{1}(s \rightarrow r) \tag{13}$$

As there exists other more sophisticated statistical parameterizations, the proposed approach is able to capture the core idea of the structural regularity without any parameter. Finally, putting two scores together, Fig.1 summarizes the overall structure of the subject network.

4.3 Focused Pruning

As discussed in section 3.2, N-Gram pruning is still subject to large amount of noise in inference due to many non-subject-mention n-grams. Motivated by this problem, we propose to reduce such noise by focusing on more probable candidates using a special-purpose sequence labeling network. Basically, a sequence labeling model is trained to tag some consecutive tokens as the subject mention. Following this idea, during inference, only the most probable n-gram predicted by the model will be retained, and then used as the subject mention to generate the candidate pool $\mathcal{C}$. Hence, we refer to this method as *focused pruning*. Formally, let $\mathcal{W}(q)$ be all the n-grams of the question q, $p(w|q)$ be the probability that the n-gram w is the subject mention of q, the focused pruning function $\mathcal{H}_s$ is defined as

$$\begin{aligned} \hat{w} &= \arg\max_{w \in \mathcal{W}(q)} p_\kappa(w|q) \\ \mathcal{C} &= \{(s, r) : \mathcal{M}(s, \hat{w}), s \rightarrow r\} \end{aligned} \tag{14}$$

where $\mathcal{M}(s, \hat{w})$ represents some predefined match between the subject s and the predicted subject mention $\hat{w}$. Intuitively, this pruning method resembles the human behavior of first identifying the subject mention with the help of context, and then using it as the key word to search the KB.

To illustrate the effectiveness of this idea, we parameterize $p_\kappa(w|q)$ with a general-purpose neural labeling model, which consists of a word embedding layer, two layers of BiGRU, and a linear-chain conditional random field (CRF). Thus, given a question q of length T, the score of a sequence label configuration $y \in \mathbb{R}^T$ is

$$s(y, q) = \sum_{t=1}^{T} \mathbf{H}(q)_{t,y_t} + \sum_{t=2}^{T} \mathbf{A}_{y_{t-1},y_t}$$

where $\mathbf{H}(q)$ is the hidden output of the top-layer BiGRU, $\mathbf{A}$ is the transition matrix possesed by the CRF, and $[\cdot]_{i,j}$ indicates the matrix element on row i collum j.

Finally, the match function $\mathcal{M}(s, \hat{w})$ is simply defined as either strict match between an alias of s and $\hat{w}$, or approximate match provided by the Freebase entity suggest API [1]. Note that more elaborative match function can further boost the performance, but we leave it for future work.

5 Parameter Estimation

In this section, we discuss the parameter estimation for the neural models presented in section 4.

With standard parameterization, the focused labeling model $p_\kappa(w|q)$ can be directly trained by maximum likelihood estimation (MLE) and back-propagation. Thus, we omit the discussion here, and refer readers to (Huang et al., 2015) for details. Also, we leave the problem of how to obtain the training data to section 6.

5.1 Decomposable Log-Likelihood

To estimate the parameters of $p_{\theta_r}(r|q)$ and $p_{\theta_s}(s|r, q)$, MLE can be utilized to maximize the empirical (log-)likelihood of subject-relation pairs

[1] The approximate match is used only when there is no strict match. The suggest API takes a string as input, and returns no more than 20 potentially matched entities.

given the associated question. Following this idea, let $\{s^{(i)}, r^{(i)}, q^{(i)}\}_{i=1}^{N}$ be the training dataset, the MLE solution takes the form

$$\theta^{\text{MLE}} = \arg\max_{\theta_r, \theta_s} \sum_{i=1}^{N} \Big(\log p_{\theta_r}(r^{(i)}|q^{(i)}) + \log p_{\theta_s}(s^{(i)}|r^{(i)}, q^{(i)}) \Big) \tag{15}$$

Note that there is no shared parameter between $p_{\theta_s}(s|q,r)$ and $p_{\theta_r}(r|q)$. [2] Therefore, the same solution can be reached by separately optimizing the two log terms, i.e.

$$\begin{aligned} \theta_r^{\text{MLE}} &= \arg\max_{\theta_r} \sum_{i=1}^{N} \log p_{\theta_r}(r^{(i)}|q^{(i)}) \\ \theta_s^{\text{MLE}} &= \arg\max_{\theta_s} \sum_{i=1}^{N} \log p_{\theta_s}(s^{(i)}|r^{(i)}, q^{(i)}) \end{aligned} \tag{16}$$

It is important to point out that the decomposability does not always hold. For example, when the parametric form of $h(s,r)$ depends on the embedding of r, the two terms will be coupled and joint optimization must be performed. From this perspective, the simple form of $h(s,r)$ also eases the training by inducing the decomposability.

5.2 Approximation with Negative Samples

As the two problems defined by equation (16) take the standard form of classification, theoretically, cross entropy can used as the training objective. However, computing the partition function is often intractable, especially for $p_{\theta_s}(s|r,q)$, since there can be millions of entities in the KB. Faced with this problem, classic solutions include contrastive estimation (Smith and Eisner, 2005), importance sampling approximation (Bengio et al., 2003), and hinge loss with negative samples (Collobert and Weston, 2008).

In this work, we utilize the hinge loss with negative samples as the training objective. Specifically, the loss function w.r.t θ_r has the form

$$\mathcal{L}(\theta_r) = \sum_{i=1}^{N} \sum_{j=1}^{M_r} \max\big[0, \gamma_r - v(r^{(i)}, q^{(i)}) + v(r^{(j)}, q^{(i)})\big] \tag{17}$$

where $r^{(j)}$ is one of the M_r negative samples (i.e. $s^{(i)} \not\rightarrow r^{(j)}$) randomly sampled from $\mathcal{R}$, and γ_r is the predefined margin. Similarly, the loss function w.r.t θ_s takes the form

$$\mathcal{L}(\theta_s) = \sum_{i=1}^{N} \sum_{j=1}^{M_s} \max\big[0, \gamma_s - u(s^{(i)}, r^{(i)}, q^{(i)}) + u(s^{(j)}, r^{(i)}, q^{(i)})\big] \tag{18}$$

Despite the negative sample based approximation, there is another practical difficulty when type vector is used as the subject representation. Specifically, computing the value of $u(s^{(j)}, r^{(i)}, q^{(i)})$ requires to query the KB for all types of each negative sample $s^{(j)}$. So, when M_s is large, the training can be extremely slow due to the limited bandwidth of KB query. Consequently, under the setting of type vector, we instead resort to the following type-wise binary cross-entropy loss

$$\tilde{\mathcal{L}}(\theta_s) = -\sum_{i=1}^{N} \sum_{k=1}^{K} \Big(\text{E}(s^{(i)})_k \log g(q^{(i)})_k + \big[1 - \text{E}(s^{(i)})_k\big] \log \big[1 - g(q^{(i)})_k\big] \Big) \tag{19}$$

where K is the total number of types, $g(q)_k$ and $\text{E}(s^{(i)})_k$ are the k-th element of $g(q)$ and $\text{E}(s^{(i)})$ respectively. Intuitively, with sigmoid squashed output, $g(q)$ can be regarded as K binary classifiers, one for each type. Hence, $g(q)_k$ reprents the predicted probability that the subject is associated with the k-th type.

6 Experiments

In this section, we conduct experiments to evaluate the proposed system empirically.

6.1 Dataset and Knowledge Base

We train and evaluate our method on the SIMPLEQUESTIONS dataset[3] — the largest question-triple dataset. It consists of 108,442 questions written in English by human annotators. Each question is paired with a subject-relation-object triple from Freebase. We follow the same splitting for training (70%), validation (10%) and testing (20%) as (Bordes et al., 2015). We use the same subset of Freebase (`FB5M`) as our knowledge base so that the results are directly comparable. It includes 4,904,397 entities, 7,523 relations, and 22,441,880 facts.

There are alternative datasets available, such as WebQuestions (Berant et al., 2013) and

[2] Word embeddings are not shared across models.

[3] `https://research.facebook.com/researchers/1543934539189348`

Free917 (Cai and Yates, 2013). However, these datasets are quite restricted in sample size — the former includes 5,810 samples (train + test) and the latter includes 917 ones. They are fewer than the number of relations in Freebase.

To train the focused labeling model, the information about whether a word is part of the subject mention is needed. We obtain such information by reverse linking from the ground-truth subject to its mention in the question. Given a question q corresponding to subject s, we match the name and aliases of s to all n-grams that can be generated from q. Once a match is found, we label the matched n-gram as the subject mention. In the case of multiple matches, only the longest matched n-gram is used as the correct one.

6.2 Evaluation and Baselines

For evaluation, we consider the same metric introduced in (Bordes et al., 2015), which takes the prediction as correct if both the subject and relation are correctly retrieved. Based on this metric, we compare CFO with a few baseline systems, which include both the Memory Network QA system (Bordes et al., 2015), and systems with alternative components and parameterizations from existing work (Yih et al., 2014; Yih et al., 2015). We did not compare with alternative subject networks because the only existing method (Yih et al., 2014) relies on unique textual name of each entity, which does not generally hold in knowledge bases (except in REVERB). Alternative approaches for pruning method, relation network, and entity representation are described below.

Pruning methods We consider two baseline methods previously used to prune the search space. The first baseline is the N-Gram pruning method introduced in Section 3, as it has been successfully used in previous work (Yih et al., 2014; Yih et al., 2015). Basically, it establishes the candidate pool by retaining subject-relation pairs whose subject can be linked to one of the n-grams generated from the question. The second one is N-Gram+, a revised version of the N-Gram pruning with additional heuristics (Bordes et al., 2015). Instead of considering all n-grams that can be linked to entities in KB, heuristics related to overlapping n-grams, stop words, interrogative pronouns, and so on are exploited to further shrink the n-gram pool. Accordingly, the search space is restricted to subject-relation pairs whose subject can be linked to one of the remaining n-grams after applying the heuristic filtering.

Relation scoring network We compare our proposed method with two previously used models. The first baseline is the embedding average model (Embed-AVG) used in (Bordes et al., 2014a; Bordes et al., 2014b; Bordes et al., 2015). Basically, it takes the element-wise average of the word embeddings of the question to be the question representation. The second one is the letter-tri-gram CNN (LTG-CNN) used in (Yih et al., 2014; Yih et al., 2015), where the question and relation are separately embedded into the vector space by two parameter shared LTG-CNNs. [4] In addition, (Yih et al., 2014; Yih et al., 2015) observed better performance of the LTG-CNN when substituting the subject mention with a special symbol. Naturally, this can be combined with the proposed focused labeling, since the latter is able to identify the potential subject mention in the question. So, we train another LTG-CNN with symbolized questions, which is denoted as LTG-CNN+. Note that this model is only tested when the focused labeling pruning is used.

Entity representation In section 4.2, we describe two possible ways to improve the vector representation of the subject, `TransE` pretrained embedding and type vectors. To evaluate their effectiveness, we also include this variation in the experiment, and compare their performance with randomly initialized entity embeddings.

6.3 Experiment Setting

During training, all word embeddings are initialized using the pretrained GloVe (Pennington et al., 2014), and then fine tuned in subsequent training. The word embedding dimension is set to 300, and the BiGRU hidden size 256. For pretraining the entity embeddings using `TransE` (see section 4.2), only triples included in `FB5M` are used. All other parameters are randomly initialized uniformly from $[-0.08, 0.08]$, following (Graves, 2013). Both hinge loss margins γ_s and γ_r are set to 0.1. Negative sampling sizes M_s and M_r are both 1024.

For optimization, parameters are trained using mini-batch AdaGrad (Duchi et al., 2011) with Momentum (Pham et al., 2015). Learning rates are

[4] In Freebase, each predefined relation has a *single* human-recognizable reference form, usually a sequence of words.

Pruning Method	Relation Network	Entity Representation		
		Random	Pretrain	Type Vec
Memory Network		62.9	63.9*	
N-Gram	Embed-AVG	39.4	42.2	50.9
	LTG-CNN	32.8	36.8	45.6
	BiGRU	43.7	46.7	55.7
N-Gram+	Embed-AVG	53.8	57.0	58.7
	LTG-CNN	46.3	50.9	56.0
	BiGRU	58.3	61.6	62.6
Focused Pruning	Embed-AVG	71.4	71.7	72.1
	LTG-CNN	67.6	67.9	68.6
	LTG-CNN+	70.2	70.4	71.1
	BiGRU	75.2	75.5	**75.7**

Table 1: Accuracy on SIMPLEQUESTIONS testing set. * indicates using ensembles. N-Gram+ uses additional heuristics. The proposed CFO (focused pruning + BiGRU + type vector) achieves the top accuracy.

tuned to be 0.001 for question embedding with type vector, 0.03 for LTG-CNN methods, and 0.02 for rest of the models. Momentum rate is set to 0.9 for all models, and the mini-batch size is 256. In addition, vertical dropout (Pham et al., 2014; Zaremba et al., 2014) is used to regularize all BiGRUs in our experiment. [5]

6.4 Results

Trained on 75,910 questions, our proposed model and baseline methods are evaluated on the testing set with 21,687 questions. Table 1 presents the accuracy of those methods. We evaluated all combinations of pruning methods, relation networks and entity representation schemes, as well as the result from memory network, as described in Section 6.1. CFO (focused pruning + BiGRU + type vector) achieves the best performance, outperforming all other methods by substantial margins.

By inspecting vertically within each cell in Table 1, for the same pruning methods and entity representation scheme, BiGRU based relation scoring network boosts the accuracy by 3.5 % to 4.8% compared to the second best alternative. This evidence suggests the superiority of RNN in capturing semantics of question utterances. Surprisingly, it turns out that Embed-AVG achieves better performance than the more complex LTG-CNN.

By inspecting Table 1 horizontally, type vector based representation constantly leads to better performance, especially when N-Gram pruning is used. It suggests that under sparse supervision, training high-quality distributed knowledge representations remains a challenging problem. That said, pretraining entity embeddings with `TransE` indeed gives better performance compared to random initialization, indicating the future potential of unsupervised methods in improving continuous knowledge representation.

In addition, all systems using our proposed focused pruning method outperform their counterparts with alternative pruning methods. Without using ensembles, CFO is already better than the memory network ensembles by 11.8%. It substantiates the general effectiveness of the focused pruning with subject labeling method regardless of other sub-modules.

6.5 Effectiveness of Pruning

According to the results in section 6.4, the focused pruning plays a critical role in achieving the best performance. To get a deeper understanding of its effectiveness, we analyze how the pruning methods affect the accuracy of the system. Due to space limit, we focus on systems with BiGRU as the relation scoring function and type vector as the entity representation.

Table 2 summarizes the recall — the percentage of pruned subject-relation candidates containing the answer — and the resulting accuracy. The single-subject case refers to the scenario that there is only one candidate entity in $\mathcal{C}_s$ (possibly with multiple relations), and the multi-subject case means there are multiple entities in $\mathcal{C}_s$. As the table shows, focused pruning achieves comparable recall rate to N-Gram pruning.[6] Given the state-of-the-art performance of sequence labeling systems, this result should not be surprising. Thus, the difference in performances entirely comes from their resulting accuracy. Notice that there exists a huge accuracy gap between the two cases. Essentially, in the single-candidate case, the system only need to identify the relation based on the more robust model $p_{\theta_r}(r|q)$. In contrast, under the multi-candidate case, the system also relies on $p_{\theta_s}(s|q,r)$, which has significantly more parameters to estimate, and thus is less robust. Consequently, by only focusing on the most probable sub-string, the proposed focused pruning produces much more single-candidate situations, leading to a better overall accuracy.

[5] For more details, source code is available at `http://zihangdai.github.io/cfo` for reference.

[6] Less than 3% of the recalled candidates rely on approximate matching in the focused pruning.

Pruning method	Pruning recall	Inference accuracy within the recalled		Overall accuracy
		Single-subject case	Multi-subject case	
N-Gram	94.8%	18 / 21 = 85.7%	12051 / 20533 = 58.7%	55.7%
N-Gram+	92.9%	126 / 138 = 91.3%	13460 / 20017 = 67.2%	62.6%
Focused pruning	94.9%	9925 / 10705 = 92.7%	6482 / 9876 = 65.6%	**75.7%**

Table 2: Comparison of different space pruning methods. N-Gram+ uses additional heuristics. Single- and multi-subject refers to the number of distinct subjects in candidates. The proposed focused pruning achieves best scores.

6.6 Additional Analysis

In the aforementioned experiments, we have kept the focused labeling model and the subject scoring network fixed. To further understand the importance and sensitivity of this specific model design, we investigate some variants of these two models.

Alternative focus with CRF RNN-CRF based models have achieved the state-of-the-art performance on various sequence labeling tasks (Huang et al., 2015; Lu et al., 2015). However, the labeling task we consider here is relatively unsophisticated in the sense that there are only two categories of labels - part of subject string (`SUB`) or not (`O`). Thus, it's worth investigating whether RNN (BiGRU in our case) is still a critical component when the task gets simple. Hence, we establish a CRF baseline which uses traditional features as input. Specifically, the model is trained with `Stanford CRF-NER` toolkit [7] on the same reversely linked labeling data (section 6.1). For evaluation, we directly compare the sentence level accuracy of these two models on the test portion of the labeling data. A sentence labeling is considered correct only when all tokens are correctly labeled. [8] It turns out the RNN-CRF achieves an accuracy of 95.5% while the accuracy of feature based CRF is only 91.2%. Based on the result, we conclude that BiGRU plays a crucial role in our focused pruning module.

Subject scoring with average embedding As discussed in section 4.2, the subject network g is chosen to be the same as f, mainly relying on a two-layer BiGRU to produce the semantic question embeding. Although it is a natural choice, it remains unclear whether the final performance is sensitive to this design. Motivated by this question, we substitute the BiGRU with an Embed-AVG model, and evalute the system performance.

[7] `http://nlp.stanford.edu/software/CRF-NER.shtml`

[8] As F-1 score is usually used as the metric for sequence labeling, sentence level accuracy is more informative here.

Relation Network	**Subject Network**	
	Embed-AVG	BiGRU
Embed-AVG	71.6	72.1
LTG-CNN	68.0	68.6
LTG-CNN+	70.4	71.1
BiGRU	75.4	75.7

Table 3: System performance with different subject network structures.

For this experiment, we always use focused pruning and type vector, but vary the structure of the relation scoring network to allow high-order interaction across models. The result is summarized in Table 3. Insepcting the table horizontally, when BiGRU is employed as the subject network, the accuracy is consistently higher regardless of relation network structures. However, the margin is quite narrow, especially compared to the effect of varying the relation network structure the same way. We suspect this difference reflects the fact that modeling $p(s|r, q)$ is intrinsically more challenging than modeling $p(r|q)$. It also suggests that learning smooth entity representations with good discriminative power remains an open problem.

7 Conclusion

In this paper, we propose CFO, a novel approach to single-fact question answering. We employ a conditional factoid factorization by inferring the target relation first and then the target subject associated with the candidate relations. To resolve the representation for millions of entities, we proposed type-vector scheme which requires no training. Our focused pruning largely reduces the candidate space without loss of recall rate, leading to significant improvement of overall accuracy. Compared with multiple baselines across three aspects, our method achieves the state-of-the-art accuracy on a 108k question dataset, the largest publicly available one. Future work could be extending the proposed method to handle more complex questions.

References

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

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*, volume 7, page 92.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 615–620.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In *Machine Learning and Knowledge Discovery in Databases*, pages 165–180. Springer.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL (1)*, pages 423–433. Citeseer.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*, volume 5, page 3.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL (1)*, pages 1608–1618. Citeseer.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Kelvin Gu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In *Empirical Methods in Natural Language Processing*.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2015. Reasoning in vector space: An exploratory study of question answering. *arXiv preprint arXiv:1511.06426*.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2014. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 5:1–29.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Zefu Lu, Lei Li, and Wei Xu. 2015. Twisted recurrent network for named entity recognition. In *Bay Area Machine Learning Symposium*.

Baolin Peng, Zhengdong Lu, Hang Li, and Kam-Fai Wong. 2015. Towards neural network-based reasoning. *arXiv preprint arXiv:1508.05508*.

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

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 285–290. IEEE.

Hieu Pham, Zihang Dai, and Lei Li. 2015. On optimization algorithms for recurrent networks with long short-term memory. In *Bay Area Machine Learning Symposium*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Noah A Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 354–362. Association for Computational Linguistics.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Lappoon R Tang and Raymond J Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Machine Learning: ECML 2001*, pages 466–477. Springer.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *International Conference on Learning Representations (ICLR2015)*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards ai-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations (ICLR2016)*.

Min-Chul Yang, Nan Duan, Ming Zhou, and Hae-Chang Rim. 2014. Joint relational embeddings for knowledge-based question answering. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 645–650.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *Proceedings of ACL*.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of ACL*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of ACL*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055.

Verbs Taking Clausal and Non-Finite Arguments as Signals of Modality – Revisiting the Issue of Meaning Grounded in Syntax

Judith Eckle-Kohler
Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt
www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de

Abstract

We revisit Levin's theory about the correspondence of verb meaning and syntax and infer semantic classes from a large syntactic classification of more than 600 German verbs taking clausal and non-finite arguments. Grasping the meaning components of Levin-classes is known to be hard. We address this challenge by setting up a multi-perspective semantic characterization of the inferred classes. To this end, we link the inferred classes and their English translation to independently constructed semantic classes in three different lexicons – the German wordnet GermaNet, VerbNet and FrameNet – and perform a detailed analysis and evaluation of the resulting German–English classification (available at www.ukp.tu-darmstadt.de/modality-verbclasses/).

1 Introduction

Verbs taking clausal and non-finite arguments add a further meaning component to their embedded argument. For example, the embedded argument is realized as *that*-clause in (1) and (2), but *understand* in (1) marks it as factual and *hope* in (2) as uncertain. The verb *pretend* in (3) realizes its embedded argument as non-finite construction and marks it as non-factual.

(1) *He understands* ***that his computer has a hardware problem***.

(2) *She hopes* ***that her experience will help others***.

(3) *He pretends* ***to take notes on his laptop****, but really is updating his Facebook profile.*

The entities expressed by embedded clausal and non-finite arguments are also called "abstract object" (AO) in the rest of this paper (following Asher (1993)); we will use the linguistic term "modality" (Hacquard, 2011) to subsume the meanings (such as factuality, non-factuality and uncertainty) denoted by AO-selecting verbs.

As AO-selecting verbs can change the meaning of a text in important ways, text understanding systems should be sensitive to them. In particular, classifications of AO-selecting verbs according to semantic criteria are important knowledge sources for a wide range of NLP applications, such as event tagging (Saurí et al., 2005), commited belief tagging (Prabhakaran et al., 2010), reported speech tagging (Krestel et al., 2008), the detection of uncertainty (Szarvas et al., 2012) and future-oriented content (Eckle-Kohler et al., 2008), textual entailment (Saurí and Pustejovsky, 2007; Lotan et al., 2013), or determining the degree of factuality of a given text (Saurí and Pustejovsky, 2012; de Marneffe et al., 2012). Accordingly, various semantic classifications of AO-selecting verbs have been developed, e.g., (Kiparsky and Kiparsky, 1970; Karttunen, 1971; Karttunen, 2012), some of them explicitly in the context of NLP (Nairn et al., 2006; Saurí, 2008).

However, these classifications are constructed manually and often quite limited in coverage. Consequently, extending or adapting them to specific domains or other languages is a major issue.

We propose to address this issue by exploiting the relationship between the syntactic behavior of verbs and their meaning following Levin's theory (Levin, 1993). This has not been done yet for verbs signaling modality, as far as we are aware. For the particular category of AO-selecting verbs, Levin's theory allows constructing verb classifications in a purely syntax-driven way, i.e. inducing semantic classes from syntactically defined

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 811–822,
Berlin, Germany, August 7-12, 2016.

classes, and thus possibly also extending given classes using large corpora.[1]

While the appeal of Levin's hypotheses is clear, we are aware of a major difficulty, making our approach a challenging research problem: it is very hard to grasp the precise meaning components which are to be associated with a syntactic "Levin" class. At the same time, it is vital to have a good semantic characterization of the meaning components in order to apply such classes to NLP tasks in an informed way.

We address these issues and make the following contributions: (i) We consider a purely syntactic classification of more than 600 German AO-selecting verbs and induce semantic classes based on findings from formal semantics about correspondences between verb syntax and meaning. This yields an initial description of the meaning components associated with the classes, along with a tentative class name. (ii) In a second step, we refine and extend the semantic characterization of the verb classes by translating it to English and linking it to existing semantic classes in lexical resources at the word sense level: we consider the coarse semantic fields in the German wordnet GermaNet (Kunze and Lemnitzer, 2002), the verb classes in the English lexicon VerbNet (Kipper et al., 2008), and the semantic frames in the English lexicon FrameNet (Baker et al., 1998). As a result, we obtain a detailed semantic characterization of the verb classes, as well as insights into the validity of Levin's theory across the related languages German and English. (iii) We also perform a task-oriented evaluation of the verb classes in textual entailment recognition, making use of insights from the previous two steps. The results suggest that the verb classes might be a promising resource for this task, for German and for English.

2 Related Work

This section summarizes related work about the correspondence between verb meaning and syntax and discusses related work on modality in NLP.

Syntactic Reflections of Verb Meaning Semantic verb classifications that are grounded in lexical-syntactic properties of verbs are particularly appealing, because they can automatically be recovered in corpora based on syntactic features. The most well known verb classification based on correspondences between verb syntax and verb meaning is Levin's classification (Levin, 1993). According to Levin (2015a), verbs that share common syntactic argument alternation patterns also have particular meaning components in common, thus they can be grouped into a semantic verb class. For example, verbs participating in the dative alternation[2] can be grouped into a semantic class of verbs sharing the particular meaning component "change of possession", thus this shared meaning component characterizes the semantic class. Recent work on verb semantics provides additional evidence for this correspondence of verb syntax and meaning: Hartshorne et al. (2014) report that the syntactic behavior of some verbs can be predicted based on their meaning.

VerbNet is a broad-coverage verb lexicon organized in verb classes based on Levin-style syntactic alternations: verbs with common subcategorization frames and syntactic alternation behavior that also share common semantic roles are grouped into VerbNet classes. VerbNet not only includes the verbs from the original verb classification by Levin, but also more than 50 additional verb classes (Kipper et al., 2006) automatically acquired from corpora (Korhonen and Briscoe, 2004). These classes contain many AO-selecting verbs that were not covered by Levin's classification. However, VerbNet does not provide information about the modal meaning of AO-selecting verbs and does not reflect fine-grained distinctions between various kinds of modality.

There is also some criticism in previous work regarding the validity of Levin's approach. Baker and Ruppenhofer (2002) and Schnorbusch (2004) both discuss various issues with Levin's original classification, in particular the difficulty to grasp the meaning components, which are to be associated with a Levin class.

While approaches to exploit the syntactic behavior of verbs for the automatic acquisition of semantic verb classes from corpora have been developed in the past, they were used to recover only small verb classifications: Schulte im Walde (2006)'s work considered a semantically balanced set of 168 German verbs, Merlo and Stevenson (2001) used 60 English verbs from three particular semantic classes.

In contrast to previous work, we consider a large

[1] Abstract objects already characterize the possible semantic roles to a certain extent.

[2] These verbs can realize an argument syntactically either as noun phrase or as prepositional phrase with *to*.

set of more than 600 German AO-selecting verbs and focus on their modal meaning (i.e., expressing factuality or uncertainty).

Related Work on Modality in NLP Previous work in NLP on the automatic (and manual) annotation of modality has often tailored the concept of modality to particular applications. Szarvas et al. (2012) introduce a taxonomy of different kinds of modality expressing uncertainty, such as deontic, bouletic, abilitative modality, and use it for detecting uncertainty in an Information Extraction setting. Their uncertainty cues also include verbs.

Saurí and Pustejovsky (2012) use discrete values in a modality continuum ranging from uncertain to absolutely certain in order to automatically determine the factuality of events mentioned in text. Their automatic approach is based on the FactBank corpus (Saurí and Pustejovsky, 2009), a corpus of newswire data with manually annotated event mentions. For the factuality annotation of the event mentions, the human annotators were instructed to primarily base their decision on lexical cues. For example, they used verbs of belief and opinion, perception verbs, or verbs expressing proof.

Nissim et al. (2013) introduce an annotation scheme for the cross-linguistic annotation of modality in corpora. Their annotation scheme defines two dimensions which are to be annotated (called layers): factuality (characterizing the embedded proposition or concept) and speaker's attitude (characterizing the embedding predicate). Their annotation scheme starts from a fixed set of modal meanings and aims at finding previously unknown triggers of modality. However, some modal meanings are not distinguished, in particular those involving future-orientation. A classification approach grounded in syntax – as in our work – can be considered as complementary: it starts from the syntactic analysis of a large set of trigger words, and induces a broad range of modal meanings based on correspondences between verbs syntax and meaning.

Our semantic classification for AO-selecting verbs covers a *wide range* of different kinds of modality in text, thus considerably extending previous work.

3 Inferring Semantic Verb Classes

In this section, we infer semantic verb classes from the syntactic alternation behavior of a large dataset of German AO-selecting verbs. The research hypotheses underlying our method can be summarized as follows: There are correspondences between verb syntax and meaning: certain syntactic alternations correspond to particular meaning components (Levin, 2015a).

3.1 German Subcategorization Lexicon

We consider a set of 637 AO-selecting verbs given in (Eckle-Kohler, 1999). These verbs are a subset of a subcategorization lexicon (i.e., pairs of lemma and subcategorization frame) that has automatically been extracted from large newspaper corpora using a shallow regular expression grammar covering more than 240 subcategorization frames (short: subcat frames). All the subcat frames extracted for a given verb were manually checked and only the correct ones were included in the final lexicon, because high quality lexical information was crucial in the target application Lexical Functional Grammar parsing.[3]

Eckle-Kohler (1999) specified the alternation behavior of each AO-selecting verb regarding different types of clausal and non-finite arguments, yielding a *syntactic signature* for each verb (e.g., 111101 for the verb *einsehen (realize)* using the encoding in Table 1, top to bottom corresponding to left to right).[4] For this, each verb was inspected regarding its ability to take any of the considered clausal and non-finite constructions as argument – either on the basis of the automatically acquired subcat frames or by making use of linguistic introspection. Linguistic introspection is necessary to reliably identify non-possible argument types, since missing subcat frames that were not extracted automatically are not sufficient as evidence.

Although there are 64 possible syntactic signatures according to basic combinatorics, in the data only 46 signatures were found, which group the verbs into 46 classes. While Eckle-Kohler (1999) points out a few semantic characteristics of these classes, most of them lack a semantic characterization. Our goal is to address this gap and to infer shared meaning components for all the classes.

[3]Today, this lexicon is part of the larger resource "IMSLex German Lexicon" (Fitschen, 2004).

[4]The automatically extracted subcategorization lexicon also contains adjectives and nouns taking clausal or infinitival arguments. However, many of the 1191 nouns and 666 adjectives are derived from verbs, which makes them the central word class.

Argument Type	Y/N	Example
daß(that)-clause	1/0	*sehen (see)*
zu(to)-infinitive, present	1/0	*versuchen (try)*
zu(to)-infinitive, past	1/0	*bereuen (regret)*
wh-clause	1/0	*einsehen (realize)*
ob(whether/if)-clause	1/0	*fragen (ask)*
declarative clause	1/0	*schreien (shout)*

Table 1: Clausal and infinitival arguments distinguished in the syntactic classification; possibility of each type is encoded as 1 (possible) or 0 (not possible).

For this, we use linguistic research findings as described in the next section.

3.2 Findings from Formal Semantics

We employ the following findings on correspondences between verb meaning and syntax in order to infer semantic classes from the syntactic signatures. This gives also rise to tentative names (labels) for the corresponding meaning components.

Factuals: the *that-wh* and the *that-wh/if* alternation. Verbs that are able to alternatively take *that* and *wh*-clauses coerce the embedded interrogative and declarative clauses into factual AOs, corresponding to a particular fact (Ginzburg, 1996). Among the verbs showing the *that-wh* alternation are the well-known factive verbs (Kiparsky and Kiparsky, 1970) (e.g., *She proves that she exists.* vs. *She proves who she is.* vs. *He proves whether he can mine gold.*).

There is a further distinction among these verbs regarding the ability to take an embedded *if/whether*-question: Schwabe and Fittler (2009) show that the *that-wh/if* alternation is connected to objective verbs entailing the existence of an independent witness, whereas the *that-wh* alternation (i.e., an *if/whether*-question is not possible) occurs with non-objective verbs (e.g., *He regrets whom he ended up with.* vs. ⋆*He regrets whether he ended up playing this game.*).

"Aspectuals": the inability to take *that*-clauses and *to*-infinitives in the past tense. Recently, linguistic research has increasingly addressed particular semantic aspects of *to*-infinitives. Kush (2011) has investigated AOs that can neither be realized as *that*-clause nor as *to*-infinitive in the past tense (e.g., *She hesitates to answer.* vs. ⋆*She hesitates to have answered.*[7] vs. ⋆*She hesitates that ...*) These AOs are selected by control verbs[8] and can be characterized as mental actions. Kush (2011) points out that the verbs selecting those AOs have an aspectual meaning in common.

Future orientation: *to*-infinitives in the present tense and the inability to take *to*-infinitives in the past tense. Laca (2013) has investigated verbs across English and Spanish that embed future-oriented AOs. Only future-oriented AOs can be used with future-oriented adverbials, such as *tomorrow*, and these AOs are often realized as non-finite constructions, e.g., *to*-infinitives. She points out that not only control verbs take future-oriented AOs, but also verbs expressing attitudes of preference. This finding implies that such future-oriented AOs are typically incompatible with past-oriented adverbials (e.g., *yesterday*) and verb forms in the past tense (e.g., ⋆*She plans having finished the assignment yesterday.*).

3.3 Mapping to Meaning Components

We automatically infer semantic classes based on a manually constructed mapping between the syntactic signatures from Eckle-Kohler (1999) and the meaning components grounded in syntax summarized in Section 3.2.[9]

We constructed this mapping in two steps: In a first step, the signatures are aligned to the meaning components from Section 3.2 based on substrings of the signatures: future-orientation matches the 110 prefix, aspectual the 010 prefix, and factuality matches 1's in fourth or fifth position. It is important to point out that future-orientation can be combined with factuality: this corresponds to an independent matching of the 110 prefix and the factuality substring. While this combination may seem contradictory, it reflects the lexical data and shows that also weak forms of factuality ("it will most likely be factual at some point in the future") are expressed in language.

In a second step, the pre-aligned signatures are merged, if the remaining slots of the signature are either 1 or 0 (i.e. the respective argument types can or can not occur); in the resulting merged sig-

[7] This is the literal translation of the German equivalent to English. In English, the ing-form in the past would be more typical instead of a *to*-infinitive in the past tense.

[8] "Control" refers to the co-reference between the implicit subject of the infinitival argument and syntactic arguments in the main clause, either the subject (subject control) or direct object (object control).

[9] We did not consider verbs can be used with all kinds of clausal and infinitival arguments, such as the majority of communication verbs (e.g., *comment, whisper*).

signature	#verbs – examples	meaning components	semantic characterization (#linked verbs)
010 ---	36 (6%) – *wagen (dare), zögern (hesitate), weigern (refuse)*	**aspectual**: verbs expressing the ability of doing an action	**VN** (2): consider-29.9, wish-62; **FN** (2): purpose, cogitation
110 0--	195 (31%) – *anbieten (offer), empfehlen (recommend), fordern (demand)*	**future-oriented**: verbs marking AOs as anticipated, planned	**VN** (89): force-59, forbid-67, wish-62, promote-102, urge-58.1, order-60, admire-31.2, order-60, promise-37.13 ; **FN** (43): request, preventing
000 **11**-	15 (2%) – *nachfragen (inquire), anfragen (ask)*	**interrogative**: verbs marking AOs as under investigation	**VN** (3): estimate-34.2, inquire-37.1.2, order-60; **FN** (1): questioning, request
111 **1**--	122 (19%) – *bedauern (regret), überwinden (overcome), danken (thank)*	**wh-factual**: opinion verbs marking AOs as factual	**VN** (45): transfer-mesg-37.1.1, wish-62, admire-31.2, complain-37.8, conjecture-29.5, say-37.7; **FN** (18): statement, reveal-secret
110 10-	30 (5%) – *befürworten (approve), verteidigen (defend), loben (praise)*	**future-oriented wh-factual**: opinion verbs marking AOs as future-oriented and factual	**VN** (15): admire-31.2, allow-64, transfer-mesg-37.1.1, suspect-81, characterize-29.2 , neglect-75, want-32.1, defend-85, comprehend-87.2; **FN** (10): judgment, grant-permission, defend, experiencer-focus, judgment-communication, justifying, hit-or-miss, statement, reasoning, tolerating, grasp
1-- **11**-	120 (19%) – *beschreiben (describe), hören (hear), erinnern (remember)*	**wh/if -factual**: objective verbs marking AOs as factual	**VN** (55): discover-84, say-37.7, see-30.1, comprehend-87.2, rely-70, seem-109, consider-29.9, transfer-mesg-37.1.1, estimate-34.2, inquire-37.1.2; **FN** (23): perception-experience, statement, cogitation, grasp
110 11-	48 (8%) – *festlegen (determine), abschätzen (assess), lehren (teach)*	**future-oriented wh/if-factual**: objective verbs marking AOs as future-oriented and factual	**VN** (28): estimate-34.2, rely-70, indicate-78, transfer-mesg-37.1.1, correspond-36.1, conjecture-29.5, discover-84, say-37.7; **FN** (16): predicting, education-teaching, assessing, reliance, reasoning
111 0--	66 (10%) – *vorwerfen (accuse), bestreiten (deny), fürchten (fear)*	**non-factual**: verbs marking AOs as not resolvable re. their factuality	**VN** (28): conjecture-29.5, wish-62, complain-37.8, admire-31.2; **FN** (13): statement, reveal-secret, experiencer-focus, certainty

Table 2: The 632 verbs in 8 semantic classes (5 verbs show idiosyncratic behavior). Signature substrings in bold correspond to meaning components, which (along with tentative class names) are based on Sec. 3.2. The cross-lingual semantic characterization shows aligned VerbNet (VN) classes covering 265 (42%) verbs and aligned FrameNet (FN) frames covering 126 (20%) verbs, see Sec. 4.1.[6]

nature, these slots are left underspecified. Merging the signatures in this way yields 8 partially underspecified signatures which correspond to the final semantic classes. This procedure covers more than 99% of the 637 verbs under investigation: only 5 verbs showed idiosyncratic syntactic behavior, 4 of those are verbs that can take an AO as subject (e.g., *bedeuten (mean)*). As a consequence of the automatic part of this procedure, every verb is assigned to exactly one class – a simplification which we plan to resolve as part of future work.

Table 2 provides an overview and a characterization of these classes, also showing the final signatures and their substrings which correspond to the meaning components. The non-factual class is derived from the wh-factual class: the only difference is the inability to take a *wh*-clause (e.g. ⋆ *He hopes, <u>when</u> he will succeed.*).

While the descriptions of the meaning components and the class names are inspired from research in linguistics (typically a very deep analysis of only few verbs), transferring them to our verb resource – which is of much larger scale – inevitably leads to outlier verbs in the classes, e.g., verbs that do not strictly match the class label. Examples include verbs such as *überlegen (consider)* in the *wh/if-factual* class (not covering the future-oriented meaning component) or *schaden (harm)* as *non-factual rather* than as *wh-factual*. For this reason, and also because of the assignment of highly polysemous verbs to only one class, the definitions of meaning components and the class names should rather be considered as loose, providing a first tentative semantic characterization of the modality classes.

In sum, this section presented an inventory

of modal meaning components that we primarily synthesized from research in linguistics. The classification work is strictly grounded in syntactic properties of the verbs and was not targeted a priori at modal meanings.

4 Evaluation

4.1 Linking to Semantic Classes

Our first set of experiments aims at refining the initial semantic characterization of the classes by linking them to independently constructed semantic classifications *at the word sense level*. Specifically, we consider three different semantic classifications from computational lexicons, which have been created by linguistic experts: (i) the so-called semantic fields in GermaNet, grouping verb senses into 15 coarse classes, such as *perception, emotion*, (ii) the verb classes given in VerbNet, and (iii) the Frame-semantic frames in FrameNet. As the GermaNet and FrameNet classes are based on different lexicographic and linguistic theories, we expect an additional semantic characterization from the linking. The VerbNet classes, which also follow Levin's hypotheses, however, are used to investigate if the syntax-semantics correspondence is maintained across the related languages German and English.

For this linking experiment, we used the UBY framework (Gurevych et al., 2012)[10], containing standardized versions of the above lexicons, as well as a linking between VerbNet and FrameNet on the word sense level.

Approach In order to link our classes to verb senses in GermaNet and VerbNet, we developed an automatic linking method based on subcat frame similarity. Recognizing subcat frame similarity requires a common standardized format for the otherwise incomparable frames. UBY provides such a standardized format which has been presented in detail by Eckle-Kohler and Gurevych (2012). It represents subcat frames uniformly across German and English, and at a fine-grained level of individual syntactic arguments. Our linking approach is based on the following hypothesis: Two verb senses with equivalent lemmas are equivalent, if they have similar subcat frames.[11] Our method interprets the pairs of verb and subcat frame listed in our classification[12] as senses. While we do not claim that this hypothesis is sufficient in general, i.e., for all verb senses, we found that it is valid for the subset of senses belonging to the class of AO-selecting verbs.

The cross-lingual linking of our classes to VerbNet senses requires an additional translation step, which we describe first.

Manual Translation While UBY also provides translations between German and English verb senses, e.g., as part of the Interlingual Index from EuroWordnet (ILI), we found that many of the translations were not present in our target lexicon VerbNet. Therefore, the main author of this paper, a native speaker of German with a good proficiency in English, translated the AO-compatible verbs (i.e., word senses) manually using Linguee[13] and dictionaries. This took about 7 hours.

For 23 German verbs, we could not find any equivalent lexicalized translation, because these verbs express very fine-grained semantic nuances. For example, we did not find an equivalent English verb for a few verbs in the aspectual class but only a translation consisting of an adjective in combination with *to be*. Examples include *be easy (leichtfallen)*, *be willing (sich bereitfinden)*, *be capable (vermögen)*, which have German equivalents that are lexicalized as verbs. As a result, we arrived at translations for 614 out of 637 German verbs. These 614 German verbs are translated to 413 English verbs, indicating that the English translation has a more general meaning in many cases.

Automatic Verb Sense Linking Our algorithm links a German verb sense (or its English translation) with a GermaNet (or VerbNet) sense, if the subcat frames of both verb senses have the same number of arguments and if the arguments have certain features in common.[14] For example, to create a link to GermaNet, features such as the complementizer of clausal arguments and the case of noun phrase arguments have to agree. In a similar way, the linking to VerbNet is based on a comparison of German subcat frames and English subcat

[10] `http://www.ukp.tu-darmstadt.de/uby/`

[11] This approach is applicable for GermaNet, because GermaNet contains fine-grained syntactic subcat frames.

[12] We consider only verb senses that are compatible with AOs, as indicated by subcat frames with clausal or non-finite arguments.

[13] Linguee (`http://www.linguee.de/`) is a translation tool combining an editorial dictionary and a search engine processing bilingual texts. In particular, it provides a large variety of contextual translation examples.

[14] We do not link the subcat frames, but we do compare them across the related languages German and English to determine their similarity in the context of linking.

frames – which are represented uniformly across German and English. In Section A.2, we provide more details about the algorithm.

Results According to a manual evaluation of a random sample of 200 sense pairs, the automatic verb sense linking yielded an accuracy of 89.95% for the linking to GermaNet, and 87.54% for the linking to VerbNet (κ agreement on the sample annotated by two annotators was 0.7 and 0.8, respectively). The main types of errors in the linking to GermaNet and VerbNet are due to specific syntactic features of the subcat frames which diverge and are not considered in the automatic linking. The differences regarding these specific features are due to cross-lingual differences (VerbNet, e.g., verb phrase arguments with ing-form) and diverging linguistic analyses of particular constructions (GermaNet, e.g., constructions with *es (it)*), see also Eckle-Kohler and Gurevych (2012).

By linking the verbs in our classification to semantic classes in GermaNet, VerbNet and FrameNet, we obtain a three-way semantic characterization of our classes. The linking to the GermaNet semantic fields covers 270 (43%) of the source verbs. Of these, 219 (81%) are linked to the three semantic fields *cognition*, *communication* and *social*. Fewer verbs (32 (12%)) are linked to the semantic fields *emotion, perception, change*. Semantic fields not among the target classes are *consumption, competition, contact, body* and *weather*.

Table 2 summarizes the linking to VerbNet and FrameNet and shows how many verbs from each source class could be linked to any of the classes in VerbNet or FrameNet.[15] As the class distribution of the verb subsets covered by our linking-based evaluation is similar as for the original classes, we consider our evaluation as valid, although less than 50% of all verbs could be evaluated this way.

The target classes in VerbNet and FrameNet reveal meaning components that are on the one hand unique for individual classes, and on the other hand shared across several German classes.

The *future-oriented* class contains object control verbs (e.g., force-59, forbid-67 in VerbNet, and *request, preventing* in FrameNet). The *wh/if-factual* class is unique regarding the *cognition and perception* verbs (e.g., discover-84, see-30.1-1, and *perception-experience*). The *future-directed wh/if-factual* class also contains *objective assessment* verbs, as shown by the estimate-34.2 class. The verbs in the two *wh-factual* classes share meaning components as well, as shown by the *opinion verb* classes admire-31.2 and defend-85 in VerbNet or *judgment, tolerating* in FrameNet.

While there are also other VerbNet and FrameNet classes shared across several classes, they turned out to be very general and underspecified regarding their meaning, thus not contributing to a more fine-grained semantic characterization. For example, the conjecture-29.5 class assembles quite diverse conjecture verbs, e.g. verbs expressing opinion (*feel, trust*) and factuality (*observe, discover*). A similar observation holds for the *statement* frame in FrameNet.

Verb class	Wiki	Web	News	News Eng.
all	25.85	50.58	33.91	25.31
aspectual	0.90	0.80	1.44	1.96
future-oriented	9.45	23.04	13.65	12.58
interrogative	0.01	0.05	0.05	0.65
wh-factual	4.26	17.89	4.99	3.48
fo. wh-factual	0.29	0.28	0.85	1.14
wh/if -factual	3.02	2.54	3.53	5.20
fo. wh/if-factual	2.36	1.77	3.14	5.75
non-factual	4.29	3.36	4.84	3.57

Table 3: Percentage of classes in corpora: German Wikipedia (Wiki), SDeWaC (Web), Tiger (News); English Reuters corpus (News Eng.).

4.2 Analysis of Frequency and Polysemy

In order to assess the usefulness of the verb resource for NLP tasks, we determined the lemma frequency of all verbs in the 8 classes in SDeWaC (Faaß and Eckart, 2013), a cleaned version of the German DeWaC corpus (Baroni and Kilgarriff, 2006). A ranking of the verbs according to their lemma frequency showed that 89% of the verbs occur more than 50 times in SDeWaC.[16]

We also analyzed the frequency distribution of the 8 verb classes in two other German corpora belonging to different genres, and also for English, see Table 3:[17] encyclopedic text (the German Wikipedia[18]), German newspaper text (the Tiger corpus (Brants et al., 2004)), and the English

[15] Based on the percentage of source class members linked to any of the target classes, we only display target classes with an overlap of at least 1.8% due to space constraints.

[16] In the verb resource we provide for download, we included this frequency information in order to enable frequency-based filtering.

[17] Details of the computation of the verb lemma frequency lists are given in the appendix A.1.

[18] `www.wikipedia.de`, dump of 2009-06-18

Reuters-21578 corpus.[19] Table 3 shows that the large verb classes constitute a substantial proportion of verb occurrences across different genres. This suggests that the verb classes might be useful features for various text classification tasks.

We performed a further analysis of the polysemy of the German and English verbs in our classes relative to several fine and coarse word sense inventories. Regarding GermaNet, there are 2.28 senses per verb (1.53 for all GermaNet verbs), whereas WordNet lists 5.11 senses per verb (2.17 for all WordNet verbs). In VerbNet, we find 1.74 senses per verb (1.42 for all VerbNet verbs), and in FrameNet 1.96 (1.52 for all FrameNet verbs). This analysis shows that the task of automatic sense linking is particularly hard for the category of AO-selecting verbs we consider. Whether the polysemy is an issue for any application where the verb classes are used as features is not a priori clear and depends on the task at hand.

4.3 Textual Entailment Experiment

For an extrinsic evaluation, we investigated the usefulness of the German *and* the English verb classes as features in recognizing textual entailment (RTE). In RTE, the task is to determine whether for a pair of text fragments – the text T and the hypothesis H – the meaning of H is entailed by T (Dagan et al., 2006); for non-entailing pairs, sometimes a further category "unknown" is used as a label.

We employed a simple classification-based approach to RTE and trained and evaluated a Naive Bayes classifier on the test sets of three RTE benchmarks, using 10-fold cross validation: the English RTE-3 data (Giampiccolo et al., 2009) and their German translation[20] (the development sets and the test sets each consist of 800 pairs), and an expanded version of the English RTE-3 data from the Sagan Textual Entailment Test Suite (Castillo, 2010) consisting of 2974 pairs. While the German dataset provides a two-way classification of the T-H pairs, the two English datasets provide a three-way classification, also using the "unknown" label. We used the DKPro TC framework (Daxenberger et al., 2014) for classification and applied POS tagging and lemmatization as preprocessing.

As a baseline feature, we use the word overlap measure between two T-H pairs (no stopword filtering, no lemmatization, no normalization of overlap score), which is quite competitive on the RTE-3 data, because this dataset shows a high difference in word overlap between positive (entailment) and negative (no entailment) pairs (Bentivogli et al., 2009).

An analysis of the development set of the German RTE-3 data showed that 62% of the pairs contain at least one occurrence of any of the verbs from the classification in either T or H. However, T and H fragments display no statistically significant differences[21] regarding the occurrences of any of the verb classes.

A detailed analysis revealed that pairs without entailment are often characterized by a mismatch between T and H regarding the presence of factuality markers. For example, the presence of verbs indicating uncertainty (all classes apart from wh-factual and wh/if-factual) in T and an absence of such verbs in H might indicate non-entailment as in the following not entailing pair from the English RTE3 development set where "long" signals non-factuality, but "researching" signals factuality:

> T: *The BBC's Americas editor Will Grant says many Mexicans are tired of conflict and long for a return to normality.*
> H: *Will Grant is researching a conflict with Mexicans.*

Thus, an insufficient overlap of modality markers in T and H might actually indicate non-entailment, but lead to an incorrect classification as entailment when considering only word overlap.

Accordingly, we implemented a factuality-mismatch feature both for German and for English, based on our new German and English classes. This feature is similar to the word overlap feature but with lemmatization and normalization of overlap score. Verb class counts are based on verb lemma counts of the member verbs; for English verbs that are members of more than one class, we included all verb classes in our factuality-mismatch feature.[22] Table 4 shows the results. While the differences

[19] Reuters-21578, Distribution 1.0, see http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html.

[20] http://www.dfki.de/~neumann/resources/RTE3_DE_V1.2_2013-12-02.zip

[21] All significance scores in this paper are based on Fisher's exact test at significance level $p<0.05$.

[22] In the German part, every verb is assigned to one class, while the translation to English resulted in 22% of the English verbs being members in more than one class. However, only 11% of the multiple class assignments involve a combination of factual and uncertainty classes.

for RTE-3 DE and RTE-3 EN are not statistically significant, the factuality-mismatch feature yielded a small but significant improvement on the expanded RTE-3 EN dataset. This is due to the different nature of the expanded RTE dataset, which was created using a paraphrasing technique. As a result, the number of occurrences of verbs from our classes increased, and the factuality-mismatch became a discriminative feature for distinguishing between CONTRADICTION and UNKNOWN/ENTAILMENT.

Considering the fact that we employed only simple overlap features that do not rely on dependency parsing and did not perform any word sense disambiguation, these results suggest that the verb classes might be promising features for RTE, both for German *and* English. As factuality can be expressed by a variety of further linguistic means, including modal verbs, negation, tense and certain adverbs, investigating the combination of our verb classes with other modality signals might be especially promising as part of future work.

	RTE-3 DE	**RTE-3 EN**	**RTE-3 EN exp.**
WO	59.87	54.75	54.98
WO+FM	59.25	54.62	**58.81**

Table 4: Accuracy of a Naive Bayes classifier (10-fold cross validation on the test sets) with word overlap (WO) and additional factuality-mismatch (WO+FM) features.

5 Results and Discussion

Our construction of semantic classes from the syntactic behavior of AO-selecting verbs results in an inventory of modal meanings that emerged from a large lexical resource. The main result of the linking based evaluation is a detailed semantic characterization of the inferred classes – a prerequisite for using them in NLP tasks in an informed way. The semantic classes seem to be particular suited for tasks related to opinion analysis, textual inference, or argumentation mining. In this context, the relationship between our large resource of lexical verbs and the closed class of modal verbs might be an interesting question for future research.

Most of all, the linking to GermaNet and FrameNet shows that it is indeed possible to narrow down meaning components for Levin classes. Moreover, the results of the linking to VerbNet also provide support for Levin's hypothesis that the correspondences between verb syntax and meaning described for English largely apply to the related language German as well (Levin, 2015b).

The English version of the semantic classes which we created by means of translation has the same semantic properties as the German classes. However, the syntactic properties of the English classes are not fully specified, because English has additional kinds of non-finite arguments, such as ing-forms or bare infinitives. Therefore, it might be interesting to address this question in the future and to build a similar semantic classification for English from scratch, in particular in the context of extracting modality classes from corpora. This would require an adaptation of the syntactic signatures, considering the various kinds of non-finite arguments particular to English. Based on large subcategorization lexicons available for English (e.g. COMLEX (Grishman et al., 1994) or VerbNet), it should be feasible to derive such signatures and to construct a mapping of signatures to modality aspects in a similar way as for German.

The question whether the syntactic signatures can be recovered in large corpora is particularly interesting, because this would allow extending the existing classes and to also acquire AO-selecting adjectives and nouns. We plan to investigate this question as part of future work.

6 Conclusion

We inferred semantic classes from a large syntactic classification of German AO-selecting verbs based on findings from formal semantics about correspondences between verb syntax and meaning. Our thorough evaluation and analysis yields detailed insights into the semantic characteristics of the inferred classes, and we hope that this allows an informed use of the resulting resource in various semantic NLP tasks.

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806 and by the German Research Foundation under grant No. GU 798/17-1 and No. GRK 1994/1. We thank the anonymous reviewers for their valuable comments. Additional thanks go to Anette Frank, Iryna Gurevych and Ani Nenkova for their helpful feedback on earlier versions of this work.

References

Nicholas Asher. 1993. *Reference to Abstract Objects in Discourse*. Studies in Linguistics and Philosophy (Book 50). Springer.

Collin F. Baker and Josef Ruppenhofer. 2002. FrameNet's Frames vs. Levin's Verb Classes. In *Proceedings of 28th Annual Meeting of the Berkeley Linguistics Society*, pages 27–38, Berkeley, CA, USA.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (COLING-ACL)*, pages 86–90, Montreal, Canada.

Marco Baroni and Adam Kilgarriff. 2006. Large Linguistically-Processed Web Corpora for Multiple Languages. In *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 87–90, Trento, Italy.

Luisa Bentivogli, Bernardo Magnini, Ido Dagan, Hoa Trang Dang, and Danilo Giampiccolo. 2009. The Fifth Pascal Recognizing Textual Entailment Challenge. In *Proceedings of the Text Analysis Conference (TAC)*, pages 14–24, Gaithersburg, Maryland, USA.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620.

Julio J. Castillo. 2010. Using Machine Translation Systems to Expand a Corpus in Textual Entailment. In Hrafn Loftsson, Eirkur Rgnvaldsson, and Sigrn Helgadttir, editors, *Advances in Natural Language Processing*, volume 6233 of *Lecture Notes in Computer Science*, pages 97–102. Springer, Berlin Heidelberg.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL Recognising Textual Entailment Challenge. In Joaquin Quionero-Candela, Ido Dagan, Bernardo Magnini, and Florence dAlch Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer Berlin Heidelberg.

Johannes Daxenberger, Oliver Ferschke, Iryna Gurevych, and Torsten Zesch. 2014. Dkpro tc: A java-based framework for supervised learning experiments on textual data. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 61–66, Baltimore, MD, USA.

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts. 2012. Did It Happen? The Pragmatic Complexity of Veridicality Assessment. *Computational Linguistics*, 38(2):301–333, June.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A Broad-Coverage Collection of Portable NLP Components for Building Shareable Analysis Pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT (OIAF4HLT) at COLING 2014*, pages 1–11, Dublin, Ireland.

Judith Eckle-Kohler and Iryna Gurevych. 2012. Subcat-LMF: Fleshing Out a Standardized Format for Subcategorization Frame Interoperability. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 550–560, Avignon, France.

Judith Eckle-Kohler, Michael Kohler, and Jens Mehnert. 2008. Automatic recognition of german news focusing on future-directed beliefs and intentions. *Computer Speech and Language*, 22(4):394–414, October.

Judith Eckle-Kohler. 1999. *Linguistisches Wissen zur automatischen Lexikon-Akquisition aus deutschen Textcorpora*. Logos-Verlag, Berlin, Germany. PhD Thesis, Universität Stuttgart, Germany.

Gertrud Faaß and Kerstin Eckart. 2013. SdeWaC – A Corpus of Parsable Sentences from the Web. In Iryna Gurevych, Chris Biemann, and Torsten Zesch, editors, *Language Processing and Knowledge in the Web: Proceedings of the 25th Conference of the German Society for Computational Linguistics (GSCL 2013), Darmstadt, Germany, September 25-27, 2013.*, pages 61–68. Springer, Berlin, Heidelberg.

Arne Fitschen. 2004. Ein Computerlinguistisches Lexikon als komplexes System. PhD Thesis, Universität Stuttgart, Germany.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2009. The Third PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the Workshop on Textual Entailment and Paraphrasing at ACL 2009*, pages 1–9, Prague, Czech Republic.

Jonathan Ginzburg. 1996. Interrogatives: Questions, Facts, and Dialogue. In Shalom Lappin, editor, *The Handbook of Contemporary Semantic Theory*, pages 385–422. Blackwell, Oxford, UK.

Ralph Grishman, Catherine Macleod, and Adam Meyers. 1994. Comlex Syntax: Building a Computational Lexicon. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, pages 268–272, Kyoto, Japan.

Iryna Gurevych, Judith Eckle-Kohler, Silvana Hartmann, Michael Matuschek, Christian M. Meyer, and Christian Wirth. 2012. UBY - A Large-Scale Unified Lexical-Semantic Resource Based on LMF. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 580–590, Avignon, France.

Valentine Hacquard. 2011. Modality. In Claudia Maienborn, Klaus von Heusinger, and Paul Portner, editors, *Semantics: An International Handbook of Natural Language Meaning. HSK 33.2*, pages 1484–1515. Berlin: Mouton de Gruyter.

Joshua K. Hartshorne, Claire Bonial, and Martha Palmer. 2014. The VerbCorner Project: Findings from Phase 1 of Crowd-Sourcing a Semantic Decomposition of Verbs. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 397–402, Baltimore, MD, USA.

Lauri Karttunen. 1971. Implicative Verbs. *Language*, pages 340–358.

Lauri Karttunen. 2012. Simple and Phrasal Implicatives. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics*, pages 124–131, Montréal, Canada.

Paul Kiparsky and Carol Kiparsky, 1970. *Fact*. Mouton, The Hague.

Karin Kipper, Anna Korhonen, Neville Ryant, and Marthe Palmer. 2006. Extending VerbNet with Novel Verb Classes. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC)*, pages 1027–1032, Genoa, Italy.

Karin Kipper, Anna Korhonen, Neville Ryant, and Martha Palmer. 2008. A Large-scale Classification of English Verbs. *Language Resources and Evaluation*, 42:21–40.

Anna Korhonen and Ted Briscoe. 2004. Extended Lexical-Semantic Classification of English Verbs. In *Proceedings of the Workshop on Computational Lexical Semantics at HLT-NAACL 2004*, pages 38–45, Boston, Massachusetts, USA.

Ralf Krestel, Sabine Bergler, and Ren Witte. 2008. Minding the Source: Automatic Tagging of Reported Speech in Newspaper Articles. In Nicoletta Calzolari et al., editor, *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC)*, pages 2823–2828, Marrakech, Morocco.

Claudia Kunze and Lothar Lemnitzer. 2002. GermaNet – Representation, Visualization, Application. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC)*, pages 1485–1491, Las Palmas, Canary Islands, Spain.

Dave Kush. 2011. Mental Action and Event Structure in the Semantics of 'try'. In *Proceedings of the 21st Semantics and Linguistic Theory Conference*, pages 413–425, New Brunswick, New Jersey, USA.

Brenda Laca. 2013. Temporal Orientation and the Semantics of Attitude Verbs. In Karina Veronica Molsing and Ana Maria Tramunt Ibaños, editors, *Time and TAME in Language*, pages 158–180. Cambridge Scholars Publishing, Newcastle upon Tyne, UK.

Beth Levin. 1993. *English Verb Classes and Alternations*. The University of Chicago Press, Chicago, USA.

Beth Levin. 2015a. Semantics and Pragmatics of Argument Alternations. *Annual Review of Linguistics*, 1(1):63–83.

Beth Levin. 2015b. Verb Classes Within and Across Languages. In Andrej Malchukov and Bernard Comrie, editors, *Valency Classes in the Worlds Languages (Volume 2): Case Studies from Austronesia, the Pacific, the Americas, and Theoretical Outlook*, pages 1627–1670. Berlin, Boston: De Gruyter Mouton.

Amnon Lotan, Asher Stern, and Ido Dagan. 2013. TruthTeller: Annotating Predicate Truth. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 752–757, Atlanta, Georgia.

Paola Merlo and Suzanne Stevenson. 2001. Automatic Verb Classification Based on Statistical Distributions of Argument Structure. *Computational Linguistics*, 27(3):373–408, September.

Rowan Nairn, Cleo Condoravdi, and Lauri Karttunen. 2006. Computing Relative Polarity for Textual Inference. *Inference in Computational Semantics (ICoS-5)*, pages 20–21.

Malvina Nissim, Paola Pietrandrea, Andrea Sanso, and Caterina Mauri. 2013. Cross-Linguistic Annotation of Modality: a Data-Driven Hierarchical Model. In *Proceedings of the 9th Joint ISO - ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 7–14, Potsdam, Germany.

Vinodkumar Prabhakaran, Owen Rambow, and Mona Diab. 2010. Automatic Committed Belief Tagging. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1014–1022, Beijing, China.

Roser Saurí and James Pustejovsky. 2007. Determining Modality and Factuality for Text Entailment. In *Proceedings of the International Conference on Semantic Computing*, ICSC '07, pages 509–516, Washington, DC, USA. IEEE Computer Society.

Roser Saurí and James Pustejovsky. 2009. FactBank: a Corpus Annotated with Event Factuality. *Language Resources and Evaluation*, 43(3):227–268.

Roser Saurí and James Pustejovsky. 2012. Are You Sure That This Happened? Assessing the Factuality Degree of Events in Text. *Computational Linguistics*, 38(2):261–299, June.

Roser Saurí, Robert Knippen, Marc Verhagen, and James Pustejovsky. 2005. Evita: A Robust Event Recognizer for QA Systems. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 700–707, Vancouver, British Columbia, Canada.

Roser Saurí. 2008. A Factuality Profiler for Eventualities in Text. PhD Thesis, Brandeis University, Waltham, MA, USA.

Daniel Schnorbusch. 2004. Semantische Klassen aus syntaktischen Klassen? In Stefan Langer and Daniel Schnorbusch, editors, *Semantik im Lexikon*, pages 33–58. Gunter Narr Verlag, Tübingen.

Sabine Schulte im Walde. 2006. Experiments on the Automatic Induction of German Semantic Verb Classes. *Computational Linguistics*, 32(2):159–194, June.

Kerstin Schwabe and Robert Fittler. 2009. Semantic Characterizations of German Question-Embedding Predicates. In Peter Bosch, David Gabelaia, and Jérôme Lang, editors, *Logic, Language, and Computation*, volume 5422 of *Lecture Notes in Computer Science*, pages 229–241. Springer Berlin Heidelberg.

György Szarvas, Veronika Vincze, Richàrd Farkas, György Mra, and Iryna Gurevych. 2012. Cross-Genre and Cross-Domain Detection of Semantic Uncertainty. *Computational Linguistics*, 38(2):335–367, June.

A Supplemental Material

A.1 Verb Lemma Frequency List

In order to count the occurrences of verb lemmas in the German corpus SDeWaC, we used a reader and pre-processing components (i.e., the LanguageTool segmenter and the TreeTagger for POS tagging and lemmatization) from the DKPro Core collection (Eckart de Castilho and Gurevych, 2014). From DKPro Core, we also used a component that detects separated particles of German particle verbs and replaces the lemma of the verb base form annotated by the TreeTagger by the true lemma of the particle verb. Our verb lemma counting pipeline is available at `github.com/UKPLab/acl2016-modality-verbclasses`.

Sense linking based on subcategorization frames

get lexical entry le_s of source verb v_s
get equivalent verb v_t in target lexicon
get lexical entry le_t of target verb v_t
forall frame f_i in le_s
  get listOfArguments l_i of f_i
  forall frame f_j in le_t
    get sense s_j of frame f_j
    get listOfArguments l_j of f_j
    if $size(l_i) = size(l_j)$
    AND $features(l_i) = features(l_j)$
      link (v_s, f_i) and s_j
    end if
  end for
end for

Table 5: Algorithm for verb sense linking.

A.2 Verb Sense Linking

For the linking-based evaluation, we used UBY (version 0.7.0) versions of the following three resources: the German wordnet GermaNet (version 9.0), the English lexicons VerbNet (version 3.2) and FrameNet (version 1.5).

The algorithm for cross-lingual verb sense linking is given in pseudo-code in Table 5. The implementation is available at `github.com/UKPLab/acl2016-modality-verbclasses`.

Tree-to-Sequence Attentional Neural Machine Translation

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka
The University of Tokyo, 3-7-1 Hongo, Bunkyo-ku, Tokyo, Japan
{eriguchi, hassy, tsuruoka}@logos.t.u-tokyo.ac.jp

Abstract

Most of the existing Neural Machine Translation (NMT) models focus on the conversion of sequential data and do not directly use syntactic information. We propose a novel end-to-end syntactic NMT model, extending a sequence-to-sequence model with the source-side phrase structure. Our model has an attention mechanism that enables the decoder to generate a translated word while softly aligning it with phrases as well as words of the source sentence. Experimental results on the WAT'15 English-to-Japanese dataset demonstrate that our proposed model considerably outperforms sequence-to-sequence attentional NMT models and compares favorably with the state-of-the-art tree-to-string SMT system.

1 Introduction

Machine Translation (MT) has traditionally been one of the most complex language processing problems, but recent advances of Neural Machine Translation (NMT) make it possible to perform translation using a simple end-to-end architecture. In the Encoder-Decoder model (Cho et al., 2014b; Sutskever et al., 2014), a Recurrent Neural Network (RNN) called the *encoder* reads the whole sequence of source words to produce a fixed-length vector, and then another RNN called the *decoder* generates the target words from the vector. The Encoder-Decoder model has been extended with an *attention* mechanism (Bahdanau et al., 2015; Luong et al., 2015a), which allows the model to jointly learn the soft alignment between the source language and the target language. NMT models have achieved state-of-the-art results in English-to-French and English-to-German translation tasks (Luong et al., 2015b; Luong et al., 2015a). However, it is yet to be seen whether NMT is competitive with traditional Statistical Machine Translation (SMT) approaches in translation tasks for structurally distant language pairs such as English-to-Japanese.

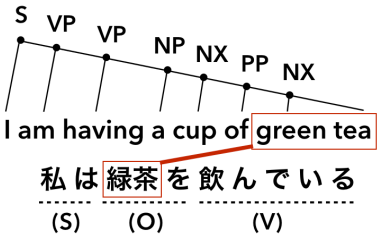


Figure 1: Alignment between an English phrase and a Japanese word.

Figure 1 shows a pair of parallel sentences in English and Japanese. English and Japanese are linguistically distant in many respects; they have different syntactic constructions, and words and phrases are defined in different lexical units. In this example, the Japanese word "緑茶" is aligned with the English words "green" and "tea", and the English word sequence "a cup of" is aligned with a special symbol "*null*", which is not explicitly translated into any Japanese words. One way to solve this mismatch problem is to consider the phrase structure of the English sentence and align the phrase "a cup of green tea" with "緑茶". In SMT, it is known that incorporating syntactic constituents of the source language into the models improves word alignment (Yamada and Knight, 2001) and translation accuracy (Liu et al., 2006; Neubig and Duh, 2014). However, the existing NMT models do not allow us to perform this kind of alignment.

In this paper, we propose a novel attentional NMT model to take advantage of syntactic infor-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 823–833,
Berlin, Germany, August 7-12, 2016.

mation. Following the phrase structure of a source sentence, we encode the sentence recursively in a bottom-up fashion to produce a vector representation of the sentence and decode it while aligning the input phrases and words with the output. Our experimental results on the WAT'15 English-to-Japanese translation task show that our proposed model achieves state-of-the-art translation accuracy.

2 Neural Machine Translation

2.1 Encoder-Decoder Model

NMT is an end-to-end approach to data-driven machine translation (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). In other words, the NMT models directly estimate the conditional probability $p(\boldsymbol{y}|\boldsymbol{x})$ given a large collection of source and target sentence pairs $(\boldsymbol{x}, \boldsymbol{y})$. An NMT model consists of an encoder process and a decoder process, and hence they are often called *Encoder-Decoder* models. In the Encoder-Decoder models, a sentence is treated as a sequence of words. In the encoder process, the encoder embeds each of the source words $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ into a d-dimensional vector space. The decoder then outputs a word sequence $\boldsymbol{y} = (y_1, y_2, \cdots, y_m)$ in the target language given the information on the source sentence provided by the encoder. Here, n and m are the lengths of the source and target sentences, respectively. RNNs allow one to effectively embed sequential data into the vector space.

In the RNN encoder, the i-th hidden unit $\boldsymbol{h}_i \in \mathbb{R}^{d\times 1}$ is calculated given the i-th input x_i and the previous hidden unit $\boldsymbol{h}_{i-1} \in \mathbb{R}^{d\times 1}$,

$$\boldsymbol{h}_i \;=\; f_{enc}(x_i, \boldsymbol{h}_{i-1}), \tag{1}$$

where f_{enc} is a non-linear function, and the initial hidden unit $\boldsymbol{h}_0$ is usually set to zeros. The encoding function f_{enc} is recursively applied until the n-th hidden unit $\boldsymbol{h}_n$ is obtained. The RNN Encoder-Decoder models assume that $\boldsymbol{h}_n$ represents a vector of the meaning of the input sequence up to the n-th word.

After encoding the whole input sentence into the vector space, we decode it in a similar way. The initial decoder unit $\boldsymbol{s}_1$ is initialized with the input sentence vector ($\boldsymbol{s}_1 = \boldsymbol{h}_n$). Given the previous target word and the j-th hidden unit of the decoder, the conditional probability that the j-th target word is generated is calculated as follows:

$$p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) \;=\; g(\boldsymbol{s}_j), \tag{2}$$

where g is a non-linear function. The j-th hidden unit of the decoder is calculated by using another non-linear function f_{dec} as follows:

$$\boldsymbol{s}_j \;=\; f_{dec}(y_{j-1}, \boldsymbol{s}_{j-1}). \tag{3}$$

We employ Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) in place of vanilla RNN units. The t-th LSTM unit consists of several *gates* and two different types of states: a hidden unit $\boldsymbol{h}_t \in \mathbb{R}^{d\times 1}$ and a memory cell $\boldsymbol{c}_t \in \mathbb{R}^{d\times 1}$,

$$\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}^{(i)}\boldsymbol{x}_t + \boldsymbol{U}^{(i)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}^{(f)}\boldsymbol{x}_t + \boldsymbol{U}^{(f)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(f)}),\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{W}^{(o)}\boldsymbol{x}_t + \boldsymbol{U}^{(o)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_t &= \tanh(\boldsymbol{W}^{(\tilde{c})}\boldsymbol{x}_t + \boldsymbol{U}^{(\tilde{c})}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1},\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t),
\end{aligned} \tag{4}$$

where each of $\boldsymbol{i}_t$, $\boldsymbol{f}_t$, $\boldsymbol{o}_t$ and $\tilde{\boldsymbol{c}}_t \in \mathbb{R}^{d\times 1}$ denotes an input gate, a forget gate, an output gate, and a state for updating the memory cell, respectively. $\boldsymbol{W}^{(\cdot)} \in \mathbb{R}^{d\times d}$ and $\boldsymbol{U}^{(\cdot)} \in \mathbb{R}^{d\times d}$ are weight matrices, $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^{d\times 1}$ is a bias vector, and $\boldsymbol{x}_t \in \mathbb{R}^{d\times 1}$ is the word embedding of the t-th input word. $\sigma(\cdot)$ is the logistic function, and the operator $\odot$ denotes element-wise multiplication between vectors.

2.2 Attentional Encoder-Decoder Model

The NMT models with an attention mechanism (Bahdanau et al., 2015; Luong et al., 2015a) have been proposed to softly align each decoder state with the encoder states. The attention mechanism allows the NMT models to explicitly quantify how much each encoder state contributes to the word prediction at each time step.

In the attentional NMT model in Luong et al. (2015a), at the j-th step of the decoder process, the attention score $\boldsymbol{\alpha}_j(i)$ between the i-th source hidden unit $\boldsymbol{h}_i$ and the j-th target hidden unit $\boldsymbol{s}_j$ is calculated as follows:

$$\boldsymbol{\alpha}_j(i) \;=\; \frac{\exp(\boldsymbol{h}_i \cdot \boldsymbol{s}_j)}{\sum_{k=1}^{n}\exp(\boldsymbol{h}_k \cdot \boldsymbol{s}_j)}, \tag{5}$$

where $\boldsymbol{h}_i \cdot \boldsymbol{s}_j$ is the inner product of $\boldsymbol{h}_i$ and $\boldsymbol{s}_j$, which is used to directly calculate the similarity score between $\boldsymbol{h}_i$ and $\boldsymbol{s}_j$. The j-th context vector

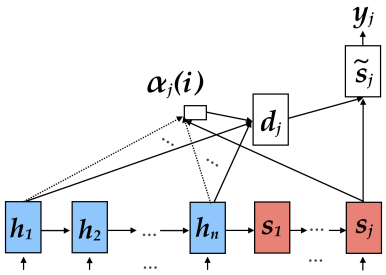


Figure 2: Attentional Encoder-Decoder model.

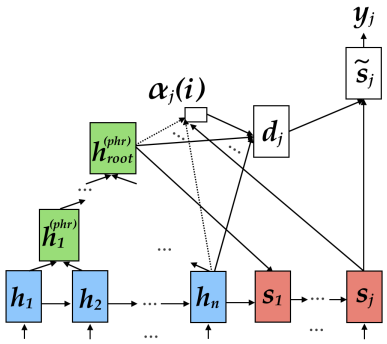


Figure 3: Proposed model: Tree-to-sequence attentional NMT model.

$\boldsymbol{d}_j$ is calculated as the summation vector weighted by $\boldsymbol{\alpha}_j(i)$:

$$\boldsymbol{d}_j = \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i. \quad (6)$$

To incorporate the attention mechanism into the decoding process, the context vector is used for the the j-th word prediction by putting an additional hidden layer $\tilde{\boldsymbol{s}}_j$:

$$\tilde{\boldsymbol{s}}_j = \tanh(\boldsymbol{W}_d[\boldsymbol{s}_j;\ \boldsymbol{d}_j] + \boldsymbol{b}_d), \quad (7)$$

where $[\boldsymbol{s}_j; \boldsymbol{d}_j] \in \mathbb{R}^{2d\times 1}$ is the concatenation of $\boldsymbol{s}_j$ and $\boldsymbol{d}_j$, and $\boldsymbol{W}_d \in \mathbb{R}^{d\times 2d}$ and $\boldsymbol{b}_d \in \mathbb{R}^{d\times 1}$ are a weight matrix and a bias vector, respectively. The model predicts the j-th word by using the softmax function:

$$p(y_j|\boldsymbol{y}_{<j}, \boldsymbol{x}) = \text{softmax}(\boldsymbol{W}_s\tilde{\boldsymbol{s}}_j + \boldsymbol{b}_s), \quad (8)$$

where $\boldsymbol{W}_s \in \mathbb{R}^{|V|\times d}$ and $\boldsymbol{b}_s \in \mathbb{R}^{|V|\times 1}$ are a weight matrix and a bias vector, respectively. $|V|$ stands for the size of the vocabulary of the target language. Figure 2 shows an example of the NMT model with the attention mechanism.

2.3 Objective Function of NMT Models

The objective function to train the NMT models is the sum of the log-likelihoods of the translation pairs in the training data:

$$J(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \log p(\boldsymbol{y}|\boldsymbol{x}), \quad (9)$$

where $\mathcal{D}$ denotes a set of parallel sentence pairs. The model parameters $\boldsymbol{\theta}$ are learned through Stochastic Gradient Descent (SGD).

3 Attentional Tree-to-Sequence Model

3.1 Tree-based Encoder + Sequential Encoder

The exsiting NMT models treat a sentence as a sequence of words and neglect the structure of a sentence inherent in language. We propose a novel tree-based encoder in order to explicitly take the syntactic structure into consideration in the NMT model. We focus on the phrase structure of a sentence and construct a sentence vector from phrase vectors in a bottom-up fashion. The sentence vector in the tree-based encoder is therefore composed of the structural information rather than the sequential data. Figure 3 shows our proposed model, which we call a *tree-to-sequence attentional NMT model*.

In Head-driven Phrase Structure Grammar (HPSG) (Sag et al., 2003), a sentence is composed of multiple phrase units and represented as a binary tree as shown in Figure 1. Following the structure of the sentence, we construct a tree-based encoder on top of the standard sequential encoder. The k-th parent hidden unit $\boldsymbol{h}_k^{(phr)}$ for the k-th phrase is calculated using the left and right child hidden units $\boldsymbol{h}_k^l$ and $\boldsymbol{h}_k^r$ as follows:

$$\boldsymbol{h}_k^{(phr)} = f_{tree}(\boldsymbol{h}_k^l, \boldsymbol{h}_k^r), \quad (10)$$

where f_{tree} is a non-linear function.

We construct a tree-based encoder with LSTM units, where each node in the binary tree is represented with an LSTM unit. When initializing the leaf units of the tree-based encoder, we employ the sequential LSTM units described in Section 2.1. Each non-leaf node is also represented with an LSTM unit, and we employ Tree-LSTM (Tai et al., 2015) to calculate the LSTM unit of the parent node which has two child LSTM units. The hidden unit $\boldsymbol{h}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ and the memory cell $\boldsymbol{c}_k^{(phr)} \in \mathbb{R}^{d\times 1}$ for the k-th parent node are calcu-

lated as follows:

$$\begin{aligned}
\boldsymbol{i}_k &= \sigma(\boldsymbol{U}_l^{(i)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(i)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(i)}),\\
\boldsymbol{f}_k^l &= \sigma(\boldsymbol{U}_l^{(f_l)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_l)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_l)}),\\
\boldsymbol{f}_k^r &= \sigma(\boldsymbol{U}_l^{(f_r)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(f_r)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(f_r)}),\\
\boldsymbol{o}_k &= \sigma(\boldsymbol{U}_l^{(o)}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(o)}\boldsymbol{h}_k^r + \boldsymbol{b}^{(o)}),\\
\tilde{\boldsymbol{c}}_k &= \tanh(\boldsymbol{U}_l^{(\tilde{c})}\boldsymbol{h}_k^l + \boldsymbol{U}_r^{(\tilde{c})}\boldsymbol{h}_k^r + \boldsymbol{b}^{(\tilde{c})}),\\
\boldsymbol{c}_k^{(phr)} &= \boldsymbol{i}_k \odot \tilde{\boldsymbol{c}}_k + \boldsymbol{f}_k^l \odot \boldsymbol{c}_k^l + \boldsymbol{f}_k^r \odot \boldsymbol{c}_k^r,\\
\boldsymbol{h}_k^{(phr)} &= \boldsymbol{o}_k \odot \tanh(\boldsymbol{c}_k^{(phr)}), \qquad (11)
\end{aligned}$$

where $\boldsymbol{i}_k$, $\boldsymbol{f}_k^l$, $\boldsymbol{f}_k^r$, $\boldsymbol{o}_j$, $\tilde{\boldsymbol{c}}_j \in \mathbb{R}^{d\times 1}$ are an input gate, the forget gates for left and right child units, an output gate, and a state for updating the memory cell, respectively. $\boldsymbol{c}_k^l$ and $\boldsymbol{c}_k^r$ are the memory cells for the left and right child units, respectively. $\boldsymbol{U}^{(\cdot)} \in \mathbb{R}^{d\times d}$ denotes a weight matrix, and $\boldsymbol{b}^{(\cdot)} \in \mathbb{R}^{d\times 1}$ represents a bias vector.

Our proposed tree-based encoder is a natural extension of the conventional sequential encoder, since Tree-LSTM is a generalization of chain-structured LSTM (Tai et al., 2015). Our encoder differs from the original Tree-LSTM in the calculation of the LSTM units for the leaf nodes. The motivation is to construct the phrase nodes in a context-sensitive way, which, for example, allows the model to compute different representations for multiple occurrences of the same word in a sentence because the sequential LSTMs are calculated in the context of the previous units. This ability contrasts with the original Tree-LSTM, in which the leaves are composed only of the word embeddings without any contextual information.

3.2 Initial Decoder Setting

We now have two different sentence vectors: one is from the sequence encoder and the other from the tree-based encoder. As shown in Figure 3, we provide another Tree-LSTM unit which has the final sequential encoder unit ($\boldsymbol{h}_n$) and the tree-based encoder unit ($\boldsymbol{h}_{root}^{(phr)}$) as two child units and set it as the initial decoder $\boldsymbol{s}_1$ as follows:

$$\boldsymbol{s}_1 = g_{tree}(\boldsymbol{h}_n, \boldsymbol{h}_{root}^{(phr)}), \qquad (12)$$

where g_{tree} is the same function as f_{tree} with another set of Tree-LSTM parameters. This initialization allows the decoder to capture information from both the sequential data and phrase structures. Zoph and Knight (2016) proposed a similar method using a Tree-LSTM for initializing the decoder, with which they translate multiple source languages to one target language. When the syntactic parser fails to output a parse tree for a sentence, we encode the sentence with the sequential encoder by setting $\boldsymbol{h}_{root}^{(phr)} = \boldsymbol{0}$. Our proposed tree-based encoder therefore works with any sentences.

3.3 Attention Mechanism in Our Model

We adopt the attention mechanism into our tree-to-sequence model in a novel way. Our model gives attention not only to sequential hidden units but also to phrase hidden units. This attention mechanism tells us which words or phrases in the source sentence are important when the model decodes a target word. The j-th context vector $\boldsymbol{d}_j$ is composed of the sequential and phrase vectors weighted by the attention score $\boldsymbol{\alpha}_j(i)$:

$$\boldsymbol{d}_j = \sum_{i=1}^{n} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i + \sum_{i=n+1}^{2n-1} \boldsymbol{\alpha}_j(i)\boldsymbol{h}_i^{(phr)}. \qquad (13)$$

Note that a binary tree has $n-1$ phrase nodes if the tree has n leaves. We set a final decoder $\tilde{\boldsymbol{s}}_j$ in the same way as Equation (7).

In addition, we adopt the *input-feeding* method (Luong et al., 2015a) in our model, which is a method for feeding $\tilde{\boldsymbol{s}}_{j-1}$, the previous unit to predict the word y_{j-1}, into the current target hidden unit $\boldsymbol{s}_j$,

$$\boldsymbol{s}_j = f_{dec}(y_{j-1}, [\boldsymbol{s}_{j-1};\ \tilde{\boldsymbol{s}}_{j-1}]), \qquad (14)$$

where $[\boldsymbol{s}_{j-1};\ \tilde{\boldsymbol{s}}_{j-1}]$ is the concatenation of $\boldsymbol{s}_{j-1}$ and $\tilde{\boldsymbol{s}}_{j-1}$. The input-feeding approach contributes to the enrichment in the calculation of the decoder, because $\tilde{\boldsymbol{s}}_{j-1}$ is an informative unit which can be used to predict the output word as well as to be compacted with attentional context vectors. Luong et al. (2015a) showed that the input-feeding approach improves BLEU scores. We also observed the same improvement in our preliminary experiments.

3.4 Sampling-Based Approximation to the NMT Models

The biggest computational bottleneck of training the NMT models is in the calculation of the softmax layer described in Equation (8), because its computational cost increases linearly with the size of the vocabulary. The speedup technique with GPUs has proven useful for sequence-based NMT models (Sutskever et al., 2014; Luong et al.,

2015a) but it is not easily applicable when dealing with tree-structured data. In order to reduce the training cost of the NMT models at the softmax layer, we employ *BlackOut* (Ji et al., 2016), a sampling-based approximation method. BlackOut has been shown to be effective in RNN Language Models (RNNLMs) and allows a model to run reasonably fast even with a million word vocabulary with CPUs.

At each word prediction step in the training, BlackOut estimates the conditional probability in Equation (2) for the target word and K negative samples using a weighted softmax function. The negative samples are drawn from the unigram distribution raised to the power $\beta \in [0, 1]$ (Mikolov et al., 2013). The unigram distribution is estimated using the training data and β is a hyperparameter. BlackOut is closely related to Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2012) and achieves better perplexity than the original softmax and NCE in RNNLMs. The advantages of Blackout over the other methods are discussed in Ji et al. (2016). Note that BlackOut can be used as the original softmax once the training is finished.

4 Experiments

4.1 Training Data

We applied the proposed model to the English-to-Japanese translation dataset of the ASPEC corpus given in WAT'15.[1] Following Zhu (2015), we extracted the first 1.5 million translation pairs from the training data. To obtain the phrase structures of the source sentences, i.e., English, we used the probabilistic HPSG parser *Enju* (Miyao and Tsujii, 2008). We used Enju only to obtain a binary phrase structure for each sentence and did not use any HPSG specific information. For the target language, i.e., Japanese, we used KyTea (Neubig et al., 2011), a Japanese segmentation tool, and performed the pre-processing steps recommended in WAT'15.[2] We then filtered out the translation pairs whose sentence lengths are longer than 50 and whose source sentences are not parsed successfully. Table 1 shows the details of the datasets used in our experiments. We carried out two experiments on a small training dataset to investigate the effectiveness of our proposed model and on a large training dataset to compare our proposed methods with the other systems.

	Sentences	Parsed successfully
Train	1,346,946	1,346,946
Development	1,790	1,789
Test	1,812	1,811

Table 1: Dataset in ASPEC corpus.

	Train (small)	Train (large)
sentence pairs	100,000	1,346,946
$\|V\|$ in English	25,478	87,796
$\|V\|$ in Japanese	23,532	65,680

Table 2: Training dataset and the vocabulary sizes.

The vocabulary consists of words observed in the training data more than or equal to N times. We set $N = 2$ for the small training dataset and $N = 5$ for the large training dataset. The out-of-vocabulary words are mapped to the special token "*unk*". We added another special symbol "*eos*" for both languages and inserted it at the end of all the sentences. Table 2 shows the details of each training dataset and its corresponding vocabulary size.

4.2 Training Details

The biases, softmax weights, and BlackOut weights are initialized with zeros. The hyperparameter β of BlackOut is set to 0.4 as recommended by Ji et al. (2016). Following Józefowicz et al. (2015), we initialize the forget gate biases of LSTM and Tree-LSTM with 1.0. The remaining model parameters in the NMT models in our experiments are uniformly initialized in $[-0.1, 0.1]$. The model parameters are optimized by plain SGD with the mini-batch size of 128. The initial learning rate of SGD is 1.0. We halve the learning rate when the development loss becomes worse. Gradient norms are clipped to 3.0 to avoid exploding gradient problems (Pascanu et al., 2012).

Small Training Dataset We conduct experiments with our proposed model and the sequential attentional NMT model with the input-feeding approach. Each model has 256-dimensional hidden units and word embeddings. The number of negative samples K of BlackOut is set to 500 or 2000.

[1] http://orchid.kuee.kyoto-u.ac.jp/WAT/WAT2015/index.html

[2] http://orchid.kuee.kyoto-u.ac.jp/WAT/WAT2015/baseline/dataPreparationJE.html

Large Training Dataset Our proposed model has 512-dimensional word embeddings and d-dimensional hidden units ($d \in \{512, 768, 1024\}$). K is set to 2500.

Our code[3] is implemented in C++ using the Eigen library,[4] a template library for linear algebra, and we run all of the experiments on multi-core CPUs.[5] It takes about a week to train a model on the large training dataset with $d = 512$.

4.3 Decoding process

We use beam search to decode a target sentence for an input sentence $\boldsymbol{x}$ and calculate the sum of the log-likelihoods of the target sentence $\boldsymbol{y} = (y_1, \cdots, y_m)$ as the beam score:

$$score(\boldsymbol{x}, \boldsymbol{y}) = \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}). \quad (15)$$

Decoding in the NMT models is a generative process and depends on the target language model given a source sentence. The score becomes smaller as the target sentence becomes longer, and thus the simple beam search does not work well when decoding a long sentence (Cho et al., 2014a; Pouget-Abadie et al., 2014). In our preliminary experiments, the beam search with the length normalization in Cho et al. (2014a) was not effective in English-to-Japanese translation. The method in Pouget-Abadie et al. (2014) needs to estimate the conditional probability $p(\boldsymbol{x}|\boldsymbol{y})$ using another NMT model and thus is not suitable for our work.

In this paper, we use statistics on sentence lengths in beam search. Assuming that the length of a target sentence correlates with the length of a source sentence, we redefine the score of each candidate as follows:

$$score(\boldsymbol{x}, \boldsymbol{y}) = L_{\boldsymbol{x},\boldsymbol{y}} + \sum_{j=1}^{m} \log p(y_j | \boldsymbol{y}_{<j}, \boldsymbol{x}), \quad (16)$$

$$L_{\boldsymbol{x},\boldsymbol{y}} = \log p(len(\boldsymbol{y}) | len(\boldsymbol{x})), \quad (17)$$

where $L_{\boldsymbol{x},\boldsymbol{y}}$ is the penalty for the conditional probability of the target sentence length $len(\boldsymbol{y})$ given the source sentence length $len(\boldsymbol{x})$. It allows the model to decode a sentence by considering the length of the target sentence. In our experiments, we computed the conditional probability $p(len(\boldsymbol{y})|len(\boldsymbol{x}))$ in advance following the statistics collected in the first one million pairs of the training dataset. We allow the decoder to generate up to 100 words.

4.4 Evaluation

We evaluated the models by two automatic evaluation metrics, RIBES (Isozaki et al., 2010) and BLEU (Papineni et al., 2002) following WAT'15. We used the KyTea-based evaluation script for the translation results.[6] The RIBES score is a metric based on rank correlation coefficients with word precision, and the BLEU score is based on n-gram word precision and a Brevity Penalty (BP) for outputs shorter than the references. RIBES is known to have stronger correlation with human judgements than BLEU in translation between English and Japanese as discussed in Isozaki et al. (2010).

5 Results and Discussion

5.1 Small Training Dataset

Table 3 shows the perplexity, BLEU, RIBES, and the training time on the development data with the Attentional NMT (ANMT) models trained on the small dataset. We conducted the experiments with our proposed method using BlackOut and softmax. We decoded a translation by our proposed beam search with a beam size of 20.

As shown in Table 3, the results of our proposed model with BlackOut improve as the number of negative samples K increases. Although the result of softmax is better than those of BlackOut ($K = 500, 2000$), the training time of softmax per epoch is about three times longer than that of BlackOut even with the small dataset.

As to the results of the ANMT model, reversing the word order in the input sentence decreases the scores in English-to-Japanese translation, which contrasts with the results of other language pairs reported in previous work (Sutskever et al., 2014; Luong et al., 2015a). By taking syntactic information into consideration, our proposed model improves the scores, compared to the sequential attention-based approach.

We found that better perplexity does not always lead to better translation scores with BlackOut as shown in Table 3. One of the possible reasons is that BlackOut distorts the target word distribution

[3] https://github.com/tempra28/tree2seq

[4] http://eigen.tuxfamily.org/index.php

[5] 16 threads on Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz

[6] http://lotus.kuee.kyoto-u.ac.jp/WAT/evaluation/automatic_evaluation_systems/automaticEvaluationJA.html

	K	Perplexity	RIBES	BLEU	Time/epoch (min.)
Proposed model	500	19.6	71.8	20.0	55
Proposed model	2000	21.0	72.6	20.5	70
Proposed model (Softmax)	—	17.9	73.2	21.8	180
ANMT (Luong et al., 2015a)	500	21.6	70.7	18.5	45
+ reverse input	500	22.6	69.8	17.7	45
ANMT (Luong et al., 2015a)	2000	23.1	71.5	19.4	60
+ reverse input	2000	26.1	69.5	17.5	60

Table 3: Evaluation results on the development data using the small training data. The training time per epoch is also shown, and K is the number of negative samples in BlackOut.

	Beam size	RIBES	BLEU (BP)
Simple BS	6	72.3	20.0 (90.1)
	20	72.3	19.5 (85.1)
Proposed BS	20	**72.6**	**20.5** (91.7)

Table 4: Effects of the Beam Search (BS) on the development data.

	RIBES	BLEU
Without sequential LSTMs	69.4	19.5
With sequential LSTMs	**72.3**	**20.0**

Table 5: Effects of the sequential LSTMs in our proposed tree-based encoder on the development data.

by the modified unigram-based negative sampling where frequent words can be treated as the negative samples multiple times at each training step.

Effects of the proposed beam search Table 4 shows the results on the development data of proposed method with BlackOut ($K = 2000$) by the simple beam search and our proposed beam search. The beam size is set to 6 or 20 in the simple beam search, and to 20 in our proposed search. We can see that our proposed search outperforms the simple beam search in both scores. Unlike RIBES, the BLEU score is sensitive to the beam size and becomes lower as the beam size increases. We found that the BP had a relatively large impact on the BLEU score in the simple beam search as the beam size increased. Our search method works better than the simple beam search by keeping long sentences in the candidates with a large beam size.

Effects of the sequential LSTM units We also investigated the effects of the sequential LSTMs at the leaf nodes in our proposed tree-based encoder. Table 5 shows the result on the development data of our proposed encoder and that of an attentional tree-based encoder without sequential LSTMs with BlackOut ($K = 2000$).[7] The results show that our proposed encoder considerably outperforms the encoder without sequential LSTMs, suggesting that the sequential LSTMs at the leaf nodes contribute to the context-aware construction of the phrase representations in the tree.

5.2 Large Training Dataset

Table 6 shows the experimental results of RIBES and BLEU scores achieved by the trained models on the large dataset. We decoded the target sentences by our proposed beam search with the beam size of 20.[8] The results of the other systems are the ones reported in Nakazawa et al. (2015).

All of our proposed models show similar performance regardless of the value of d. Our ensemble model is composed of the three models with $d = 512, 768$, and 1024, and it shows the best RIBES score among all systems.[9]

As for the time required for training, our implementation needs about one day to perform one epoch on the large training dataset with $d = 512$. It would take about 11 days without using the BlackOut sampling.

Comparison with the NMT models The model of Zhu (2015) is an ANMT model (Bahdanau et al., 2015) with a bi-directional LSTM encoder, and uses 1024-dimensional hidden units and 1000-

[7] For this evaluation, we used the 1,789 sentences that were successfully parsed by Enju because the encoder without sequential LSTMs always requires a parse tree.

[8] We found two sentences which ends without *eos* with $d = 512$, and then we decoded it again with the beam size of 1000 following Zhu (2015).

[9] Our ensemble model yields a METEOR (Denkowski and Lavie, 2014) score of 53.6 with language option "-l other".

Model	RIBES	BLEU
Proposed model ($d = 512$)	81.46	34.36
Proposed model ($d = 768$)	81.89	34.78
Proposed model ($d = 1024$)	81.58	34.87
Ensemble of the above three models	**82.45**	36.95
ANMT with LSTMs (Zhu, 2015)	79.70	32.19
+ Ensemble, *unk* replacement	80.27	34.19
+ System combination, 3 pre-reordered ensembles	80.91	36.21
ANMT with GRUs (Lee et al., 2015) + character-based decoding, Begin/Inside representation	81.15	35.75
PB baseline	69.19	29.80
HPB baseline	74.70	32.56
T2S baseline	75.80	33.44
T2S model (Neubig and Duh, 2014)	79.65	36.58
+ ANMT Rerank (Neubig et al., 2015)	81.38	38.17

Table 6: Evaluation results on the test data.

dimensional word embeddings. The model of Lee et al. (2015) is also an ANMT model with a bi-directional Gated Recurrent Unit (GRU) encoder, and uses 1000-dimensional hidden units and 200-dimensional word embeddings. Both models are sequential ANMT models. Our single proposed model with $d = 512$ outperforms the best result of Zhu (2015)'s end-to-end NMT model with ensemble and unknown replacement by $+1.19$ RIBES and by $+0.17$ BLEU scores. Our ensemble model shows better performance, in both RIBES and BLEU scores, than that of Zhu (2015)'s best system which is a hybrid of the ANMT and SMT models by $+1.54$ RIBES and by $+0.74$ BLEU scores and Lee et al. (2015)'s ANMT system with special character-based decoding by $+1.30$ RIBES and $+1.20$ BLEU scores.

Comparison with the SMT models PB, HPB and T2S are the baseline SMT systems in WAT'15: a phrase-based model, a hierarchical phrase-based model, and a tree-to-string model, respectively (Nakazawa et al., 2015). The best model in WAT'15 is Neubig et al. (2015)'s tree-to-string SMT model enhanced with reranking by ANMT using a bi-directional LSTM encoder. Our proposed end-to-end NMT model compares favorably with Neubig et al. (2015).

5.3 Qualitative Analysis

We illustrate the translations of test data by our model with $d = 512$ and several attentional relations when decoding a sentence. In Figures 4 and 5, an English sentence represented as a binary tree is translated into Japanese, and several attentional relations between English words or phrases and

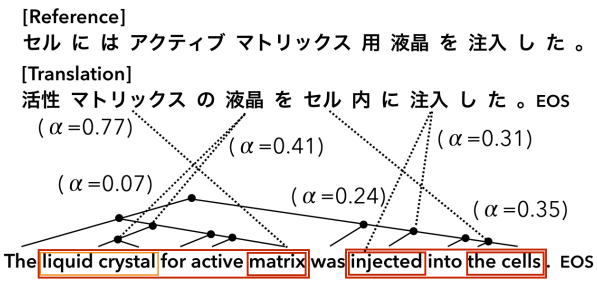


Figure 4: Translation example of a short sentence and the attentional relations by our proposed model.

Japanese word are shown with the highest attention score α. The additional attentional relations are also illustrated for comparison. We can see the target words softly aligned with source words and phrases.

In Figure 4, the Japanese word "液晶" means "liquid crystal", and it has a high attention score ($\alpha = 0.41$) with the English phrase "liquid crystal for active matrix". This is because the j-th target hidden unit $\boldsymbol{s}_j$ has the contextual information about the previous words $\boldsymbol{y}_{<j}$ including "活性 マトリックス の" ("for active matrix" in English). The Japanese word "セル" is softly aligned with the phrase "the cells" with the highest attention score ($\alpha = 0.35$). In Japanese, there is no definite article like "the" in English, and it is usually aligned with *null* described as Section 1.

In Figure 5, in the case of the Japanese word "示" ("showed" in English), the attention score with the English phrase "showed excellent performance" ($\alpha = 0.25$) is higher than that with the English word "showed" ($\alpha = 0.01$). The Japanese word "の" ("of" in English) is softly aligned with the phrase "of Si dot MOS capacitor" with the highest attention score ($\alpha = 0.30$). It is because our attention mechanism takes each previous context of the Japanese phrases "優れ た 性能" ("excellent performance" in English) and "Ｓｉ ドット ＭＯＳ コンデンサ" ("Si dot MOS capacitor" in English) into account and softly aligned the target words with the whole phrase when translating the English verb "showed" and the preposition "of". Our proposed model can thus flexibly learn the attentional relations between English and Japanese.

We observed that our model translated the word "active" into "活性", a synonym of the reference word "アクティブ". We also found similar examples in other sentences, where our model outputs

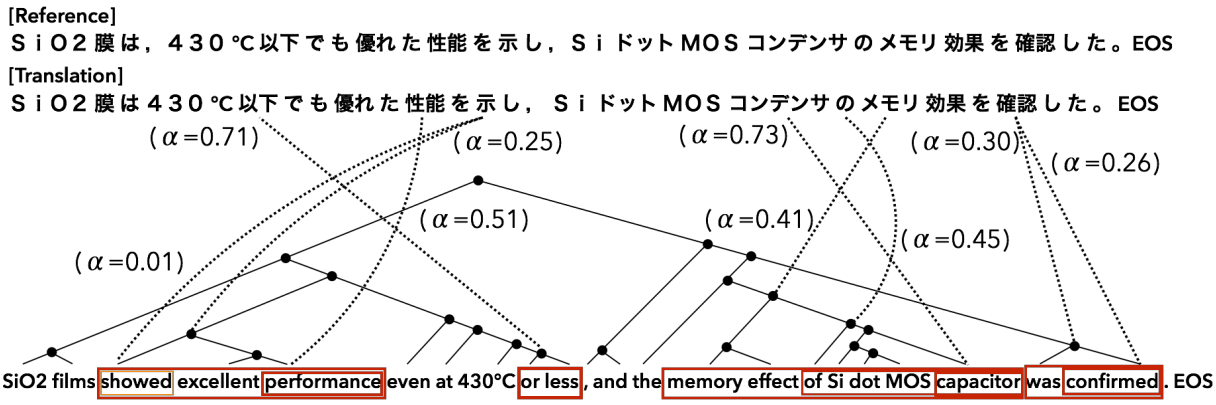


Figure 5: Translation example of a long sentence and the attentional relations by our proposed model.

synonyms of the reference words, e.g. “女” and “女性” (“female” in English) and “NASA” and “航空宇宙局” (“National Aeronautics and Space Administration” in English). These translations are penalized in terms of BLEU scores, but they do not necessarily mean that the translations were wrong. This point may be supported by the fact that the NMT models were highly evaluated in WAT’15 by crowd sourcing (Nakazawa et al., 2015).

6 Related Work

Kalchbrenner and Blunsom (2013) were the first to propose an end-to-end NMT model using Convolutional Neural Networks (CNNs) as the source encoder and using RNNs as the target decoder. The Encoder-Decoder model can be seen as an extension of their model, and it replaces the CNNs with RNNs using GRUs (Cho et al., 2014b) or LSTMs (Sutskever et al., 2014).

Sutskever et al. (2014) have shown that making the input sequences reversed is effective in a French-to-English translation task, and the technique has also proven effective in translation tasks between other European language pairs (Luong et al., 2015a). All of the NMT models mentioned above are based on sequential encoders. To incorporate structural information into the NMT models, Cho et al. (2014a) proposed to jointly learn structures inherent in source-side languages but did not report improvement of translation performance. These studies motivated us to investigate the role of syntactic structures explicitly given by existing syntactic parsers in the NMT models.

The attention mechanism (Bahdanau et al., 2015) has promoted NMT onto the next stage. It enables the NMT models to translate while aligning the target with the source. Luong et al. (2015a) refined the attention model so that it can dynamically focus on local windows rather than the entire sentence. They also proposed a more effective attentional path in the calculation of ANMT models. Subsequently, several ANMT models have been proposed (Cheng et al., 2016; Cohn et al., 2016); however, each model is based on the existing sequential attentional models and does not focus on a syntactic structure of languages.

7 Conclusion

In this paper, we propose a novel syntactic approach that extends attentional NMT models. We focus on the phrase structure of the input sentence and build a tree-based encoder following the parsed tree. Our proposed tree-based encoder is a natural extension of the sequential encoder model, where the leaf units of the tree-LSTM in the encoder can work together with the original sequential LSTM encoder. Moreover, the attention mechanism allows the tree-based encoder to align not only the input words but also input phrases with the output words. Experimental results on the WAT’15 English-to-Japanese translation dataset demonstrate that our proposed model achieves the best RIBES score and outperforms the sequential attentional NMT model.

Acknowledgments

We thank the anonymous reviewers for their constructive comments and suggestions. This work was supported by CREST, JST, and JSPS KAKENHI Grant Number 15J12597.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations*.

Yong Cheng, Shiqi Shen, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2016. Agreement-based Joint Training for Bidirectional Attention-based Neural Machine Translation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. to appear.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Proceedings of Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

Trevor Cohn, Cong Duy Vu Hoang, Ekaterina Vymolova, Kaisheng Yao, Chris Dyer, and Gholamreza Haffari. 2016. Incorporating Structural Alignment Biases into an Attentional Neural Translation Model. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. to appear.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics 2014 Workshop on Statistical Machine Translation*.

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

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13(1):307–361.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 944–952.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. 2016. BlackOut: Speeding up Recurrent Neural Network Language Models With Very Large Vocabularies. In *Proceedings of the 4th International Conference on Learning Representations*.

Rafal Józefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An Empirical Exploration of Recurrent Network Architectures. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 2342–2350.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709.

Hyoung-Gyu Lee, JaeSong Lee, Jun-Seok Kim, and Chang-Ki Lee. 2015. NAVER Machine Translation System for WAT 2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 69–73.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 609–616.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the Rare Word Problem in Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Yusuke Miyao and Jun'ichi Tsujii. 2008. Feature Forest Models for Probabilistic HPSG Parsing. *Computational Linguistics*, 34(1):35–80.

Toshiaki Nakazawa, Hideya Mino, Isao Goto, Graham Neubig, Sadao Kurohashi, and Eiichiro Sumita. 2015. Overview of the 2nd Workshop on Asian Translation. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 1–28.

Graham Neubig and Kevin Duh. 2014. On the elements of an accurate tree-to-string machine translation system. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–149.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 529–533.

Graham Neubig, Makoto Morishita, and Satoshi Nakamura. 2015. Neural Reranking Improves Subjective Quality of Machine Translation: NAIST at WAT2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 35–41.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *arXiv: 1211.5063*.

Jean Pouget-Abadie, Dzmitry Bahdanau, Bart van Merrienboer, Kyunghyun Cho, and Yoshua Bengio. 2014. Overcoming the curse of sentence length for neural machine translation using automatic segmentation. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 78–85.

Ivan A. Sag, Thomas Wasow, and Emily Bender. 2003. *Syntactic Theory: A Formal Introduction*. Center for the Study of Language and Information, Stanford, 2nd edition.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, pages 3104–3112.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Zhongyuan Zhu. 2015. Evaluating Neural Machine Translation in English-Japanese Task. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 61–68.

Barret Zoph and Kevin Knight. 2016. Multi-Source Neural Translation. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. to appear.

Coordination Annotation Extension in the Penn Tree Bank

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

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

Abstract

Coordination is an important and common syntactic construction which is not handled well by state of the art parsers. Coordinations in the Penn Treebank are missing internal structure in many cases, do not include explicit marking of the conjuncts and contain various errors and inconsistencies. In this work, we initiated manual annotation process for solving these issues. We identify the different elements in a coordination phrase and label each element with its function. We add phrase boundaries when these are missing, unify inconsistencies, and fix errors. The outcome is an extension of the PTB that includes consistent and detailed structures for coordinations. We make the coordination annotation publicly available, in hope that they will facilitate further research into coordination disambiguation. [1]

1 Introduction

The Penn Treebank (PTB) (Marcus et al., 1993) is perhaps the most commonly used resource for training and evaluating syntax-based natural language processing systems. Despite its widespread adoption and undisputed usefulness, some of the annotations in PTB are not optimal, and could be improved. The work of Vadas and Curran (2007) identified and addressed one such annotation deficiency – the lack of internal structure in base NPs. In this work we focus on the annotation of coordinating conjunctions.

Coordinating conjunctions (e.g. *"John **and** Mary"*, *"to be **or** not to be"*) are a very common syntactic construction, appearing in 38.8% of the sentences in the PTB. As noted by Hogan (2007), coordination annotation in the PTB are not consistent, include errors, and lack internal structure in many cases (Hara et al., 2009; Hogan, 2007; Shimbo and Hara, 2007). Another issue is that PTB does not mark whether a punctuation is part of the coordination or not. This was resolved by Maier et al. (2012) which annotated punctuation in the PTB .

These errors, inconsistencies, and in particular the lack of internal structural annotation turned researchers that were interested specifically in coordination disambiguation away from the PTB and towards much smaller, domain specific efforts such as the Genia Treebank (Kim et al., 2003) of biomedical texts (Hara et al., 2009; Shimbo and Hara, 2007).

In addition, we also find that the PTB annotation make it hard, and often impossible, to correctly identify the elements that are being coordinated, and tell them apart from other elements that may appear in a coordination construction. While most of the coordination phrases are simple and include only conjuncts and a coordinator, many cases include additional elements with other syntactic functions , such as markers (e.g. *"**Both** Alice and Bob"*), connectives (e.g. *"Fast and **thus** useful"*) and shared elements (e.g. *"**Bob's** principles and opinions"*) (Huddleston et al., 2002). The PTB annotations do not differentiate between these elements. For example, consider the following coordination phrases which begin with a PP:

(a) *"[in the open market]$_{PP}$, [in private transactions] or [otherwise]."*
(b) *"[According to Fred Demler]$_{PP}$, [Highland Valley has already started operating] and [Cananea is expected to do so soon]."*

Even though the first element is a conjunct only in (a), both phrases are represented with the

[1] The data is available in:
https://github.com/Jess1ca/CoordinationExtPTB

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 834–842,
Berlin, Germany, August 7-12, 2016.

marked elements as siblings.

Our goal in this work is to fix these deficiencies. We aim for an annotation in which:

- All coordination phrases are explicitly marked and are differentiated from non-coordination structures.
- Each element in the coordination structure is explicitly marked with its role within the coordination structure.
- Similar structures are assigned a consistent annotation.

We also aim to fix existing errors involving coordination, so that the resulting corpus includes as few errors as possible. On top of these objectives, we also like to stay as close as possible to the original PTB structures.

We identify the different elements that can participate in a coordination phrase, and enrich the PTB by labeling each element with its function. We add phrase boundaries when these are missing, unify inconsistencies, and fix errors. This is done based on a combination of automatic processing and manual annotation. The result is an extension of the PTB trees that include consistent and more detailed coordination structures. We release our annotation as a diff over the PTB.

The extended coordination annotation fills an important gap in wide-scale syntactic annotation of English syntax, and is a necessary first step towards research on improving coordination disambiguation.

2 Background

Coordination is a very common syntactic structure in which two or more elements are linked. An example for a coordination structure is *"Alice and Bob traveled to Mars"*. The elements (*Alice* and *Bob*) are called the *conjuncts* and *and* is called the *coordinator*. Other coordinator words include *or*, *nor* and *but*. Any grammatical function can be coordinated. For examples: *"[relatively active]$_{ADJP}$ but [unfocused]$_{ADJP}$"* ; *"[in]$_{IN}$ and [out]$_{IN}$ the market"*. While it is common for the conjuncts to be of the same syntactic category, coordination of elements with different syntactic categories are also possible (e.g. *"Alice will visit Earth [tomorrow]$_{NP}$ or [in the next decade]$_{PP}$"*).

Less common coordinations are those with non-constituent elements. These are cases such as *"equal to or higher than"*, and coordinations from the type of Argument-Cluster (e.g. *"Alice has visited 4 planets in 2014 and 3 more since then"*) and Gapping (e.g. *"Bob lives in Earth and Alice in Saturn"*) (Dowty, 1988).

2.1 Elements of Coordination Structure

While the canonical coordination cases involve conjuncts linked with a coordinator, other elements may also take part in the coordination structure: markers, connective adjectives, parentheticals, and shared arguments and modifiers. These elements are often part of the same syntactic phrase as the conjuncts, and should be taken into account in coordination structure annotation. We elaborate on the possible elements in a coordination phrase:

Shared modifiers Modifiers that are related to each of the conjuncts in the phrase. For instance, in *"Venus's density and mean temperature are very high"*, *Venus's* is a shared modifier of the conjuncts *"density"* and *"mean temperature"* [2].

Shared arguments Phrases that function as arguments for each of the conjuncts. For instance, in *"Bob cleaned and refueled the spaceship."*, *"the spaceship"* and *"Bob"* are arguments of the conjuncts *cleaned* and *refuel* [3].

Markers Determiners such as *both* and *either* that may appear at the beginning of the coordination phrase (Huddleston et al., 2002). As for example in *"Both Alice and Bob are Aliens"* and *"Either Alice or Bob will drive the spaceship"*. In addition to the cases documented by Huddleston et al, our annotation of the Penn Treebank data reveals additional markers. For examples: *"between 15 million and 20 million* ; *"first and second respectively*".

Connective adjectives Adverbs such as *so*, *yet*, *however*, *then*, etc. that commonly appear right after the coordinator (Huddleston et al., 2002). For instance *"We plan to meet in the middle of the way and then continue together"*.

Parenthetical Parenthetical remarks that may appear between the conjuncts. For examples:

[2] Here, the NP containing the coordination ("Venus's density and mean temperature") is itself an argument of "are very high".

[3] While both are shared arguments, standard syntactic analyses consider the subject (Bob) to be outside the VP containing the coordination, and the direct object (the spaceship) as a part of the VP.

"The vacation packages include hotel accommodations and, in some cases, tours"; *"Some shows just don't impress, he says, and this is one of them"*.

Consider the coordinated PP phrase in *"Alice traveled [both inside and outside the galaxy]$_{PP}$."* Here, *inside* and *outside* are the conjuncts, *both* is a marker, and *"the galaxy"* is a shared argument. A good representation of the coordination structure would allow us to identify the different elements and their associated functions. As we show below, it is often not possible to reliably extract such information from the existing PTB annotation scheme.

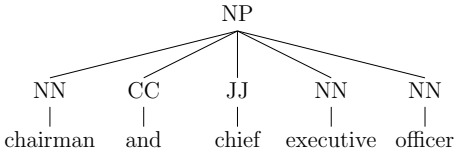
3 Coordinations in the Penn Tree Bank

We now turn to describe how coordination is handled in the PTB, focusing on the parts where we find the annotation scheme to be deficient.

There is no explicit annotation for coordination phrases Some coordinators do not introduce a coordination structure. For example, the coordinator *"and"* can be a discourse marker connecting two sentences (e.g. *"And they will even serve it themselves"*), or introduce a parenthetical (e.g. *"The Wall Street Journal is an excellent publication that I enjoy reading (and must read) daily"*). These are not explicitly differentiate in the PTB from the case where *"and"* connects between at least two elements (e.g. *"loyalty and trust"*).

NPs without internal structure The PTB guidelines (Bies et al., 1995) avoid giving any structure to NPs with nominal modifiers. Following this, 4759 NPs that include coordination were left flat, i.e. all the words in the phrase are at the same level. For example *(NP (NNP chairman) (CC and) (NP chief executive officer))* which is annotated in the PTB as:

[1]



It is impossible to reliably extract conjunct boundaries from such structures. Although work has been done for giving internal structures for flat NPs (Vadas and Curran, 2007), only 48% of the flat NP coordinators that include more than two nouns were given an internal structure, leaving 1744 cases of flat NPs with ambiguous conjunct boundaries.

Coordination parts are not categorized Coordination phrases may include markers, shared modifiers, shared arguments, connective adjectives and parentheticals. Such elements are annotated on the same level as the conjuncts[4]. This is true not only in the case of flat NPs but also in cases where the coordination phrase elements do have internal structures. For examples:

- The *Both* marker in *(NP (DT both) (NP the self) (CC and) (NP the audience))*
- The parenthetical *maybe* in *(NP (NP predictive tests) (CC and) (PRN , maybe ,) (NP new therapies))*
- The shared-modifier "the economy's" in *(NP (NP the economy's) (NNS ups) (CC and) (NNS downs))*

Automatic categorization of the phrases elements is not trivial. Consider the coordination phrase *"a phone, a job, and even into a school"*, which is annotated in the PTB where the NPs *"a phone"* and *"a job"*, the ADVP *"even"* and the PP *"into a school"* are siblings. A human reader can easily deduce that the conjuncts are *"a phone"*, *"a job"* and *"into a school"*, while *"even"* is a connective. However, for an automatic analyzer, this structure is ambiguous: NPs can be conjoined with ADVPs as well as PPs, and a coordination phrase of the form NP NP CC ADVP PP has at least two possible interpretations: (1) Coord Coord CC Conn Coord (2) Coord Coord CC Coord Shared.

Inconsistency in shared elements and markers level The PTB guidelines allows inconsistency in the case of shared ADVP pre-modifiers of VPs (e.g. *"deliberately chewed and winked"*). The pre-modifier may be annotated in the same level of the VP *((ADVP deliberately) (VP chewed and winked))* or inside it *(VP (ADVP deliberately) chewed and winked)*). In addition to this documented inconsistency, we also found markers that are inconsistently annotated in and outside the coordination phrase, such as *respectively* which is

[4] shared arguments may appear in the PTB outside the coordination phrase. For example *He* is an argument for *bought* and for *sold* in *((He) ((bought) (and) (sold) (stocks)))*.

tagged as sibling to the conjuncts in *(NP (NP Feb. 1 1990) (CC and) (NP May. 3 1990), (ADVP respectively))* and as sibling to the conjuncts parent in *(VP (VBD were) (NP 7.37% and 7.42%), (ADVP respectively))*.

Inconsistency in comparative quantity coordination Quantity phrases with a second conjunct of *more*, *less*, *so*, *two* and *up* are inconsistently tagged. Consider the following sentences: *"[50] [or] [so] projects are locked up"*, *"Street estimates of [$ 1] [or so] are low"*. The coordination phrase is similar in both the sentences but is annotated differently.

Various errors The PTB coordination structures include errors. Some are related to flat coordinations (Hogan, 2007). In addition, we found cases where a conjunct is not annotated as a complete phrase, but with two sequenced phrases. For instance, the conjuncts in the sentence *"But less than two years later, the LDP started to crumble, and dissent rose to unprecedented heights"* are *"the LDP started to crumble"* and *"dissent rose to unprecedented heights"*. In the PTB, this sentence is annotated where the first conjunct is splitted into two phrases: *"[the LDP] [started to crumble], and [dissent rose to unprecedented heights]"*.

4 Extended Coordination Annotation

The PTB annotation of coordinations makes it difficult to identify phrases containing coordination and to distinguish the conjuncts from the other parts of a coordination phrase. In addition it contains various errors, inconsistencies and coordination phrases with no internal structure. We propose an improved representation which aims to solve these problems, while keeping the deviation from the original PTB trees to a minimum.

4.1 Explicit Function Marking

We add function labels to non-terminal symbols of nodes participating in coordination structures. The function labels are indicated by appending a -XXX suffix to the non-terminal symbol, where the XXX mark the function of the node. Phrases containing a coordination are marked with a CCP label. Nodes directly dominated by a CCP node are assigned one of the following labels according to their function: *CC* for coordinators, *CONJ* for conjuncts, *MARK* for markers[5], *CONN* for connectives and parentheticals, and *SHARED* for shared modifiers/arguments. For shared elements, we deal only with those that are inside the coordination phrase. We do not assign function labels to punctuation symbols and empty elements. For example, our annotation for the sentence *"... he observed among his fellow students and, more important, among his officers and instructors ..."* is:

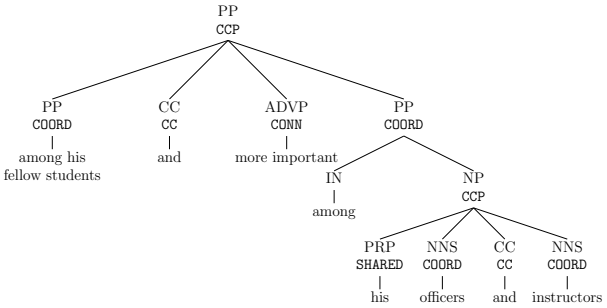


Table 1 summarizes the number of labels for each type in the enhanced version of the Penn Treebank.

Function label	#
CC	24,572
CCP	24,450
COORD	52,512
SHARED	3372
CONN	526
MARK	522

Table 1: The number of labels that were added to the Penn Treebank by type.

4.2 Changes in Tree Structure

As a guiding principle, we try not to change the structure of the original PTB trees. The exceptions to this rule are cases where the structure is changed to provide internal structure when it is missing, as well as when fixing systematic inconsistencies and occasional errors.

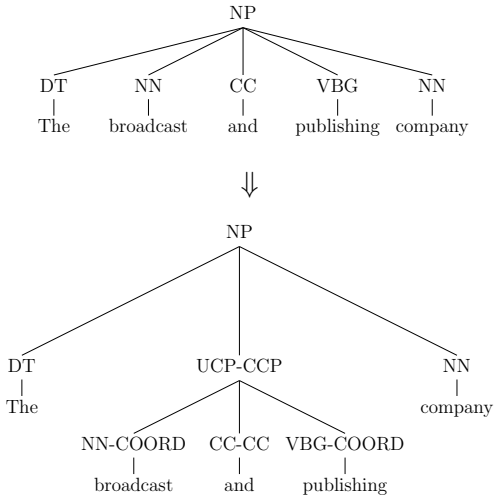
1. In flat coordination structures which include elements with more than one word, we add brackets to delimit the element spans. For instance, in the flat NP in [1] we add brackets to delimit the conjunct *"chief executive officer"*. The full phrase

[5] *both*, *either*, *between*, *first*, *neither*, *not*, *not only*, *respectively* and *together*

structure is: *(NP-CCP (NN-COORD chairman) (CC-CC and) (NP-COORD chief executive officer))*.

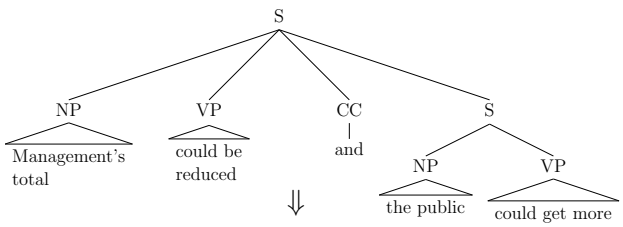
2. Comparative quantity phrases (*"5 dollars or less"*) are inconsistently analyzed in the PTB. When needed, we add an extra bracket with a QP label so they are consistently analyzed as *"5 dollars [or less]$_{QP}$"*. Note that we do not consider these cases as coordination phrases.

3. We add brackets to delimit the coordination phrase in flat cases that include coordination between modifiers while the head is annotated in the same phrase:



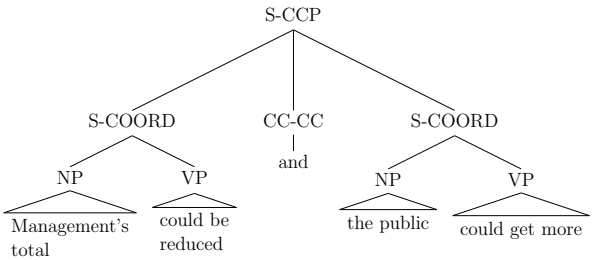
company, which is the head of the phrase, is originally annotated at the same level as the conjuncts *broadcast* and *publishing*, and the determiner *the*. In such cases, the determiner and modifiers are related to the head which is not part of the coordination phrase, requiring the extra bracketing level to delimit the coordination. This is in contrast to the case of coordination between verbs (e.g *"Bob (VP cleaned and refueled the spaceship)"*), where the non coordinated elements (*"the spaceship"*) are shared.

4. When a conjunct is split into two phrases or more due to an error, we add extra brackets to delimit the conjunct as a complete phrase:



Type	#
(1) Flat structures	1872
(2) Comparative quantity phrases	52
(3) Coordination between modifiers	1264
(4) Coordination with errors	213
(5) ADVP inconsistency	206

Table 2: The number of subtrees in the Penn Treebank that were changed in our annotation by type.



5. We consolidate cases where markers and ADVP pre-modifiers are annotated outside the coordination phrase, so they are consistently annotated inside the coordination phrase.

Table 2 summarizes the numbers and types of subtrees that receive a new tree structure in the enhanced version of the Penn Treebank.

5 The Annotation Process

Some of the changes can be done automatically, while other require human judgment. Our annotation procedure combines automatic rules and manual annotation that was performed by a dedicated annotator that was trained for this purpose.

5.1 Explicit marking of coordination phrases

We automatically annotate coordination phrases with a CCP function label. We consider a phrase as coordination phrase if it includes a coordinator and at least one phrase on each side of the coordinator, unlike coordinators that function as discourse markers or introduce parentheticals, which appear as the first element in the phrase.

5.2 Assigning internal structure to flat coordinations

Flat coordinations that include only a coordinator and two conjuncts (e.g. *(NP (NNP Poland) (CC and) (NNP Hungary))*) are trivial and are left with the same structure. For the rest of the flat coordinations (3498 cases), we manually annotated the elements spans. For example, given the flat

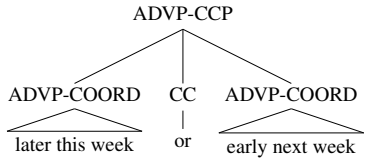
NP: *"[General]$_{NNP}$ [Electric]$_{NNP}$ [Co.]$_{NNP}$ [executives]$_{NNS}$ [and]$_{CC}$ [lawyers]$_{NNS}$"*. The annotator is expected to provide the analysis: *"[General Electric Co.] [executives] [and] [lawyers]"*. We then add brackets around multi-token elements (e.g. *"General Electric Co."*), and set the label according the syntactic structure. The annotation was done while ignoring inner structures that were given in the NP-Bracketing extension of Vadas and Curran (2007). We compare agreement with their annotations in the next section.

To handle cases such as in 4.2(3), where the coordination is between modifiers of a head which is annotated in the PTB on the same level of the conjuncts, we first identify potential candidate phrases of this type by looking for coordination phrases where the last element was not tagged by the annotator as a conjunct. Out of this set, we remove cases where we can reliably identify the non-conjunct element as a marker. For the rest of the cases, we distinguish between NP phrases and non-NP phrases. For NP phrases, we automatically add extra brackets to delimit the coordination phrase span so that it includes only the coordinated modifiers. For the rest of the phrases we found that an such automatic procedure was not feasible (consider the ADVP phrases: *(ADVP (RBR farther) (CC and) (RBR farther) (RB apart))* ; *(ADVP (RB up) (CC and) (RB down) (NP (NNP Florida)))*. The first phrase head is *apart* while in the second phrase, *Florida* is a complement). We manually annotated the coordination phrase boundary in these cases.

When adding an extra tree level in this cases, we set its syntactic label to UCP when the conjuncts are from different types and same as the conjuncts label when the conjuncts are from the same type.[6]

5.3 Annotating roles within coordination phrases

Cases where there are only a coordinator and two siblings in the coordinated phrase are trivial to automatically annotate, marking both siblings as conjuncts:



To categorize the phrase elements for the rest of the coordination phrases, we first manually marked the conjuncts in the sentence (for flat structures, the conjuncts were already annotated in the internal structure annotation phase). The annotator was given a sentence where the coordinator and the coordination phrase boundaries are marked. For example "`Coke has been able to improve (bottlers' efficiency and production, {and} in some cases, marketing)`". The annotation task was to mark the conjuncts.[7] We automatically concluded the types of the other elements according to their relative position – elements before or after the conjuncts are categorized as markers/shared, while an element between conjuncts is a connective or the coordinator itself.

Mismatches with the PTB phrase boundaries In 5% of the cases of coordination with inner structure, a conjunct span as it was annotated by our annotator was not consistent with the elements spans in the PTB. For example, the annotator provided the following annotation: "`(The [economic loss], [jobs lost], [anguish],[frustration] {and} [humiliation]) are beyond measure`", treating the determiner "The" as a shared modifier. In contrast, the PTB analysis considers "The" as part of the first conjunct ("`[The economic loss]`").

The vast majority of the mismatches were on the point of a specific word such as *the* (as demonstrated in the above example), *to*, *a* and punctuation symbols. In a small number of cases the mismatch was because of an ambiguity. For example, in "`The declaration immediately made the counties eligible for (temporary housing, grants {and} low-cost loans to cover uninsured property losses)`" the annotator marked *"temporary housing"*, *"grants"*, and *"low-cost loans"* as conjuncts (leaving *"to cover uninsured property loss"* as a shared

[6]When the conjuncts are in POS level, a corresponding syntactic label is set. For example: ***(NP-CCP (NN-COORD head) (CC-CC and) (NNS-COORD shoulders))***

[7]The coordination phrase boundaries were taken from the PTB annotations and were used to focus the annotators attention, rather than to restrict the annotation. The annotators were allowed to override them if they thought they were erronous. We did not encounter such cases.

modifier, while the PTB annotation considers *"to cover..."* as part of the last conjunct. Following our desiderata of minimizing changes to existing tree structures, in a case of a mismatch we extend the conjunct spans to be consistent with the PTB phrasing (each such case was manually verified).

5.4 Handling inconsistencies and errors

We automatically recognize ADVPs that appear right before a VP coordination phrase and markers that are adjunct to a coordination phrase. We change the structure such that such ADVPs and markers appear inside the coordination phrase.

Quantity phrases that includes two conjuncts with a second conjunct of *more, less, so, two* and *up* are automatically recognized and consolidated by adding an extra level.

Errors in conjuncts span are found during the manual annotation that is done for the categorization. When the manual annotation includes a conjunct that is originally a combination of two siblings phrases, we add extra brackets and name the new level according to the syntactic structure.

6 Annotator Agreement

We evaluate the resulting corpus with inter-annotators agreement for coordination phrases with inner structure as well as agreement with the flat conjuncts that were annotated in the NP bracketing annotation effort of Vadas and Curran (2007).

6.1 Inter-annotator agreement

To test the inter-annotator agreement, we were assisted with an additional linguist who annotated 1000 out of 7823 coordination phrases with inner structure. We measured the number of coordination phrases where the spans are inconsistent at least in one conjunct. The annotators originally agreed in 92.8% of the sentences. After revision, the agreement increased to 98.1%. The disagreements occurred in semantically ambiguous cases. For instance, *"potato salad, baked beans and pudding, plus coffee or iced tea"* was tagged differently by the 2 annotators. One considered *"pudding"* as the last conjunct and the other marked *"pudding, plus coffee or iced tea"*.

6.2 Agreement with NP Bracketing for flat coordinations

The NP Bracketing extension of Vadas and Curran (2007) includes inner structures for flat NP phrases in the PTB, that are given an internal structure using the NML tag. For instance, in *(NP (NNP Air) (NNP Force) (NN contract))*, *"Air Force"* is considered as an independent entity and thus is delimited with the NML tag: *(NP (NML (NNP Air) (NNP Force)) (NN contract))*.

As mentioned, 48% (1655 sentences) of the NP flat coordination were disambiguated in this effort.[8] For these, the agreement on the conjuncts spans with the way they were marked by our annotators is 88%. The disagreements were in cases where a modifier is ambiguous. For examples consider *"luxury"* in *"The luxury airline and casino company"*, *"scientific"* in *"scientific institutions or researchers"* and *"Japanese"* in *"some Japanese government officials and businessmen"*. In cases of disagreement we followed our annotators decisions.[9]

	R	P	F1
PTB + NPB	90.41	86.12	88.21
PTB + NPB + CCP	90.83	91.18	91.01

Table 3: The parser results on section 22.

7 Experiments

We evaluate the impact of the new annotation on the PTB parsing accuracy. We use the state-of-the-art Berkeley parser (Petrov et al., 2006), and compare the original PTB annotations (including Vadas and Curran's base-NP bracketing – **PTB+NPB**) to the coordination annotations in this work (**PTB+NPB+CCP**). We use sections 2-21 for training, and report accuracies on the traditional dev set (section 22). The parse trees are scored using EVALB (Sekine and Collins, 1997).

Structural Changes We start by considering how the changes in tree structures affect the parser performance. We compared the parsing performance when trained and tested on PTB+NPB, to the parsing performance when trained and tested on PTB+NPB+CCP. The new function labels were ignored in both training and testing. The results

[8] We consider a flat NP coordination as disambiguated if it includes a coordinator and two other elements, i.e.: *(NML (NML (NN eye) (NN care)) (CC and) (NML (NN skin) (NN care)))* ; *(NML (NN buy) (CC or) (NN sell))*.

[9] A by-product of this process is a list of ambiguous modifier attachment cases, which can be used for future research on coordination disambiguation, for example in designing error metrics that take such annotator disagreements into account.

Gold \ Pred	CC	CCP	COORD	MARK	SHARED	CONN	None	Err
CC	849					1	5	
CCP		552	1				91	205
COORD		3	1405		2		184	200
MARK				9			2	1
SHARED	1				29		85	3
CONN						1	4	2
None	4	124	113	4	26	14		

Table 4: Confusion-matrix over the predicted function labels. **None** indicate no function label (a constituent which is not directly inside a CCP phrase). **Err** indicate cases in which the gold span was not predicted by the parser.

are presented in Table 3. Parsing accuracy on the coordination-enhanced corpus is higher than on the original trees. However, the numbers are not strictly comparable, as the test sets contain trees with somewhat different number of constituents. To get a fairer comparison, we also evaluate the parsers on the subset of trees in section 22 whose structures did not change. We check two conditions: trees that include coordination, and trees that do not include coordination. Here, we see a small drop in parsing accuracy when using the new annotation. When trained and tested on PTB+NPB+CCP, the parser results are slightly decreased compared to PTB+NPB – from 89.89% F1 to 89.4% F1 for trees with coordination and from 91.78% F1 to 91.75% F1 for trees without coordination. However, the drop is small and it is clear that the changes did not make the corpus substantially harder to parse. We also note that the parsing results for trees including coordinations are lower than those for trees without coordination, highlighting the challenge in parsing coordination structures.

Function Labels How good is the parser in predicting the function labels, distinguishing between conjuncts, markers, connectives and shared modifiers? When we train and test the parser on trees that include the function labels, we see a rather large drop in accuracy: from 89.89% F1 (for trees that include a coordination) to 85.27% F1. A closer look reveals that a large part of this drop is superficial: taking function labels into account cause errors in coordination scope to be punished multiple times.[10] When we train the parser with function labels but ignore them at evaluation time, the results climb back up to 87.45% F1. Furthermore, looking at coordination phrases whose structure was perfectly predicted (65.09% of the cases), the parser assigned the correct function label for all the coordination parts in 98.91% of the cases. The combined results suggest that while the parser is reasonably effective at assigning the correct function labels, there is still work to be done on this form of disambiguation.

The availability of function labels annotation allows us to take a finer-grained look at the parsing behavior on coordination. Table 4 lists the parser assigned labels against the gold labels. Common cases of error are (1) conjuncts identification – where 200 out of 1794 gold conjuncts were assigned an incorrect span and 113 non-conjunct spans were predicted as participating as conjuncts in a coordination phrase; and (2) Shared elements identification, where 74.57% of the gold shared elements were analyzed as either out of the coordination phrase or as part of the last coordinates. These numbers suggest possible areas of future research with respect to coordination disambiguation which are likely to provide high gains.

8 Conclusions

Coordination is a frequent and important syntactic phenomena, that pose a great challenge to automatic syntactic annotation. Unfortunately, the current state of coordination annotation in the PTB is lacking. We present a version of the PTB with improved annotation for coordination structure. The

[10] Consider the gold structure (NP (NP-CCP (DT-MARK a) (NP-COORD b) (CC and) (NP-COORD c) (PP-SHARED d))) and the incorrect prediction (NP (DT a) (NP-CCP (NP-COORD b) (CC and) (NP-COORD c)) (PP d)). When taking only the syntactic labels into account there is only the mistake of the coordination span. When taking the coordination roles into account, there are two additional mistakes – the missing labels for a and d.

new annotation adds structure to the previously flat NPs, unifies inconsistencies, fix errors, and marks the role of different participants in the coordination structure with respect to the coordination. We make our annotation available to the NLP community. This resource is a necessary first step towards better disambiguation of coordination structures in syntactic parsers.

Acknowledgments

This work was supported by The Allen Institute for Artificial Intelligence as well as the German Research Foundation via the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

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

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

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

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

Rodney Huddleston, Geoffrey K Pullum, et al. 2002. The cambridge grammar of english. *Language. Cambridge: Cambridge University Press*, pages 1273–1362.

J-D Kim, Tomoko Ohta, Yuka Tateisi, and Junichi Tsujii. 2003. Genia corpusa semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl 1):i180–i182.

Wolfgang Maier, Erhard Hinrichs, Sandra Kübler, and Julia Krivanek. 2012. Annotating coordination in the penn treebank. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 166–174. Association for Computational Linguistics.

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

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

Satoshi Sekine and Michael Collins. 1997. Evalb bracket scoring program. *URL http://nlp. cs. nyu. edu/evalb/EVALB. tgz.*

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

David Vadas and James Curran. 2007. Adding noun phrase structure to the penn treebank. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 240.

Analyzing Biases in Human Perception of User Age and Gender from Text

Lucie Flekova[*]
Ubiquitous Knowledge Processing Lab
Department of Computer Science
Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

Jordan Carpenter and **Salvatore Giorgi**
Positive Psychology Center
University of Pennsylvania

Lyle Ungar and **Daniel Preoţiuc-Pietro**
Computer & Information Science
University of Pennsylvania

Abstract

User traits disclosed through written text, such as age and gender, can be used to personalize applications such as recommender systems or conversational agents. However, human perception of these traits is not perfectly aligned with reality. In this paper, we conduct a large-scale crowdsourcing experiment on guessing age and gender from tweets. We systematically analyze the quality and possible biases of these predictions. We identify the textual cues which lead to miss-assessments of traits or make annotators more or less confident in their choice. Our study demonstrates that differences between real and perceived traits are noteworthy and elucidates inaccurately used stereotypes in human perception.

1 Introduction

There are notable differences between actual user traits and their perception by others (John and Robins, 1994; Kobrynowicz and Branscombe, 1997). Assessments of the perceived traits are dependent, for example, on the interpretation skills of a judge (Kenny and Albright, 1987) and the ability of users to deliberately adjust their behavior to the way they intend to be perceived e.g., for following a social goal (Kanellakos, 2002). People typically use stereotypes – a set of beliefs, generalizations, and associations about a social group – to make judgements about others. The discrepancy between stereotypes and actual group differences is an important topic in psychological research (Eagly, 1995; Dovidio et al., 1996; John and Robins, 1994; Kobrynowicz and Branscombe, 1997). Such differences are likely reflected through one's writing.

With the Internet a substantial part of daily life, users leave enough footprints which allow algorithms to learn a range of individual traits, some with even higher accuracy than the users' own family (Youyou et al., 2015). With an increase in readily available user generated content, prediction of user attributes has become more popular than ever. Researchers built learning models to infer different user traits from text, such as age (Rao et al., 2010), gender (Burger et al., 2011; Flekova and Gurevych, 2013), location (Eisenstein et al., 2010), political orientation (Volkova et al., 2014), income (Preoţiuc-Pietro et al., 2015c), socio-economic status (Lampos et al., 2016), popularity (Lampos et al., 2014), personality (Schwartz et al., 2013) or mental illnesses (De Choudhury et al., 2013; Coppersmith et al., 2014; Preoţiuc-Pietro et al., 2015a).

Prediction models are trained on large data sets with labels extracted either from user self-reports (Preoţiuc-Pietro et al., 2015b) or perceived from annotations (Volkova et al., 2015; Volkova and Bachrach, 2015). The former is useful in obtaining accurate prediction models for unknown users while the latter is more suitable in applications that interact with humans. Previous studies showed the implications of perceived individual traits to the believability and likability of autonomous agents (Bates, 1994; Loyall and Bates, 1997; Baylor and Kim, 2004).

This study aims to emphasize the differences between real user traits and how these are perceived by humans from Twitter posts. In this context, we address the following research questions:

[*] Project carried out during a research stay at the University of Pennsylvania

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 843–854,
Berlin, Germany, August 7-12, 2016.

- How accurate are people at judging traits of other users?
- Are there systematic biases humans are subject to?
- What are the implications of using human perception as a proxy for truth?
- Which textual cues lead to a false perception of the truth?
- Which textual cues make people more or less confident in their ratings?

We use age and gender as target traits for our analysis, as these are considered basic categories in person assessment (Quinn and Macrae, 2005) and are highly studied by previous research. Using a large-scale crowdsourcing experiment, we demonstrate that human annotators are generally accurate in assessing the traits of others. However, they make systematically different types of errors compared to a prediction model trained using the bag-of-words assumption. This hints at the fact that annotators over-emphasize some linguistic features based on their stereotypes. We show how this phenomenon can be leveraged to improve prediction performance and demonstrate that by replacing self-reports with perceived annotations we introduce systematic biases into our models.

In our analysis section, we directly test the accuracy of these stereotypes, as the human predictions must rely on these theories of relative differences between groups if no explicit cues are mentioned. We uncover remarkable differences between actual and perceived traits by using multiple lexical features: unigrams, clusters of words built from word embeddings and emotions expressed through posts. In our analysis of features that lead to wrong assessments we uncover that humans mostly rely on accurate stereotypes from textual cues, but sometimes over-emphasize them. For example, annotators assume that males post more than they do about sports and business, females show more joy, older users more interest in politics and younger users use more slang and are more self-referential. Similarly, we highlight the textual features which lead to higher self-reported confidence in guesses, such as the mentions of family and beauty products for gender or college and school related topics for age.

2 Related Work

Studying gender differences has been a popular psychological interest over the past decades (Gleser et al., 1959; McMillan et al., 1977). Traditional studies worked on small data sets, which sometimes led to contradictory results – (Mulac et al., 1990) cf. (Pennebaker et al., 2003). Over the past years, researchers discovered a wide range of gender differences using large collections of data from social media or books combined with more sophisticated techniques. For example, Schler et al. (2006) apply machine learning techniques to a corpus of 37,478 blogs from the Blogger platform and find differences in the topics males and females discuss. Newman et al. (2008) showed that female authors are more likely to include pronouns, verbs, references to home, family, friends and to various emotions. Male authors use longer words, more articles, prepositions and numbers. Topical differences include males writing more about current concerns (e.g., money, leisure or sports). More recent author profiling experiments (Rangel et al., 2014; Rangel et al., 2015) revealed that gender can be well predicted from a large spectrum of textual features, ranging from paraphrase choice (Preoţiuc-Pietro et al., 2016), emotions (Volkova and Bachrach, 2016), part-of-speech (Johannsen et al., 2015) and abbreviation usage to social network metadata, web traffic (Culotta et al., 2015), apps installed (Seneviratne et al., 2015) or Facebook likes (Kosinski et al., 2013). Bamman et al. (2014) also examine individuals whose language does not match their automatically predicted gender. Most of these experiments were based on self-reported gender in social media profiles.

The relationship between age and language has also been extensively studied by both psychologists and computational linguists. Schler et al. (2006) automatically classified blogposts into three age groups based on self-reported age using features from the Linguistic Inquiry and Word Count Framework (Pennebaker et al., 2001), online slang and part-of-speech information. Rosenthal and McKeown (2011) analyzed how both stylistic and lexical cues relate to gender on blogs. On Twitter, Nguyen et al. (2013) analyzed the relationship between language use and age, modelled as a continuous variable. They found similar language usage trends for both genders, with increasing word and tweet length with age, and an increasing tendency to write more grammatically correct, standardized

text. Flekova et al. (2016) identified age specific differences in writing style and analyzed their impact beyond income. Recently, Nguyen et al. (2014) showed that age prediction is more difficult as age increases, specifically over 30 years. Hovy and Søgaard (2015) showed that the author age is a factor influencing training part-of-speech taggers.

Recent results on social media data report a performance of over 90% for gender classification and a correlation of $r \sim 0.85$ for age prediction (Sap et al., 2014). However, authors can introduce their biases in text (Recasens et al., 2013). Accurate prediction of the true user traits is important for applications such as recommender systems (Braunhofer et al., 2015) or medical diagnoses (Chattopadhyay et al., 2011). Influencing perceived traits, on the other hand, enables a whole different range of applications - for example, researchers demonstrated that the perceived demographics influence student attitude towards a tutor (Baylor and Kim, 2004; Rosenberg-Kima et al., 2008). Perception alterations do not only strive for likeability - people intentionally use linguistic nuances to express social power (Kanellakos, 2002), which can be recognized by computational means (Bramsen et al., 2011). McConnell and Fazio (1996) show how gender-marked language colors the perception of target personality characteristics – enhanced accessibility of masculine and feminine attributes brought about by frequent exposure to occupation title suffixes influences the inferences drawn about the target person.

3 Data

In this study, we focus on analyzing human perception of two user traits: gender and age. For judging, we build data sets using publicly available Twitter posts from users with known self-reported age and gender. To study gender, we use the users from Burger et al. (2011), which are mapped to their self-identified gender as mentioned in other user public profiles linked to their Twitter account. This data set consists of 67,337 users, from which we subsample 2,607 users for human assessment. The age data set consists of 826 users that self-reported their year of birth and Twitter handle as part of an online survey.

We use the Twitter API to download up to 3200 tweets from these users. These are filtered for English language using an automatic method (Lui and Baldwin, 2012) and duplicate tweets are eliminated (i.e., having the same first 6 tokens) as these are usually generated automatically by apps. Tweet URLs and @-mentions are anonymized as they may contain sensitive information or cues external to language use. For human assessment, we randomly select 100 tweets posted in the same 6 month time interval from the users where gender is known. For the users of known age we randomly select 100 tweets posted during the year 2015.

4 Experimental Setup

We use Amazon Mechanical Turk to create crowdsourcing tasks for predicting age and gender from tweets. Each HIT consists of 20 tweets randomly sampled from the pool of 100 tweets of a single user. Each user was assessed independently by 9 different annotators. Using only these tweets as cues, the annotators were asked to predict either age (integer value) or gender (forced choice binary male/female) and self-rate the confidence of their guess on a scale from 1 (not at all confident) to 5 (very confident).

Participants received a small compensation (.02$) for each rating and could repeat the task as many times as they wished, but never for the same author. They were also presented with an initial bonus (.25$) and a similar one upon completing a number of guesses. For quality control, we used a set of HITs where the user's age or gender was explicitly stated within the top 10 tweets displayed in the task. The control HIT appeared 10% of the time and all annotators missing the correct answer twice were excluded from annotation and all their HITs invalidated. A total of 28 annotators were banned from the study. Further, we limited annotator location to the US and they had to spend at least 10 seconds on each HIT before they were allowed to submit their guess.

5 Crowdsourcing Results

We first analyze the annotator performance on the gender and age prediction tasks from text. For gender, individual ratings have an overall accuracy of 75.7% (78.3% for females and 72.8% for males). The pairwise inter-annotator agreement for 9 annotators is 70.0%, Fleiss' Kappa 39.6% and Krippendorf's Alpha 39.6%, while keeping in mind that the annotators are not the same for all Twitter users. In terms of confidence, average self-rated confidence for correct guesses is $\mu = 3.47$, while average confidence for wrong guesses is $\mu = 2.84$. In total,

1083 individual annotators performed an average of $\mu = 22.3$ ratings with the standard deviation $\sigma = 32.76$ and the median of 12.

We use the majority vote as the method of label aggregation for gender prediction. The majority vote accuracy on predicting the gender of Twitter users is 85.8% with the majority class baseline being 51.9% female, a result comparable to a previous study (Nguyen et al., 2014). Table 1a presents the gender confusion matrix. Female users were more often classified into a correct class (88.3% recall for females cf. 83.5% for males). The majority of errors was caused by male users mislabeled as female. This results in higher precision on classifying male users (86.9% cf. 85.3% for females). In terms of overall self-reported confidence of the annotators, decisions on actual female users were on average more confidently rated ($\mu = 3.60$) compared to males ($\mu = 3.31$), which is in consensus with higher accuracy for females. Figure 2 shows the relationship between annotation accuracy and average confidence per Twitter users. The relationship is non-linear, with the average confidence in the 1–3 range for gender having little impact on the prediction accuracy.

For the age annotations, the correlation between predicted and real age for individual ratings is $r = 0.416$. The mean absolute error (MAE) is 7.31, while the baseline MAE obtained if predicted the sample mean real age is 8.61. The intraclass correlation coefficient between the 9 ratings is 0.367 and taking into account the fact that the annotators were different across users (Shrout and Fleiss, 1979), while the average standard deviation of the 9 user guesses for a single Twitter user is $\sigma = 5.60$. Individual rating confidence and the Mean Absolute Error (MAE) are anti-correlated with $r = -0.112$, matching the expectation that higher self-reported confidence leads to lower errors. The 691 different annotators performed on average $\mu = 10.68$ ratings with standard deviation $\sigma = 21.95$ and a median of only 4 ratings. Based on feedback, this was due to the difficulty of the age task.

In the rest of the age experiments, we consider the predicted age of a user as a mean of the 9 human guesses. Overall, the correlation between average predicted age and real age is $r = 0.631$. The MAE of the average predicted age is 6.05. MAE and average self-rated confidence by user are negatively correlated with $r = -0.21$. Figure 3 plots annotation confidence on a Twitter user level and MAE of

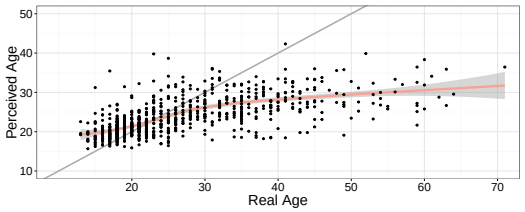


Figure 1: Real age predictions compared to average predicted age. The line shows a LOESS fit.

age guesses. Again, the relationship between confidence and MAE is non-linear, with confidences of 1–2 having similar average MAE, with the error decreasing as the average of the confidence ratings per author is higher. Figure 1 shows a scatter plot comparing real and predicted age together with a non-linear fit of the data. From this figure, we observe that annotators under-predict age, especially for older users. The correlation of MAE with real age is very high ($r = 0.824$) and the residuals are not normally distributed.

Figures 4 and 5 show the accuracy if only a subsample of the ratings is used and the labels are aggregated using majority vote for gender and using average ratings for age. For gender, we notice that accuracy abruptly increases from 1 to 3 votes and to a lesser extent from 3 to 5 votes, but the differences between 5, 7 and 9 votes are very small. Similarly, for age, MAE decreases up until using 4 guesses, where it reaches a plateau. These experiments suggest that a human perception accuracy can be sufficiently approximated using up to 5 ratings - additional annotations after this point have negligible contribution.

Finally, the individual annotator accuracy is independent on the number of users rated. For gender, the Pearson correlation between accuracy and number of ratings performed is $r = .009$ ($p = .75$) and for age the Pearson correlation between MAE and the number of ratings performed by a user is $r = -.013$ ($p = .71$). This holds even when excluding users who performed few ratings.

6 Uncovering Systematic Biases

In this section, we use the extended gender data set in order to investigate if human guesses contain systematic biases by comparing these guesses to those from a bag-of-words prediction model. We then test what is the impact of using human guesses as labels and if human ratings offer additional in-

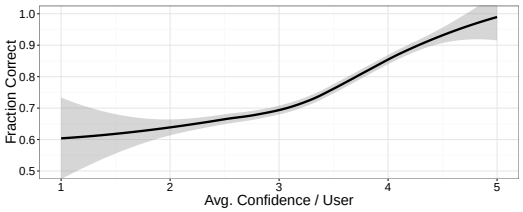


Figure 2: Gender – Fraction of correct guesses as a function of average confidence per rated Twitter user. Black line shows a LOESS fit.

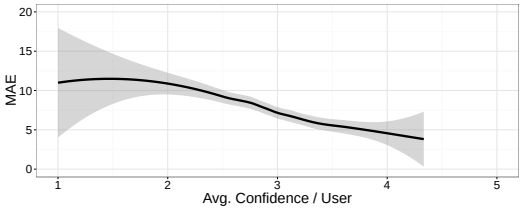


Figure 3: Age – Mean Absolute Error as a function of average confidence per rated Twitter user. Black line shows a LOESS fit.

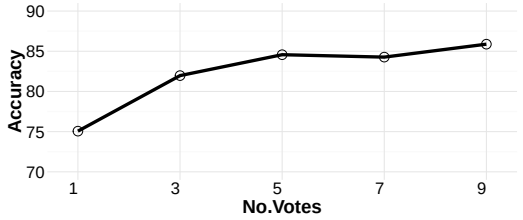


Figure 4: Gender – Majority vote accuracy based on number of annotator guesses aggregated.

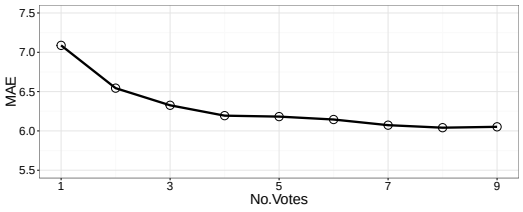


Figure 5: Age – Average Mean Absolute Error based on number of annotator guesses aggregated.

formation to predictive models.[1]

6.1 Comparison to Bag-of-Words Predictions

First, we test the hypothesis that annotators emphasize certain stereotypical words to make their guesses. To study their impact, we compare human guesses with those from a statistical model using the bag-of-words assumption for systematic differences. The automatic prediction method using bag-of-words text features offers a generalisation of individual word usage patterns shielded from biases.

We use Support Vector Machines (SVM) with a linear kernel and ℓ_1 regularization (Tibshirani, 1996), similarly to the state-of-the-art method in predicting user age and gender (Sap et al., 2014). The features for these models are unigram frequency distributions computed over the aggregate set of messages from each user. Due to the sparse and large vocabulary of social media data, we limit the unigrams to those used by at least 1% of users.

We train a classifier on a balanced set of 11,196 Twitter users from our extended data set. We test on the 2,607 users rated by the annotators using only the 100 tweets the humans had access when making their predictions. Table 1b shows the system performance reaching an accuracy of 82.9%, with the human performance on the same data at 85.88%. In contrast to the human prediction, the precision is higher for classifying females (84.9% cf. 80.9% for males) and the recall is higher for males (85.4% cf. 80.4% for female). This is caused by both higher classifier accuracy for males and by a switch in rank between the type I and type II errors.

In Table 1c we directly compare the human and automatic predictions, highlighting that 13.6% of the labels are different. Moreover, there is an asymmetry between the tendency of humans to mislabel males with females and the classifier. This leads to the conclusion that humans are sensitive to biases which we will qualitatively investigate in the following sections.

6.2 Human Predictions as Labels

Previously, we have shown that perceived annotated traits are different in many aspects to actual traits. To quantify their impact, we use these labels for training two classifiers and compare them on predicting the true gender for unseen users.

Both systems are trained on the 260,700 messages from 2,607 users and only differ in the labels assigned to users: majority annotator vote or self-reports. Results on the held-out set of 11,196 users (of which 6,851 males and 7,596 females) are presented in Table 2. The system trained on real labels outperforms that trained on perceived ones (accuracy of 85.32% cf. 83.40%). Furthermore, in the system trained on perceived labels, the same type of error as for the human annotation is more prevalent and is overemphasized compared to our

[1]Experiments for age could not be replicated due to insufficient labeled users.

(a) Majority annotator vote.

Real		Pred.H Male	Pred.H Female
	Male	40.1%	7.9%
	Female	6.1%	45.8%

(b) Classifier.

Real		Pred.C Male	Pred.C Female
	Male	42.2%	7.2%
	Female	9.9%	40.7%

(c) Classifier compared to majority annotator vote.

Pred.C		Pred.H Male	Pred.H Female
	Male	40.3%	8.0%
	Female	5.6%	46.1%

Table 1: Normalized confusion matrices of human annotations (**Pred.H**) to ground truth (**Real**), classifier performance (**Pred.C**) to ground truth (**Real**), and human annotations (**Pred.H**) to classifier performance (**Pred.C**) on the same data set.

previous results – males are predicted with high precision (85%) but low recall (79%) and many of them are misclassified as women. In the system trained on ground truth, both types of errors are more balanced with more males classified correctly – similar precision (84%) but higher recall (86%).

6.3 Combining Human and Automatic Predictions

We have shown that human perceived labels and automatic methods capture different information. This information may be leveraged to obtain better overall predicting performance. We test this by using a linear model that combines two features: the human guesses – measured as the proportion of guesses for female – and classifier prediction – binary value. Even this simple method of label combination obtains a classification accuracy of 87.7%, significantly above majority vote of human guesses (85.8%) and automatic prediction (82.9%) individually. This demonstrates that both methods can complement each other if an increase in accuracy is needed.

(a) Trained on perceived gender. Accuracy = 83.4%

Real		Pred. Male	Pred. Female
	Male	37.5%	9.9%
	Female	6.6%	45.9%

(b) Trained on actual gender. Accuracy = 85.3%

Real		Pred. Male	Pred. Female
	Male	40.5%	6.9%
	Female	7.8%	44.7%

Table 2: Normalized confusion matrices for system comparison when using perceived or ground truth labels.

7 Textual Differences between Perceived and Actual Traits

We have so far demonstrated that differences exist between the human perception of traits and real traits. Further, human errors differ systematically from a statistical model which generalizes word occurrence patterns. In this section, we directly identify the textual cues that bias humans and cause them to mislabel users.

In addition to unigram analysis, in order to aid interpretability of the feature analysis, we group words into clusters of semantically similar words or *topics* using a method from (Preoţiuc-Pietro et al., 2015b). We first obtain word representations using the popular skip-gram model with negative sampling introduced by Mikolov et al. (2013) and implemented in the Gensim package (layer size 50, context window 5). We train this model on a **separate** reference corpus containing ~ 400 million tweets. After computing the word vectors, we create a word $\times$ word semantic similarity matrix using cosine similarity between the vectors and group the words into clusters using spectral clustering (Shi and Malik, 2000). Each word is only assigned to one cluster. We choose a number of 1,000 topics based on preliminary experiments. Further, we use the NRC Emotion Lexicon (Mohammad and Turney, 2013) to measure eight *emotions* (anger, fear, anticipation, trust, surprise, sadness, joy and disgust) and two *sentiments* (negative and positive). A user's score in each of these 10 dimensions is represented as a weighted sum of its words multiplied by their lexicon score.

7.1 Gender Perception

To study gender perception, we first define a measure of *perceived gender expression*, calculated as the fraction of female guesses out of the 9 guesses for each Twitter user. We then compute univariate correlations the text-derived features and the user

Perceived – Female				**Perceived – Male**			
Topic	**Perc**	**Real**	**Cont**	Topic	**Perc**	**Real**	**Cont**
such, loving, pretty, beautiful, gorgeous	.416	.348	.176	nation, held, rally, defend, supporters	-.372	-.281	-.176
bed, couch, blanket, lying, cozy	.424	.376	.165	players, teams, crowds, athletes, clubs	-.370	-.284	-.171
hair, blonde, shave, eyebrows, dye	.379	.325	.152	training, team, field, coach, career	-.323	-.246	-.148
friend, boyfriend, bf, bff, gf	.365	.308	.149	heat, game, nba, lakers, playoff	-.314	-.237	-.145
girl, lucky, she's, you're, he's	.378	.336	.143	draft, trade, deadline, stat, retire	-.303	-.223	-.143
sweet, angel, honey, pumpkin, bunny	.365	.322	.138	ref, offensive, foul, defensive, refs	-.324	-.255	-.142
cleaning, laundry, packing, dishes, washing	.350	.307	.133	second, third, grade, century, period	-.282	-.195	-.142
awake, dream, sleep, asleep, nights	.327	.276	.130	former, leader, chief, vice, minister	-.316	-.244	-.142
cry, heart, smile, deep, whenever	.331	.288	.125	private, claim, jail, removed, banned	-.299	-.224	-.138
cake, christmas, gift, cupcakes, gifts	.330	.287	.125	war, action, army, battle, zone	-.323	-.263	-.135
evening, day, rest, today, sunday	.249	.180	.118	security, transition, administration, support	-.295	-.225	-.134
light, dark, colors, bright, rainbow	.244	.178	.114	general, major, impact, signs, conflict	-.295	-.227	-.132
shopping, home, spend, packed, grocery	.326	.301	.111	largest, launches, announces, lands, add	-.273	-.196	-.132
dreams, live, forget, remember, along	.247	.194	.107	guns, planes, riot, weapons, soldiers	-.251	-.165	-.131
darling, xo, hugs	.259	.211	.106	title, tech, stats, division, technical	-.314	-.258	-.129
brother, mom, daddy, daughter, sister	.302	.275	.105	breaking, turns, breaks, falls, puts	-.266	-.190	-.128
moment, awkward, laugh, excitement, laughter	.282	.247	.103	million, billion	-.277	-.206	-.128
totally, awesome, favorite, love, fave	.272	.233	.103	steve, joe, dave, larry, phil	-.294	-.236	-.124
breakfast, dinner, lunch, cooking, meal	.280	.245	.103	football, pitch, blues, derby, lineup	-.276	-.211	-.124
makeup, glasses, lipstick	.264	.223	.102	ceo, warren	-.240	-.160	-.123
Unigrams	**Perc**	**Real**	**Cont**	**Unigrams**	**Perc**	**Real**	**Cont**
love,my,so,!,you,I,her,hair,feel,today,	.339	.259	.156	game,the,sports,against,football,teams,	-.270	-.236	-.130
friends,baby,cute,girls,beautiful,me,heart,				player,fans,report,team,ebola,vs,nba,games,			
little,shopping,happy,because,wonderful,			↓	economy,score,government,ceo,americans,			↓
gorgeous,bed,clothes,am,have,yay,your	.179	.081	.071	goals,app,penalties,play,shit,political,war	-.117	-.062	-.065
Emotion	**Perc**	**Real**	**Cont**	**Emotion**	**Perc**	**Real**	**Cont**
Joy	.255	.245	.091	Anger	-.156	-.117	-.076
				Fear	-.183	-.145	-.084

Table 3: Textual features highlighting errors in human perception of gender compared to ground truth labels. Table shows correlation to perceived gender expression (**Perc**), to ground truth (**Real**) and to perceived gender expression controlled for ground truth (**Cont**). All correlations of gender unigrams, topics and emotions are statistically significant at $p < .001$ (t-test)

Gender – High Confidence				**Gender – Low Confidence**			
Topic	Conf	Real	Cont	Topic	Conf	Real	Cont
sibling,flirted,married,husband,wife	(.028)	(.071)	.240	wiser,easier,shittier,happier,worse	-.277	(.081)	-.295
fellaz,boyss,dayz,girlz,gurlz,sistas	(.118)	(.113)	.221	agenda,planning,activities,schedule	-.285	(.020)	-.289
brother, mom, daddy, daughter, sister	(.127)	.241	.214	horoscope,zodiac,gemini,taurus,virgo	-.269	(.087)	-.288
bathroom,wardrobe,toilet,clothes,bath	(.017)	.220	.212	reshape,enable,innovate,enhance,create	-.253	(-.110)	-.235
looked,winked,smiled,lol'd,yell,stare	(.035)	(.089)	.201	imperfect,emotional,break-down,commit	-.227	.024	-.232
hair, blonde, shave, eyebrows, dye	.163	.182	.199	major,brief,outlined,indicates,wrt	-.234	(-.045)	-.226
pyjama,shirt,coat,hoody,trousers	(.077)	(-.010)	.191	justification,circumstance,boundaries	-.224	(-.014)	-.221
awake, dream, sleep, asleep, nights	.160	(.132)	.184	experiencing, explanations, expressive	-.225	(-.039)	-.217
totally, awesome, favorite, love, fave	(.063)	(.135)	. 183	inferiority,sufficiently,adequately	-.209	(-.015)	-.206
days,minutes,seconds,years,months	(.087)	(-.013)	.177	specified,negotiable,exploratory,expert	-.190	(-.014)	-.187
baldy,gangster,boy,kid,skater,dude	(.071)	(.027)	.173	multiple,desirable,extensive,increasingly	-.199	(-.092)	-.183
shopping,grocery,ikea,manicure	(.052)	.204	.173	anticipate,optimist,unrealistic,exceed	(.053)	(.023)	-.182
happy,birthdayyyy,happyyyy,bday	.180	.222	.172	organisation,communication,corporate	-.200	-.148	-.175
girl, lucky, she's, you're, he's	(.118)	(.060)	.172	hostile,choppy,chaotic,cautious,neutral	-.178	(-.033)	-.172
worst,happiest,maddest,slowest,funniest	.173	(.113)	.172	security, transition, administration, supports	.185	(-.079)	-.170
bazillion,shitload,nonstop,spent,aand	.162	(.084)	.167	diminished,unemployment,rapidly	-.181	(-.101)	-.163
Emotion	**Conf**	**Real**	**Cont**	**Emotion**	**Conf**	**Real**	**Cont**
Joy	.202	.245	.164	–			
Anticipation	.140	(.086)	.124				
Unigrams	**Conf**	**Real**	**Cont**	**Unigrams**	**Conf**	**Real**	**Cont**
I,my,this,was,me,so,had,like,	.312	.267	.360	more,may,might,although,	.290	.081	.310
her,night,she,just,hair,gonna,				emotional,your,eager,url,			
ever,last,shirt,			↓	desires,relationship,seem,existing,			↓
kid,girls,love	(.076)	(.047)	.160	emotions,surface,practical,source	.150	-.014	.180

Table 4: Textual features highlighting high and low confidence in human perception of gender. Table shows correlation to average self-reported confidence (**Conf**), to ground truth (**Real**) and with self-reported confidence controlled for ground truth (**Cont**). All correlations of gender unigrams, topics and emotions are statistically significant at $p < .001$ (t-test), except of the values in brackets.

Perceived – Older				Perceived – Younger			
Topic	**Perc**	**Real**	**Cont**	**Topic**	**Perc**	**Real**	**Cont**
golf, sport, semi, racing	.278	(.085)	.226	she's, youre, hes, lucky, girl, slut	-.328	-.243	-.184
bill, union, gov, labor, cuts	.349	.287	.181	boys, girls, hella, homies, ya'll	-.297	-.236	-.155
states, public, towns, area, employees, immigrants	.301	.213	.173	dumb, petty, weak, lame, bc, corny	-.295	-.232	-.155
roger, stanley, captain	.232	(.105)	.167	miss, doing, chilling, how's	-.305	-.268	-.145
available, service, apply, package, customer	.279	.197	.160	heart, cry, smile, deep, hug	-.258	-.186	-.144
serving, prime, serve, served, freeze	.215	(.097)	.154	friend, bestfriend, boyfriend, bff, bestest	-.281	-.254	-.127
support, leaders, group, youth, educate	.228	.121	.153	ugly, stubborn, bein, rude, childish, greedy	-.238	-.182	-.126
hillary, clinton, obama, president, scott, ed, sarah	.289	.230	.150	bitch, fuck, hoe, dick, slap, suck	-.278	-.251	-.125
via, daily, press, latest, report, globe	.311	.272	.149	kinda, annoying, weird, silly, emo, retarded, random	-.242	-.193	-.124
diverse, developed, multiple, among, several, highly	.266	.195	.147	everyone, everything, nothing, does, anyone, else	-.201	-.218	-.118
military, terrorist, citizens, iraq, refugees	.287	.235	.146	bruh, aye, fam, doin, yoo, dawg	-.227	-.178	-.117
julia, emma, annie, claire	.180	(.056)	.145	ever, cutest, worst, weirdest, biggest, happiest	-.275	-.264	-.115
liberty, pacific, north, eastern, 2020	.260	.198	.139	seriously, crazy, bad, shitty, yikes, insane	-.208	-.152	-.114
brooklyn, nyc, downtown, philly, hometown	.213	.120	.139	whoops, oops, remembered, forgot	-.179	(-.104)	-.113
Unigrams	**Perc**	**Real**	**Cont**	**Unigrams**	**Perc**	**Real**	**Cont**
golf, our, end, delay, favourite, low, holes, original,	.321	.(063)	.282	me, i, when, like, you, so, dude, don't, hate, im, u,	-.535	-.489	-.294
branch, the, of, stanley, our, . , story, , ,			↓	girl, hate, life, my, wanna, literally,			↓
forever, exciting, great, what, community, hurricane,				r, really, cute, someone, youre, miss, me , want, this			
for, brands, toward, kids, regarding, upcoming	.208	(.101)	.145	okay, rt, school, snapchat, shit, crying	-.256	(-.051)	-.117
Emotion	**Perc**	**Real**	**Cont**	**Emotion**	**Perc**	**Real**	**Cont**
Positive	.325	.268	.166	Disgust	-.177	-.131	-.094
Trust	.243	.184	.130	Negative	-.104	(-.031)	-.084
Anticipation	.212	.176	.102	Sadness	-.126	-.072	-.081
				Anger	-.070	(-.009)	-.065

Table 5: Textual features highlighting errors in human perception of age compared to ground truth labels. Table shows correlation to perceived age expression (**Perc**), to ground truth (**Real**) and to perceived age expression controlled for ground truth (**Cont**). All correlations of age unigrams, topics and emotions are statistically significant at $p < .001$ (t-test), except of the values in brackets.

Age – High Confidence				Age – Low Confidence			
Topic	**Conf**	**Real**	**Cont**	**Topic**	**Conf**	**Real**	**Cont**
school, student, college, teachers, grad, classroom	.242	(-.054)	.227	mocho, gracias, chicos, corazon, quiero	-.195	(-.042)	-.207
done, homework, finished, essay, procrastinating	.251	-.125	.219	sweepstakes, giveaway, enter, retweet, prize	(-.044)	-.278	-.134
math, chem, biology, test, study, physics	.227	(-.060)	.210	injures, shot, penalty, strikes, cyclist, suffered	-.149	.153	-.108
cant, can't, wait, till, believe, afford	.226	-.171	.183	final, cup, europa, arsenal, match, league	-.135	.107	-.106
tomorrow, friday, saturday, date, starts	.175	(-.014)	.171	juventus, munich, lyon, bayern, 0-1	(-.101)	(-.005)	-.103
invitations, prom, attire, wedding, outfit, gowns	.172	(.005)	.170	castlevania, angels, eagles, demons, flames	-.138	.138	(-.101)
soexcited, next, week, weekend, summer, graduation	.153	(.009)	.155	devil, sword, curse, armor, die, obey	(-.081)	(-.055)	(-.097)
aaand, after, before, literally, off, left, gettingold	.182	(-.103)	.154	football, reds, kickoff, derby, pitch, lineup	-.125	.106	(-.096)
sleepy, work, shifts, longday, exhausted, nap	.126	(.064)	.144	anime, invader, shock, madoka, dragonball	(-.071)	(-.080)	(-.095)
life, daydream, remember, cherish, eternally, reminiscing	.200	-.228	.143	paranormal, dragon, alien, zombie, dead	(-.099)	(.025)	(-.092)
happyyyy, birthdaaaay, b-day, bday, belated	.187	-.173	.142	earthquake, magniture, aftermath, devastating, victims	(-.101)	(.040)	(-.090)
Unigrams	**Conf**	**Real**	**Cont**	**Unigrams**	**Conf**	**Real**	**Cont**
my, i'm, can't, i, school, so, to, class,	.375	-.350	.314	rt, his, league, epic	(-.023)	-.320	-.128
semester, college, homework, prom, me, in my,			↓	warriors, ! ,			↓
friends, literally, when, exam, nap	.180	(.080)	.157	vintage	-.130	(.071)	-.111
Emotion	**Conf**	**Real**	**Cont**	**Emotion**	**Conf**	**Real**	**Cont**
Trust	(.077)	.184	.134	–			
Joy	.125	(.009)	.128				
Positive	(.031)	.268	.115				
Anticipation	(.060)	.176	.114				

Table 6: Textual features highlighting high and low confidence in human perception of age. Table shows correlation to average self-reported confidence (**Conf**), to ground truth (**Real**) and with self-reported confidence controlled for ground truth (**Cont**). Correlation values of age unigrams, topics and emotions statistically significant at $p < .001$ (t-test) unless in brackets.

labels. Table 3 displays the features with significant correlation to perceived gender expression when controlled for real gender using partial correlation, as well as the standalone correlations with the real gender label and perceived gender expression. Note that all correlations with both males and females have the same sign for both perceived gender and real gender. This highlights that humans are *not* wrong in using these features to make gender assessments. Rather, these stereotypical associates are overestimated by humans.

By analyzing the topics that are still correlated with perception after controlling for ground truth correlation, we see that topics related to sports, politics, business and technology are considered by annotators to be stronger cues for predicting males than they really are. Female perception is dominated by topics and words relating to feelings,

shopping, dreaming, housework and beauty. For emotions, joy is perceived to be more associated to females than the data shows, while users expressing more anger and fear are significantly more likely to be perceived as males than the data supports.

Our crowdsourcing experiment allowed annotators to self-report their confidence in each choice. This gives us the opportunity to measure which textual features lead to higher self-reported confidence in predicting user traits. Table 4 shows the textual features most correlated with self-reported confidence of the annotators when controlled for ground truth, in order to account for the effect that overall confidence is on average higher for groups of users that are easier to predict (i.e., females in case of gender, younger people in case of age).

Annotations are most confident when family relationships or other people are mentioned, which aid them to easily assign a label to a user (e.g., 'husband'). Other topics leading to high confidence are related to apparel or beauty. Also the presence of joy leads to higher confidence (for predicting females based on the previous result). Low confidence is associated with work related topics or astrology as well as to clusters of general adverbs and verbs and tentatively, to a more formal vocabulary e.g., 'specified', 'negotiable', 'exploratory'. Intriguingly, low confidence in predicting gender is also related to unigrams like 'emotions', 'relationship', 'emotional'.

7.2 Age Perception

Table 5 displays the features most correlated with perceived age – the average of the 9 annotator guesses – when controlled for real age, and the individual correlations to perceived and real age.

Again, annotators relied on correct stereotypes, but relied on them more heavily than warranted by data. The results show that the perception of users as being older compared to their biological age, is driven by topics including politics, business and news events. Vocabulary contains somewhat longer words (e.g., 'regarding', 'upcoming', 'original'). Additionally, annotators perceived older users to express more positive emotions, trust and anticipation. This is in accordance with psychology research, which showed that both positive emotion (Mather and Carstensen, 2005) and trust (Poulin and Haase, 2015) increase as people get older.

The perception of users being younger than their biological age is highly correlated with the use of short and colloquial words, and self-references, such as the personal pronoun 'I'. Remarkably, the negative sentiment is perceived as more specific of younger users, as well as the negative emotions of disgust, sadness and anger, the later of which is actually uncorrelated to age.

Table 6 displays the features with the highest correlation to annotation confidence in predicting age when controlling for the true age, as well as separate correlations to real and perceived age. Annotators appear to be more confident in their guess when the posts display more joy, positive emotion, trust and anticipation words. In terms of topics mentioned, these are more informal, self-referential or related to school or college. Topics leading to lower confidence are either about sports or online contests or are frequently retweets.

8 Conclusions

This is the first study to systematically analyze differences between real user traits and traits as perceived from text, here Twitter posts. Overall, participants were generally accurate in guessing a person's traits supporting earlier research that stereotypical associations are frequently accurate (McCauley, 1995). However, we have demonstrated that humans use stereotypes which lead to systematic biases by comparing their guesses to predictions from statistical models using the bag-of-words assumption. While qualitatively different, these predictions were shown to offer complimentary information in case of gender, boosting overall accuracy when used jointly.

Our experimental design allowed us to directly test which textual cues lead to inaccurate assessments. Correlation analysis showed that aspects of stereotypes associated with errors tended not to be completely wrong but rather poorly applied. Annotators generally exaggerated the diagnostic utility of behaviors that they correctly associated with one group or another. Further, we used the same methodology to analyze self-reported confidence.

Follow-up studies can analyze the perception of other user traits such as education level, race or political orientation. Another avenue of future research can look at the annotators' own traits and how these relate to perception (Flekova et al., 2015). This would allow to uncover demographic or psychological traits that influence the ability to make more accurate judgements. This is particularly useful in offering task requesters a prior over which annotators are expected to perform tasks better.

Acknowledgments

The authors acknowledge the support from Templeton Religion Trust, grant TRT-0048. The work of the first author was further supported by the German Research Foundation under grant No. GU 798/14-1 and the German Federal Ministry of Education and Research (BMBF) under the promotional reference 01-S12054. We wish to thank Prof. Iryna Gurevych for supporting the collaboration, the Mechanical Turk annotators for their diligence and the reviewers for their thoughtful comments.

References

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender Identity and Lexical Variation in Social Media. *Journal of Sociolinguistics*, 18(2):135–160.

Joseph Bates. 1994. The Role of Emotion in Believable Agents. *Communications of the ACM*, 37(7):122–125.

Amy L Baylor and Yanghee Kim. 2004. Pedagogical Agent Design: The Impact of Agent Realism, Gender, Ethnicity, and Instructional Role. In *Intelligent Tutoring Systems*, volume 3220, pages 592–603.

Philip Bramsen, Martha Escobar-Molano, Ami Patel, and Rafael Alonso. 2011. Extracting Social Power Relationships from Natural Language. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 773–782.

Matthias Braunhofer, Mehdi Elahi, and Francesco Ricci. 2015. User personality and the new user problem in a context-aware point of interest recommender system. In *Information and Communication Technologies in Tourism*, pages 537–549.

D. John Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating Gender on Twitter. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1301–1309.

Subhagata Chattopadhyay, Preetisha Kaur, Fethi Rabhi, and Rajendra Acharya. 2011. An Automated System to Diagnose the Severity of Adult Depression. In *Second International Conference on Emerging Applications of Information Technology*, EAIT, pages 121–124.

Glen Coppersmith, Mark Dredze, and Craig Harman. 2014. Quantifying Mental Health Signals in Twitter. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, ACL, pages 51–60.

Aron Culotta, Nirmal Kumar Ravi, and Jennifer Cutler. 2015. Predicting the Demographics of Twitter Users from Website Traffic Data. In *Proceedings of the 9th International AAAI Conference on Weblogs and Social Media*, ICWSM, pages 72–78.

Munmun De Choudhury, Scott Counts, and Eric Horvitz. 2013. Social Media as a Measurement Tool of Depression in Populations. In *Proceedings of the 5th Annual ACM Web Science Conference*, pages 47–56.

John F Dovidio, John C Brigham, Blair T Johnson, and Samuel L Gaertner. 1996. Stereotyping, Prejudice, and Discrimination: Another Look. *Stereotypes and Stereotyping*, 276:319.

Alice H Eagly. 1995. The Science and Politics of Comparing Women and Men. *American Psychologist*, 50(3):145–158.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2010. A Latent Variable Model for Geographic Lexical Variation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1277–1287.

Lucie Flekova and Iryna Gurevych. 2013. Can We Hide in the Web? Large Scale Simultaneous Age and Gender Author Profiling in Social Media - Notebook for PAN at CLEF 2013. In *CLEF 2013 Labs and Workshops - Online Working Notes*.

Lucie Flekova, Daniel Preoţiuc-Pietro, Jordan Carpenter, Salvatore Giorgi, and Lyle Ungar. 2015. Analyzing Crowdsourced Assessment of User Traits through Twitter Posts. In *Third AAAI Conference on Human Computation and Crowdsourcing*, HCOMP.

Lucie Flekova, Lyle Ungar, and Daniel Preoctiuc-Pietro. 2016. Exploring Stylistic Variation with Age and Income on Twitter. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL.

Goldine C. Gleser, Louis A. Gottschalk, and Watkins John. 1959. The Relationship of Sex and Intelligence to Choice of Words: A Normative Study of Verbal Behavior. *Journal of Clinical Psychology*, 15(2):182–191.

Dirk Hovy and Anders Søgaard. 2015. Tagging Performance Correlates with Author Age. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, ACL, pages 483–488.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual Syntactic Variation over Age and Gender. In *Proceedings of the 19th Conference on Computational Language Learning*, CONNL, pages 103–112.

Oliver P John and Richard W Robins. 1994. Accuracy and Bias in Self-Perception: Individual Differences in Self-enhancement and the Role of Narcissism. *Journal of Personality and Social Psychology*, 66(1):206–219.

Leda E. Kanellakos. 2002. Formal Vocabulary as a Status Cue: Interactions with Diffuse Status Characteristics.

David A. Kenny and Linda Albright. 1987. Accuracy in Interpersonal Perception: A Social Relations Analysis. *Psychological Bulletin*, 102(3):390–402.

Diane Kobrynowicz and Nyla R Branscombe. 1997. Who Considers themselves Victims of Discrimination?: Individual Difference Predictors of Perceived Gender Discrimination in Women and Men. *Psychology of Women Quarterly*, 21(3):347–363.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private Traits and Attributes are Predictable from Digital Records of Human Behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805.

Vasileios Lampos, Nikolaos Aletras, Daniel Preoţiuc-Pietro, and Trevor Cohn. 2014. Predicting and Characterising User Impact on Twitter. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, EACL, pages 405–413.

Vasileios Lampos, Nikolaos Aletras, Jens K. Geyti, Bin Zou, and Ingemar J. Cox. 2016. Inferring the Socioeconomic Status of Social Media Users based on Behaviour and Language. In *Proceedings of the 38th European Conference on Information Retrieval*, ECIR, pages 689–695.

A Bryan Loyall and Joseph Bates. 1997. Personality-rich believable agents that use language. In *First International Conference on Autonomous Agents*, AGENTS, pages 106–113.

Marco Lui and Timothy Baldwin. 2012. Langid.Py: An Off-the-shelf Language Identification Tool. In *Proceedings of the ACL 2012 System Demonstrations*, ACL, pages 25–30.

Mara Mather and Laura L Carstensen. 2005. Aging and Motivated Cognition: The Positivity Effect in Attention and Memory. *Trends in Cognitive Sciences*, 9(10):496–502.

Clark R. McCauley. 1995. Are Stereotypes Exaggerated? A Sampling of Racial, Gender, Academic, Occupational, and Political Stereotypes. *Stereotype accuracy: Toward appreciating group differences*, pages 215–243.

Allen R McConnell and Russell H Fazio. 1996. Women as Men and People: Effects of Gender-marked Language. *Personality and Social Psychology Bulletin*, 22(10).

Julie R. McMillan, A. Kay Clifton, Diane McGrath, and Wanda S. Gale. 1977. Women's language: Uncertainty or interpersonal sensitivity and emotionality? *Sex Roles*, 3(6):545–559.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *Proceedings of Workshop at the International Conference on Learning Representations*, ICLR, pages 1–12.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3):436–465.

Anthony Mulac, Lisa B. Studley, and Sheridan Blau. 1990. The Gender-linked Language Effect in Primary and Secondary Students' Impromptu Essays. *Sex Roles*, 23(9-10):439–470.

Matthew L Newman, Carla J Groom, Lori D Handelman, and James W Pennebaker. 2008. Gender Differences in Language Use: An Analysis of 14,000 Text Samples. *Discourse Processes*, 45(3):211–236.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. 'How Old do you Think I am?'; A Study of Language and Age in Twitter. In *Proceedings of the 7th International AAAI Conference on Weblogs and Social Media*, ICWSM, pages 439–448.

Dong-Phuong Nguyen, RB Trieschnigg, AS Doğruöz, Rilana Gravel, Mariët Theune, Theo Meder, and FMG de Jong. 2014. Why Gender and Age Prediction from Tweets is Hard: Lessons from a Crowdsourcing Experiment. In *Proceedings of the 25th International Conference on Computational Linguistics*, COLING, pages 1950–1961.

James W. Pennebaker, Martha E. Francis, and Roger J. Booth. 2001. *Linguistic Inquiry and Word Count*.

James W. Pennebaker, Matthias R. Mehl, and Kate G. Niederhoffer. 2003. Psychological Aspects of Natural Language Use: Our Words, our Selves. *Annual Review of Psychology*, 54(1):547–577.

Michael Poulin and Claudia Haase. 2015. Growing to Trust. Evidence That Trust Increases and Sustains Well-Being Across the Life Span. *Social Psychological and Personality Science*, 6(6):614–621.

Daniel Preoţiuc-Pietro, Johannes Eichstaedt, Gregory Park, Maarten Sap, Laura Smith, Victoria Tobolsky, H Andrew Schwartz, and Lyle H Ungar. 2015a. The Role of Personality, Age and Gender in Tweeting about Mental Illnesses. In *Proceedings of the Workshop on Computational Linguistics and Clinical Psychology: From Linguistic Signal to Clinical Reality*, NAACL, pages 21–30.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015b. An Analysis of the User Occupational Class through Twitter Content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL, pages 1754–1764.

Daniel Preoţiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015c. Studying User Income through Language, Behaviour and Affect in Social Media. *PLoS ONE*, 10(9).

Daniel Preoţiuc-Pietro, Wei Xu, and Lyle Ungar. 2016. Discovering User Attribute Stylistic Differences via Paraphrasing. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI, pages 3030–3037.

Kimberly Quinn and Neil Macrae. 2005. Categorizing Others: The Dynamics of Person Construal. *Journal of Personality and Social Psychology*, 88(3):467–479.

Francisco Rangel, Paolo Rosso, Irina Chugur, Martin Potthast, Martin Trenkmann, Benno Stein, Ben Verhoeven, and Walter Daelemans. 2014. Overview of the 2nd Author Profiling Task at PAN 2014. In *Proceedings of the Conference and Labs of the Evaluation Forum (Working Notes)*, CLEF.

Francisco Rangel, Paolo Rosso, Martin Potthast, Benno Stein, and Walter Daelemans. 2015. Overview of the 3rd Author Profiling Task at PAN 2015. In *Proceedings of the Conference and Labs of the Evaluation Forum (Working Notes)*, CLEF.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying Latent User Attributes in Twitter. In *Proceedings of the 2nd International Workshop on Search and Mining User-generated Contents*, SMUC, pages 37–44.

Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1650–1659.

Rinat B Rosenberg-Kima, Amy L Baylor, E Ashby Plant, and Celeste E Doerr. 2008. Interface Agents as Social Models for Female Students: The Effects of Agent Visual Presence and Appearance on Female Students' Attitudes and Beliefs. *Computers in Human Behavior*, 24(6):2741–2756.

Sara Rosenthal and Kathleen McKeown. 2011. Age Prediction in Blogs: A Study of Style, Content, and Online Behavior in pre- and post-Social Media Generations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, ACL, pages 763–772.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, Lyle Ungar, and H Andrew Schwartz. 2014. Developing Age and Gender Predictive Lexica over Social Media. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP, pages 1146–1151.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James W Pennebaker. 2006. Effects of Age and Gender on Blogging. AAAI Spring Symposium, pages 199–205.

H Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Stephanie M Ramones, Megha Agrawal, Achal Shah, Michal Kosinski, David Stillwell, Martin EP Seligman, and Lyle H Ungar. 2013. Personality, Gender, and Age in the Language of Social Media: The Open-Vocabulary Approach. *PLoS ONE*, 8(9).

Suranga Seneviratne, Aruna Seneviratne, Prasant Mohapatra, and Anirban Mahanti. 2015. Your Installed Apps Reveal your Gender and More! *ACM SIGMOBILE Mobile Computing and Communications Review*, 18(3):55–61.

Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905.

Patrick E. Shrout and Joseph L. Fleiss. 1979. Intraclass Correlations: Uses in Assessing Rater Reliability. *Psychological Bulletin*, 86:420–428.

Robert Tibshirani. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58:267–288.

Svitlana Volkova and Yoram Bachrach. 2015. On Predicting Sociodemographic Traits and Emotions from Communications in Social Networks and Their Implications to Online Self-Disclosure. *Cyberpsychology, Behavior, and Social Networking*, 18(12):726–736.

Svitlana Volkova and Yoram Bachrach. 2016. Inferring Perceived Demographics from User Emotional Tone and User-Environment Emotional Contrast. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring User Political Preferences from Streaming Communications. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, ACL, pages 186–196.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring Latent User Properties from Texts Published in Social Media. In *Proceedings of the Twenty-Ninth Conference on Artificial Intelligence (Demo)*, AAAI, pages 4296–4297.

Wu Youyou, Michal Kosinski, and David Stillwell. 2015. Computer-based Personality Judgments are more Accurate than those Made by Humans. *PNAS*, 112(4):1036–1040.

Modeling Social Norms Evolution for Personalized Sentiment Classification

Lin Gong[1], **Mohammad Al Boni**[2], **Hongning Wang**[1]
[1]Department of Computer Science, [2]Department of System and Information Engineering
University of Virginia, Charlottesville VA, 22904 USA
{lg5bt, ma2sm, hw5x}@virginia.edu

Abstract

Motivated by the findings in social science that people's opinions are diverse and variable while together they are shaped by evolving social norms, we perform personalized sentiment classification via *shared model adaptation* over time. In our proposed solution, a global sentiment model is constantly updated to capture the homogeneity in which users express opinions, while personalized models are simultaneously adapted from the global model to recognize the heterogeneity of opinions from individuals. Global model sharing alleviates data sparsity issue, and individualized model adaptation enables efficient online model learning. Extensive experimentations are performed on two large review collections from Amazon and Yelp, and encouraging performance gain is achieved against several state-of-the-art transfer learning and multi-task learning based sentiment classification solutions.

1 Introduction

Sentiment is personal; the same sentiment can be expressed in various ways and the same expression might carry distinct polarities across different individuals (Wiebe et al., 2005). Current mainstream solutions of sentiment analysis overlook this fact by focusing on population-level models (Liu, 2012; Pang and Lee, 2008). But the idiosyncratic and variable ways in which individuals communicate their opinions make a global sentiment classifier incompetent and consequently lead to suboptimal opinion mining results. For instance, a shared statistical classifier can hardly recognize that in restaurant reviews, the word "expensive" may indicate some users' satisfaction with a restaurant's quality, although it is generally associated with negative attitudes. Hence, a personalized sentiment classification solution is required to achieve fine-grained understanding of individuals' distinctive and dynamic opinions and benefit downstream opinion mining applications.

Sparse observations of individuals' opinionated data (Max, 2014) prevent straightforward solutions from building personalized sentiment classification models, such as estimating supervised classifiers on a per-user basis. Semi-supervised methods are developed to address the data sparsity issue. For example, leveraging auxiliary information from user-user and user-document relations in transductive learning (Hu et al., 2013; Tan et al., 2011). However, only one global model is estimated there, and the details of how individual users express diverse opinions cannot be captured. More importantly, existing solutions build static sentiment models on historic data; but the means in which a user expresses his/her opinion is changing over time. To capture temporal dynamics in a user's opinions with existing solutions, repeated model reconstruction is unavoidable, albeit it is prohibitively expensive. As a result, personalized sentiment analysis requires effective exploitation of users' own opinionated data and efficient execution of model updates across all users.

To address these challenges, we propose to build personalized sentiment classification models via *shared model adaptation*. Our solution roots in the social psychology theories about humans' dispositional tendencies (Briley et al., 2000). Humans' behaviors are shaped by social norms, a set of socially shared "feelings" and "display rules" about how one should feel and express opinions (Barsäde and Gibson, 1998; Sherif, 1936). In the context of content-based sentiment classification, we interpret social norms as global model sharing and adaptation across users. Formally, we assume a global sentiment model serves as the basis to capture self-enforcing sentimental regulari-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 855–865,
Berlin, Germany, August 7-12, 2016.

ties across users, and each individual user tailors the shared model to realize his/her personal preference. In addition, social norms also evolve over time (Ehrlich and Levin, 2005), which leads to shifts in individuals' behaviors. This can again be interpreted as model adaptation: a new global model is adapted from an existing one to reflect the newly adopted sentimental norms. The temporal changes in individuals' opinions can be efficiently captured via online model adaptation at the levels of both global and personalized models.

Our proposed solution can also be understood from the perspective of multi-task learning (Evgeniou and Pontil, 2004; Jacob et al., 2009). Intuitively, personalized model adaptations can be considered as a set of related tasks in individual users, which contribute to a shared global model adaptation. In particular, we assume the distinct ways in which users express their opinions can be characterized by a linear classifier's parameters, i.e., the weights of textual features. Personalized models are thus achieved via a series of linear transformations over a globally shared classifier's parameters (Wang et al., 2013), e.g., shifting and scaling the weight vector. This globally shared classifier itself is obtained via another set of linear transformations over a given base classifier, which can be estimated from an isolated collection beforehand and serves as a prior for shared sentiment classification. The shared global model adaptation makes personalized model estimation no longer independent, such that regularity is formed across individualized learning tasks.

We empirically evaluated the proposed solution on two large collections of reviews, i.e., Amazon and Yelp reviews. Extensive experiment results confirm its effectiveness: the proposed method outperformed user-independent classification methods, several state-of-the-art model adaption methods, and multi-task learning algorithms.

2 Related Work

Text-based sentiment classification forms the foundation of sentiment analysis (Liu, 2012; Pang and Lee, 2008). There are two typical types of studies in sentiment classification. The first is classifying input text units (such as documents, sentences and phrases) into predefined categories, e.g., positive v.s., negative (Pang et al., 2002; Gao et al., 2014) and multiple classes (Pang and Lee, 2005). Both lexicon-based and learning-based solutions have been explored. The second is identifying topical aspects and corresponding opinions, e.g., developing topic models to predict fine-grained aspect ratings (Titov and McDonald, 2008; Wang et al., 2011). However, all those works emphasize population-level analysis, which applies a global model on all users and therefore fails to recognize the heterogeneity in which different users express their diverse opinions.

Our proposed solution is closely related to multi-task learning, which exploits the relatedness among multiple learning tasks to benefit each single task. Tasks can be related in various ways. A typical assumption is that all learnt models are close to each other in some matrix norms (Evgeniou and Pontil, 2004; Jacob et al., 2009). This has been empirically proved to be effective for capturing preferences of individual users (Evgeniou et al., 2007). Task relatedness has also been imposed via constructing a common underlying representation across different tasks (Argyriou et al., 2008; Evgeniou and Pontil, 2007). Our solution postulates task relatedness via a two-level model adaptation procedure. The global model adaptation accounts for the homogeneity and shared dynamics in users' opinions; and personalized model adaptation realizes heterogeneity in individual users.

The idea of model adaptation has been extensively explored in the context of transfer learning (Pan and Yang, 2010), which focuses on applying knowledge gained while solving one problem to different but related problems. In opinion mining community, transfer learning is mostly exploited for domain adaptation, e.g., adapting sentiment classifiers trained on book reviews to DVD reviews (Blitzer et al., 2006; Pan et al., 2010). Personalized model adaptation has also been studied in literature. The idea of linear transformation based model adaptation is introduced in (Wang et al., 2013) for personalized web search. Al Boni et al. applied a similar idea to achieve personalized sentiment classification (Al Boni et al., 2015). (Li et al., 2010) developed an online learning algorithm to continue training personalized classifiers based on a given global model. However, all of these aforementioned solutions perform model adaptation from a fixed global model, such that the learning of personalized models is independent from each other. Data sparsity again is the major bottleneck for such solutions. Our solution associates individual model adaptation via a shared global model adaptation, which leverages observations across users and thus reduces preference learning complexity.

3 Methodology

We propose to build personalized sentiment classifiers via shared model adaptation for both a global sentiment model and individualized models. Our solution roots in the social psychology theories about humans' dispositional tendencies, e.g., social norms and the evolution of social norms over time. In the following discussions, we will first briefly discuss the social theories that motivate our research, and then carefully describe the model assumptions and technical details about the proposed personalized model adaptation solution.

3.1 The Evolution of Social Norms

Social norms create pressures to establish socialization of affective experience and expression (Shott, 1979). Within the limit set by social norms and internal stimuli, individuals construct their sentiment, which is not automatic, physiological consequences but complex consequences of learning, interpretation, and social influence. This motivates us to build a global sentiment classification model to capture the shared basis on which users express their opinions. For example, the phrase "a waste of money" generally represents negative opinions across all users; and it is very unlikely that anybody would use it in a positive sense. On the other hand, members of some segments of a social structure tend to feel certain emotions more often or more intensely than members of other segments (Hochschild, 1975). Personalized model adaptation from the shared global model becomes necessary to capture the variability in affective expressions across users. For example, the word "expensive" may indicate some users' satisfaction with their received service.

Studies in social psychology also suggest that social norms shift and spread through infectious transfer mediated by webs of contact and influence over time (Ostrom, 2014; Ehrlich and Levin, 2005). Members inside a social structure influence the other members; confirmation of shifted beliefs leads to the development and evolution of social norms, which in turn regulate the shared social behaviors as a whole over time. The evolving nature of social norms urges us to take a dynamic view of the shared global sentiment model: instead of treating it as fixed, we further assume this model is also adapted from a predefined one, which serves as prior for sentiment classification. All individual users are coupled and contribute to this shared global model adaptation. This two-level model adaptation assumption leads us to the proposed multi-task learning solution, which will be carefully discussed in the next section.

3.2 Shared Linear Model Adaptation

In this paper, we focus on linear models for personalized sentiment classification due to their empirically superior performance in text-based sentiment analysis (Pang et al., 2002; Pang and Lee, 2005). We assume the diverse ways in which users express their opinions can be characterized by different settings of a linear model's parameters, i.e., the weights of textual features.

Formally, we denote a given set of opinionated text documents from user u as $D^u = \{(x_d^u, y_d^u)\}_{d=1}^{|D^u|}$, where each document x_d^u is represented by a V-dimensional vector of textual features and y_d^u is the corresponding sentiment label. The task of personalized sentiment classification is to estimate a personalized model $y = f^u(x)$ for user u, such that $f^u(x)$ best captures u's opinions in his/her generated text content. Instead of assuming $f^u(x)$ is solely estimated from user u's own opinionated data, which is prone to overfitting, we assume it is derived from a globally shared sentiment model $f^s(x)$ via model adaptation (Al Boni et al., 2015; Wang et al., 2013), i.e., shifting and scaling $f^s(x)$'s parameters for each individual user. To simplify the following discussions, we will focus on binary classification, i.e., $y_d \in \{0, 1\}$, and use the logistic regression as our reference model. But the developed techniques are general and can be easily extended to multi-class classification and generalized linear models.

We only consider scaling and shifting operations, given rotation requires to estimate much more free parameters (i.e., $O(V^2)$ v.s., $O(V)$) but contributes less in final classification performance (Al Boni et al., 2015). We further assume the adaptations can be performed in a group-wise manner (Wang et al., 2013): features in the same group will be updated synchronously by enforcing the same shifting and scaling operations. This enables the observations from seen features to be propagated to unseen features in the same group during adaptation. Various feature grouping methods have been explored in (Wang et al., 2013).

Specifically, we define $g(i) \rightarrow j$ as a feature grouping method, which maps feature i in $\{1, 2, \ldots, V\}$ to feature group j in $\{1, 2, \ldots, K\}$. A personalized model adaptation matrix can then be represented as a $2K$-dimensional vector $A^u = (a_1^u, a_2^u, \ldots, a_K^u, b_1^u, b_2^u, \ldots, b_K^u)$, where a_k^u and b_k^u

represent the scaling and shifting operations in feature group k for user u accordingly. Plugging this group-wise model adaptation into the logistic function, we can get a personalized logistic regression model $P^u(y_d = 1|x_d)$ for user u as follows,

$$P^u(y_d = 1|x_d) = \frac{1}{1 + e^{-\sum_{k=1}^{K} \sum_{g(i)=k} (a_k^u w_i^s + b_k^u) x_i}} \quad (1)$$

where w^s is the feature weight vector in the global model $f^s(x)$. As a result, personalized model adaptation boils down to identifying the optimal model transformation operation A^u for each user based on w^s and D^u.

In (Al Boni et al., 2015; Wang et al., 2013), $f^s(x)$ is assumed to be given and fixed. It leads to isolated estimation of personalized models. Based on the social norms evolution theory, $f^s(x)$ should also be dynamic and ever-changing to reflect shifted social norms. Hence, we impose another layer of model adaptation on top of the shared global sentiment model $f^s(x)$, by assuming itself is also adapted from a predefined base sentiment model. Denote this base classifier as $f^0(x)$, which is parameterized by a feature weight vector w^0 and serves as a prior for sentiment classification. Then w^s can be derived via the same aforementioned model adaptation procedure: $w^s = A^s \tilde{w}^0$, where $\tilde{w}^0$ is an augmented vector of w^0, i.e., $\tilde{w}^0 = (w^0, 1)$, to facilitate shifting operations, and A^s is the adaptation matrix for the shared global model. We should note A^s can take a different configuration (i.e., feature groupings) from individual users' adaptation matrices.

Putting these two levels of model adaptation together, a personalized sentiment classifier is achieved via,

$$w^u = A^u A^s \tilde{w}^0 \quad (2)$$

which can then be plugged into Eq (1) for personalized sentiment classification.

We name this resulting algorithm as Mutli-Task Linear Model Adaptation, or *MT-LinAdapt* in short. The benefits of shared model adaptation defined in Eq (2) are three folds. First, the homogeneity in which users express their diverse opinions are captured in the jointly estimated sentiment model $f^s(x)$ across users. Second, the learnt individual models are coupled together to reduce preference learning complexity, i.e., they collaboratively serve to reduce the models' overall prediction error. Third, non-linearity is achieved via the two-level model adaptation, which introduces more flexibility in capturing heterogeneity in different users' opinions. In-depth discussions of those unique benefits will be provided when we introduce the detailed model estimation methods.

3.3 Joint Model Estimation

The ideal personalized model adaptation should be able to adjust the individualized classifier $f^u(x)$ to minimize misclassification rate on each user's historical data in D^u. In the meanwhile, the shared sentiment model $f^s(x)$ should serve as the basis for each individual user to reduce the prediction error, i.e., capture the homogeneity. These two related objectives can be unified under a joint optimization problem.

In logistic regression, the optimal adaptation matrix A^u for an individual user u, together with A^s can be retrieved by a maximum likelihood estimator (i.e., minimizing logistic loss on a user's own opinionated data). The log-likelihood function in each individual user is defined as,

$$L(A^u, A^s) = \sum_{d=1}^{|D^u|} \Big[y_d \log P^u(y_d = 1|x_d) + (1 - y_d) \log P^u(y_d = 0|x_d) \Big] \quad (3)$$

To avoid overfitting, we penalize the transformations which increase the discrepancy between the adapted model and its source model (i.e., between w^u and w^s, and between w^s and w^0) via a L2 regularization term,

$$R(A) = \frac{\eta_1}{2} ||\boldsymbol{a} - 1||_2 + \frac{\eta_2}{2} ||\boldsymbol{b}||_2 \quad (4)$$

and it enforces scaling to be close to one and shifting to be close to zero.

By defining a new model adaptation matrix $\mathring{A} = \{A^{u_1}, A^{u_2}, \ldots, A^{u_N}, A^s\}$ to include all unknown model adaptation parameters for individual users and shared global model, we can formalize the joint optimization problem in MT-LinAdapt as,

$$\max L(\mathring{A}) = \sum_{i=1}^{N} \Big[L(A^{u_i}) - R(A^{u_i}) \Big] - R(A^s) \quad (5)$$

which can be efficiently solved by a gradient-based optimizer, such as quasi-Newton method (Zhu et al., 1997).

Direct optimization over $\mathring{A}$ requires synchronization among all the users. But in practice, users will generate their opinionated data with different paces, such that we have to postpone model adaptation until all the users have at least one observation to update their own adaptation matrix.

This delayed model update is at high risk of missing track of active users' recent opinion changes, but timely prediction of users' sentiment is always preferred. To monitor users' sentiment in realtime, we can also estimate MT-LinAdapt in an asynchronized manner: whenever there is a new observation available, we update the corresponding user's personalized model together with the shared global model immediately. i.e., online optimization of MT-LinAdapt.

This asychronized estimation of MT-LinAdapt reveals the insight of our two-level model adaptation solution: the immediate observations in user u will not only be used to update his/her own adaptation parameters in A^u, but also be utilized to update the shared global model, thus to influence the other users, who do not have adaptation data yet. Two types of competing force drive the adaptation among all the users: $w_s = A^s \tilde{w}_0$ requires timely update of global model across users; and $w_u = A^u w_s$ enforces the individual user to conform to the newly updated global model. This effect can be better understood with the actual gradients used in this asychronized update. We illustrate the decomposed gradients for scaling operation in A^u and A^s from the log-likelihood part in Eq (5) on a specific adaptation instance (x_d^u, y_d^u):

$$\frac{\partial L(A^u,A^s)}{\partial a_k^u} = \Delta_d^u \sum_{g^u(i)=k} \left(a^s_{g^s(i)} w_i^0 + b^s_{g^s(i)} \right) x^u_{di} \quad (6)$$

$$\frac{\partial L(A^u,A^s)}{\partial a_l^s} = \Delta_d^u \sum_{g^s(i)=l} a^u_{g^u(i)} w_i^0 x^u_{di} \quad (7)$$

where $\Delta_d^u = y_d^u - P^u(y_d^u = 1|x_d^u)$, and $g^u(\cdot)$ and $g^s(\cdot)$ are feature grouping functions in individual user u and shared global model $f^s(x)$.

As stated in Eq (6) and (7), the update of scaling operation in the shared global model and individual users depends on each other; the gradient with respect to global model adaptation will be accumulated among all the users. As a result, all users are coupled together via the global model adaptation in MT-LinAdapt, such that model update is propagated through users to alleviate data sparsity issue in each single user. This achieves the effect of multi-task learning. The same conclusion also applies to the shifting operations.

It is meaningful for us to compare our proposed MT-LinAdapt algorithm with those discussed in the related work section. Different from the model adaptation based personalized sentiment classification solution proposed in (Al Boni et al., 2015), which treats the global model as fixed, MT-LinAdapt adapts the global model to capture the evolving nature of social norms. As a result, in (Al Boni et al., 2015) the individualized model adaptations are independent from each other; but in MT-LinAdapt, the individual learning tasks are coupled together to enable observation sharing across tasks, i.e., multi-task learning. Additionally, as illustrated in Eq (6) and (7), nonlinear model adaptation is achieved in MT-LinAdapt because of the different feature groupings in individual users and global model. This enables observations sharing across different feature groups, while in (Al Boni et al., 2015) observations can only be shared within the same feature group, i.e., linear model adaptation. Multi-task SVM introduced in (Evgeniou and Pontil, 2004) can be considered as a special case of MT-LinAdapt. In Multi-task SVM, only shifting operation is considered in individual users and the global model is simply estimated from the pooled observations across users. Therefore, only linear model adaptation is achieved in Multi-task SVM and it cannot leverage prior knowledge conveyed in a predefined sentiment model.

4 Experiments

In this section, we perform empirical evaluations of the proposed MT-LinAdapt model. We verified the effectiveness of different feature groupings in individual users' and shared global model adaptation by comparing our solution with several state-of-the-art transfer learning and multi-task learning solutions for personalized sentiment classification, together with some qualitative studies to demonstrate how our model recognizes users' distinct expressions of sentiment.

4.1 Experiment Setup

• **Datesets.** We evaluated the proposed model on two large collections of review documents, i.e., Amazon product reviews (McAuley et al., 2015) and Yelp restaurant reviews (Yelp, 2016). Each review document contains a set of attributes such as author ID, review ID, timestamp, textual content, and an opinion rating in discrete five-star range. We applied the following pre-processing steps on both datasets: 1) filtered duplicated reviews; 2) labeled reviews with overall rating above 3 stars as positive, below 3 stars as negative, and removed the rest; 3) removed reviewers who posted more than 1,000 reviews and those whose positive review ratio is more than 90% or less than 10%

(little variance in their opinions and thus easy to classify). Since such users can be easily captured by the base model, the removal emphasizes comparisons on adapted models; 4) sorted each user's reviews in chronological order. Then, we performed feature selection by taking the union of top unigrams and bigrams ranked by Chi-square and information gain metrics (Yang and Pedersen, 1997), after removing a standard list of stopwords and porter stemming. The final controlled vocabulary consists of 5,000 and 3,071 textual features for Amazon and Yelp datasets respectively; and we adopted TF-IDF as the feature weighting scheme. From the resulting data sets, we randomly sampled 9,760 Amazon reviewers and 11,733 Yelp reviewers for testing purpose. There are 105,472 positive reviews and 37,674 negative reviews in the selected Amazon dataset; 108,105 positive reviews and 32,352 negative reviews in the selected Yelp dataset.

• **Baselines.** We compared the performance of MT-LinAdapt against seven different baselines, ranging from user-independent classifiers to several state-of-the-art model adaption methods and multi-task learning algorithms. Due to space limit, we will briefly discuss the baseline models below.

Our solution requires a user-independent classifier as base sentiment model for adaptation. We estimated logistic regression models from a separated collection of reviewers outside the preserved testing data on Amazon and Yelp datasets accordingly. We also included these isolated base models in our comparison and name them as *Base*. In order to verify the necessity of personalized sentiment models, we trained a global SVM based on the pooled adaptation data from all testing reviewers, and name it as *Global SVM*. We also estimated an independent SVM model for each single user only based on his/her adaptation reviews, and name it as *Individual SVM*. We included an instance-based transfer learning method (Brighton and Mellish, 2002), which considers the k-nearest neighbors of each testing review document from the isolated training set for personalized model training. As a result, for each testing case, we estimated an independent classification model, which is denoted as *ReTrain*. (Geng et al., 2012) used L2 regularization to enforce the adapted models to be close to the global model. We applied this method to get personalized logistic regression models and refer to it as *RegLR*. *LinAdapt* developed in (Al Boni et al., 2015) also performs group-wise linear model adaptation to build personalization classifiers. But it isolates model adaptation in individual users. *MT-SVM* is a multi-task learning method, which encodes task relatedness via a shared linear kernel (Evgeniou and Pontil, 2004).

• **Evaluation Settings.** We evaluated all the models with both synchronized (batch) and asynchronized (online) model update. We should note MT-SVM can only be tested in batch mode, because it is prohibitively expensive to retrain SVM repeatedly. In batch evaluation, we split each user's reviews into two sets: the first 50% for adaptation and the rest 50% for testing. In online evaluation, once we get a new testing instance, we first evaluate the up-to-date personalized classifier against the ground-truth; then use the instance to update the personalized model. To simulate the real-world situation where user reviews arrive sequentially and asynchronously, we ordered all reviews chronologically and accessed them one at a time for online model update. In particular, we utilized stochastic gradient descent for this online optimization (Kiwiel, 2001). Because of the biased class distribution in both datasets, we computed F1 measure for both positive and negative class in each user, and took macro average among users to compare the different models' performance.

4.2 Effect of Feature Grouping

In MT-LinAdapt, different feature groupings can be postulated in individual users' and shared global model adaptation. Nonlinearity is introduced when different grouping functions are used in these two levels of model adaptation. Therefore, we first investigated the effect of feature grouping in MT-LinAdapt.

We adopted the feature grouping method named "*cross*" in (Wang et al., 2013) to cluster features into different groups. More specifically, we evenly spilt the training collection into N non-overlapping folds, and train a single SVM model on each fold. Then, we create a $V \times N$ matrix by putting the learned weights from N folds together, on which k-means clustering is applied to extract K feature groups. We compared the batch evaluation performance of varied combinations of feature groups in MT-LinAdapt. The experiment results are demonstrated in Table 1; and for comparison purpose, we also included the base classifier's performance in the table.

In Table 1, the two numbers in the first column denote the feature group sizes in personalized models and global model respectively. And *all* indicates one feature per group (i.e., no fea-

Table 1: Effect of different feature groupings in MT-LinAdapt.

Method	Amazon		Yelp	
	Pos F1	Neg F1	Pos F1	Neg F1
Base	0.8092	0.4871	0.7048	0.3495
400-800	0.8318	0.5047	0.8237	0.4807
400-1600	0.8385	0.5257	0.8309	0.4978
400-all	0.8441	0.5423	0.8345	0.5105
800-800	0.8335	0.5053	0.8245	0.4818
800-1600	0.8386	0.5250	0.8302	0.4962
800-all	0.8443	**0.5426**	**0.8361**	**0.5122**
1600-all	**0.8445**	0.5424	0.8357	0.5106
all-all	0.8438	0.5416	0.8361	0.5100

ture grouping). The adapted models in MT-LinAdapt achieved promising performance improvement against the base sentiment classifier, especially on the Yelp data set. As we increased the feature group size for global model, MT-LinAdapt's performance kept improving; while with the same feature grouping in the shared global model, a moderate size of feature groups in individual users is more advantageous.

These observations are expected. Because the global model is shared across users, all their adaptation reviews can be leveraged to adapt the global model so that sparsity is no longer an issue. Since more feature groups in the global model can be afforded, more accurate estimation of adaptation parameters can be achieved. But at the individual user level, data sparsity is still the bottleneck for accurate adaptation estimation, and trade-off between observation sharing and estimation accuracy has to be made. Based on this analysis, we selected **800** and **all** feature groups for individual models and global model respectively in the following experiments.

4.3 Personalized Sentiment Classification

• **Synchronized model update.** Table 2 demonstrated the classification performance of MT-LinAdapt against all baselines on both Amazon and Yelp datasets, where binomial tests on win-loss comparison over individual users were performed between the best algorithm and runner-up to verify the significance of performance improvement. We can clearly notice that MT-LinAdapt significantly outperformed all baselines in negative class, and it was only slightly worse than MT-SVM on positive class. More specifically, per-user classifier estimation clearly failed to obtain a usable classifier, due to the sparse observations in single users. Model-adaptation based baselines, i.e., RegLR and LinAdapt, slightly improved over the base model. But because the adaptations across users are isolated and the base model is fixed, their improvement is very limited. As for negative class, MT-LinAdapt outperformed Global SVM significantly on both datesets. Since negative class suffers more from the biased prior distribution, the considerable performance improvement indicates effectiveness of our proposed personalized sentiment classification solution. As for positive class, the performance difference is not significant between MT-LinAdapt and MT-SVM on Amazon data set nor between MT-LinAdapt and Global SVM on Yelp data set. By looking into detailed results, we found that MT-LinAdapt outperformed MT-SVM on users with fewer adaptation reviews. Furthermore, though MT-SVM benefits from multi-task learning, it cannot leverage information from the given base classifier. Considering the biased class prior in these two data sets (2.8:1 on Amazon and 3.3:1 on Yelp), the improved classification performance on negative class from MT-LinAdapt is more encouraging.

Table 2: Classification results in batch mode.

Method	Amazon		Yelp	
	Pos F1	Neg F1	Pos F1	Neg F1
Base	0.8092	0.4871	0.7048	0.3495
Global SVM	0.8352	0.5403	**0.8411**	0.5007
Individual SVM	0.5582	0.2418	0.3515	0.3547
ReTrain	0.7843	0.4263	0.7807	0.3729
RegLR	0.8094	0.4896	0.7103	0.3566
LinAdapt	0.8091	0.4894	0.7107	0.3575
MT-SVM	**0.8484**	0.5367	0.8408	0.5079
MT-LinAdapt	0.8441	**0.5422***	0.8358	**0.5119***

$*$ indicates p-value$<$0.05 with Binomial test.

• **Asynchronized model update.** In online model estimation, classifiers can benefit from immediate update, which provides a feasible solution for timely sentiment analysis in large datasets. In this setting, only two baseline models are applicable without model reconstruction, i.e., RegLR and LinAdapt. To demonstrate the utility of online update in personalized sentiment models, we illustrate the relative performance gain of these models over the base sentiment model in Figure 1. The x-axis indicates the number of adaptation instances consumed in online update from all users, i.e., the 1st review means after collecting the first review of each user.

MT-LinAdapt converged to satisfactory performance with only a handful of observations in each user. LinAdapt also quickly converged, but its performance was very close to the base model, since no observation is shared across users. RegLR needs the most observations to estimate satisfac-

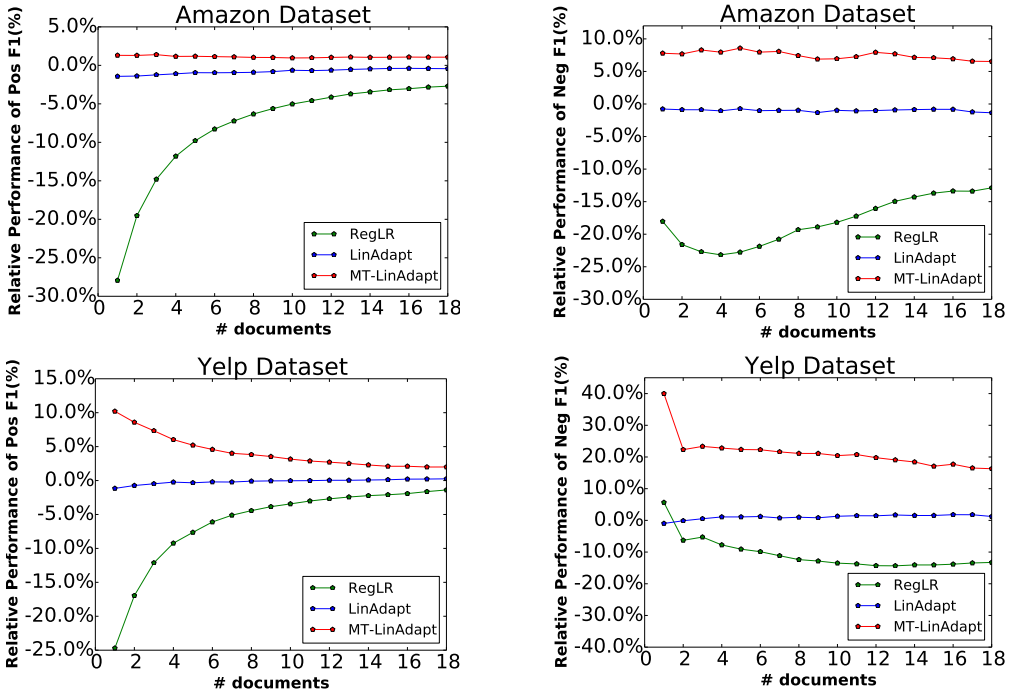


Figure 1: Relative performance gain between MT-LinAdapt and baselines on Amazon and Yelp datasets.

tory personalized models. The improvement in MT-LinAdapt demonstrates the benefit of shared model adaptation, which is vital when the individuals' adaptation data are not immediately available but timely sentiment classification is required.

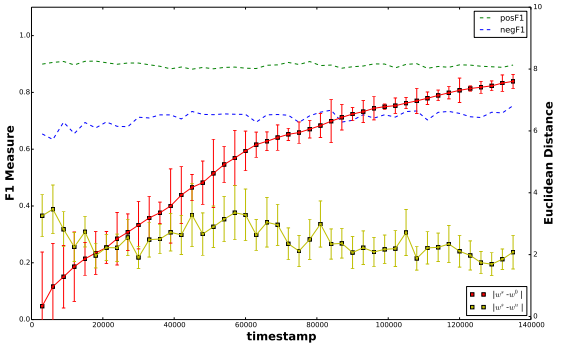


Figure 2: Online model update trace on Amazon.

It is meaningful to investigate how the shared global model and personalized models are updated during online learning. The shift in the shared global model reflects changes in social norms, and the discrepancy between the shared global model and personalized models indicates the variances of individuals' opinions. In particular, we calculated Euclidean distance between global model w^s and base model w^0 and that between individualized model w^u and shared global model w^s during online model updating. To visualize the results, we computed and plotted the average Euclidean distances in every 3000 observations during online learning, together with the corresponding variance. To illustrate a comprehensive picture of online model update, we also plotted the corresponding average F1 performance for both positive and negative class. Because the Euclidean distance between w^s and w^0 is much larger than that between w^s and w^u, we scaled $||w^s - w^0||$ by 0.02 on Amazon dataset in Figure 2. Similar results were observed on Yelp data as well; but due to space limit, we do not include them.

As we can clearly observe that the difference between the base model and newly adapted global model kept increasing during online update. At the earlier stage, it is increasing much faster than the later stage, and the corresponding classification performance improves more rapidly (especially in negative class). The considerably large variance between w^0 and w^s at the beginning indicates the divergence between old and new social norms across users. Later on, variance decreased and converged with more observations, which can be understood as the formation of the new social norms among users. On the other hand, the distance between personalized models and shared global model fluctuated a lot at the beginning; with more observations, it became stable later on. This is also reflected in the range of variance: the variance is much smaller in later stage than earlier stage, which indicates users comply to the newly established social norms.

Table 3: Shared model adaptation for cold start on Amazon and Yelp.

Obs.	Amazon						Yelp					
	Shared-SVM		MT-SVM		MT-LinAdapt		Shared-SVM		MT-SVM		MT-LinAdapt	
	Pos F1	Neg F1	Pos F1	Neg F1	Pos F1	Neg F1	Pos F1	Neg F1	Pos F1	Neg F1	Pos F1	Neg F1
1^{st}	0.9004	0.7013	**0.9264**	0.7489	0.9122	**0.7598**	0.7882	0.5537	**0.9040**	0.7201	0.8809	**0.7306**
2^{nd}	0.9200	0.6872	**0.9200**	**0.7319**	0.8945	0.7292	0.7702	0.5266	**0.8962**	0.6959	0.8598	**0.6968**
3^{rd}	0.9164	0.6967	**0.9164**	0.7144	0.8967	**0.7260**	0.7868	0.5278	**0.9063**	**0.7099**	0.8708	0.7069

4.4 Shared Adaptation Against Cold Start

Cold start refers to the challenge that a statistic model cannot draw any inference for users before sufficient observations are gathered (Schein et al., 2002). The shared model adaptation in MT-LinAdapt helps alleviate cold start in personalized sentiment analysis, while individualized model adaptation method, such as RegLR and LinAdapt, cannot achieve so. To verify this aspect, we separated both Amazon and Yelp reviewers into two sets: we randomly selected 1,000 reviewers from the isolated training set and exhausted all their reviews to estimate a shared SVM model, MT-LinAdapt and MT-SVM. Then those models were directly applied onto the testing reviewers for evaluation. Again, because it is time consuming to retrain a SVM model repeatedly, only MT-LinAdapt performed online model update in this evaluation. We report the performance on the first three observations from all testing users accordingly in Table 3.

MT-LinAdapt achieved promising performance on the first testing cases, especially on the negative class. This indicates its estimated global model is more accurate on the new testing users. Because MT-SVM cannot be updated during this online test, only its previously estimated global model from the 1,000 training users can be applied here. As we can notice, its performance is very similar to the shared SVM model (especially on Amazon dataset). MT-LinAdapt adapts to this new collection of users very quickly, so that improved performance against the static models at later stage is achieved.

4.5 Vocabulary Stability

One derivative motivation for personalized sentiment analysis is to study the diverse use of vocabulary across individual users. We analyzed the variance of words' sentiment polarities estimated in the personalized models against the base model. Table 4 shows the most and the least variable features on both datasets. It is interesting to find that words with strong sentiment polarities tend to be more stable across users, such as "disgust," "regret," and "excel." This demonstrates the sign of conformation to social norms. There are also words exhibiting high variances in sentiment polarity, such as "was-yummi," "lazi," and "cheat," which indicates the heterogeneity of users' opinionated expressions.

Table 4: Top six words with the highest and lowest variances of learned polarities by MT-LinAdapt.

Amazon	Highest	cheat astound	healthi the-wrong	enjoy-read the-amaz
	Lowest	mistak regret	favor perfect-for	excel great
Yelp	Highest	total-worth advis	lazi impress	was-yummi so-friend
	Lowest	omg frustrat	veri-good disgust	hungri a-must

5 Conclusions

In this work, we proposed to perform personalized sentiment classification based on the notion of shared model adaptation, which is motivated by the social theories that humans' opinions are diverse but shaped by the ever-changing social norms. In the proposed MT-LinAdapt algorithm, global model sharing alleviates data sparsity issue, and individualized model adaptation captures the heterogeneity in humans' sentiments and enables efficient online model learning. Extensive experiments on two large review collections from Amazon and Yelp confirmed the effectiveness of our proposed solution.

The idea of shared model adaptation is general and can be further extended. We currently used a two-level model adaptation scheme. The adaptation can be performed at the user group level, i.e., three-level model adaptation. The user groups can be automatically identified to maximize the effectiveness of shared model adaptation. In addition, this method can also be applied to domain adaptation, where a domain taxonomy enables a hierarchically shared model adaptation.

6 Acknowledgments

We thank the anonymous reviewers for their insightful comments. This paper is based upon work supported by the National Science Foundation under grant IIS-1553568.

References

[Al Boni et al.2015] Mohammad Al Boni, Keira Qi Zhou, Hongning Wang, and Matthew S Gerber. 2015. Model adaptation for personalized opinion analysis. *In Proceedings of ACL*.

[Argyriou et al.2008] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2008. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272.

[Barsäde and Gibson1998] Sigal G Barsäde and Donald E Gibson. 1998. Group emotion: A view from top and bottom. *Research on managing groups and teams*, 1:81–102.

[Blitzer et al.2006] John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 EMNLP*, pages 120–128. ACL.

[Brighton and Mellish2002] Henry Brighton and Chris Mellish. 2002. Advances in instance selection for instance-based learning algorithms. *Data mining and knowledge discovery*, 6(2):153–172.

[Briley et al.2000] Donnel A Briley, Michael W Morris, and Itamar Simonson. 2000. Reasons as carriers of culture: Dynamic versus dispositional models of cultural influence on decision making. *Journal of consumer research*, 27(2):157–178.

[Ehrlich and Levin2005] Paul R Ehrlich and Simon A Levin. 2005. The evolution of norms. *PLoS Biol*, 3(6):e194.

[Evgeniou and Pontil2004] Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi–task learning. In *Proceedings of the 10th ACM SIGKDD*, pages 109–117. ACM.

[Evgeniou and Pontil2007] A Evgeniou and Massimiliano Pontil. 2007. Multi-task feature learning. *Advances in neural information processing systems*, 19:41.

[Evgeniou et al.2007] Theodoros Evgeniou, Massimiliano Pontil, and Olivier Toubia. 2007. A convex optimization approach to modeling consumer heterogeneity in conjoint estimation. *Marketing Science*, 26(6):805–818.

[Gao et al.2014] Wenliang Gao, Nobuhiro Kaji, Naoki Yoshinaga, and Masaru Kitsuregawa. 2014. Collective sentiment classification based on user leniency and product popularity. *Ił*, 21(3):541–561.

[Geng et al.2012] Bo Geng, Yichen Yang, Chao Xu, and Xian-Sheng Hua. 2012. Ranking model adaptation for domain-specific search. *IEEE Transactions on Knowledge and Data Engineering*, 24(4):745–758.

[Hochschild1975] Arlie Russell Hochschild. 1975. The sociology of feeling and emotion: Selected possibilities. *Sociological Inquiry*, 45(2-3):280–307.

[Hu et al.2013] Xia Hu, Lei Tang, Jiliang Tang, and Huan Liu. 2013. Exploiting social relations for sentiment analysis in microblogging. In *Proceedings of the 6th WSDM*, pages 537–546. ACM.

[Jacob et al.2009] Laurent Jacob, Jean-philippe Vert, and Francis R Bach. 2009. Clustered multi-task learning: A convex formulation. In *NIPS*, pages 745–752.

[Kiwiel2001] Krzysztof C Kiwiel. 2001. Convergence and efficiency of subgradient methods for quasiconvex minimization. *Mathematical programming*, 90(1):1–25.

[Li et al.2010] Guangxia Li, Steven CH Hoi, Kuiyu Chang, and Ramesh Jain. 2010. Micro-blogging sentiment detection by collaborative online learning. In *ICDM*, pages 893–898. IEEE.

[Liu2012] Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

[Max2014] Woolf Max. 2014. A statistical analysis of 1.2 million amazon reviews. `http://minimaxir.com/2014/06/reviewing-reviews`.

[McAuley et al.2015] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM.

[Ostrom2014] Elinor Ostrom. 2014. Collective action and the evolution of social norms. *Journal of Natural Resources Policy Research*, 6(4):235–252.

[Pan and Yang2010] Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

[Pan et al.2010] Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of the 19th WWW*, pages 751–760. ACM.

[Pang and Lee2005] Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd ACL*, pages 115–124. ACL.

[Pang and Lee2008] Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

[Pang et al.2002] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86. ACL.

[Schein et al.2002] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM.

[Sherif1936] Muzafer Sherif. 1936. The psychology of social norms.

[Shott1979] Susan Shott. 1979. Emotion and social life: A symbolic interactionist analysis. *American journal of Sociology*, pages 1317–1334.

[Tan et al.2011] Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of the 17th ACM SIGKDD*, pages 1397–1405. ACM.

[Titov and McDonald2008] Ivan Titov and Ryan T McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *ACL*, volume 8, pages 308–316. Citeseer.

[Wang et al.2011] Hongning Wang, Yue Lu, and ChengXiang Zhai. 2011. Latent aspect rating analysis without aspect keyword supervision. In *Proceedings of the 17th ACM SIGKDD*, pages 618–626. ACM.

[Wang et al.2013] Hongning Wang, Xiaodong He, Ming-Wei Chang, Yang Song, Ryen W White, and Wei Chu. 2013. Personalized ranking model adaptation for web search. In *Proceedings of the 36th ACM SIGIR*, pages 323–332. ACM.

[Wiebe et al.2005] Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

[Yang and Pedersen1997] Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML*, volume 97, pages 412–420.

[Yelp2016] Yelp. 2016. Yelp dataset challenge. `https://www.yelp.com/dataset_challenge`.

[Zhu et al.1997] Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. 1997. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software (TOMS)*, 23(4):550–560.

Modeling Concept Dependencies in a Scientific Corpus

Jonathan Gordon, **Linhong Zhu**, **Aram Galstyan**, **Prem Natarajan**, and **Gully Burns**
USC Information Sciences Institute
Marina del Rey, CA, USA
{jgordon, linhong, galstyan, pnataraj, burns}@isi.edu

Abstract

Our goal is to generate reading lists for students that help them optimally learn technical material. Existing retrieval algorithms return items directly relevant to a query but do not return results to help users read about the concepts supporting their query. This is because the dependency structure of concepts that must be understood before reading material pertaining to a given query is never considered. Here we formulate an information-theoretic view of concept dependency and present methods to construct a "concept graph" automatically from a text corpus. We perform the first human evaluation of concept dependency edges (to be published as open data), and the results verify the feasibility of automatic approaches for inferring concepts and their dependency relations. This result can support search capabilities that may be tuned to help users learn a subject rather than retrieve documents based on a single query.

1 Introduction

Corpora of technical documents, such as the ACL Anthology, are valuable for learners, but it can be difficult to find the most appropriate documents to read in order to learn about a concept. This problem is made more complicated by the need to trace the ideas back to those that need to be learned first (e.g., before you can learn about Markov logic networks, you should understand first-order logic and probability). That is, a crucial question when learning a new subject is "What do I need to know before I start reading about this?"

To answer this question, learners typically rely on the guidance of domain experts, who can devise pedagogically valuable reading lists that order documents to progress from prerequisite to target concepts. Thus, it is desirable to have a model where each concept is linked to the prerequisite concepts it depends upon – a *concept graph*. A manually constructed concept graph excerpt related to automatic speech recognition is shown in Figure 1. The dependency relation between two concepts is interpreted as whether understanding one concept would help a learner understand the other.

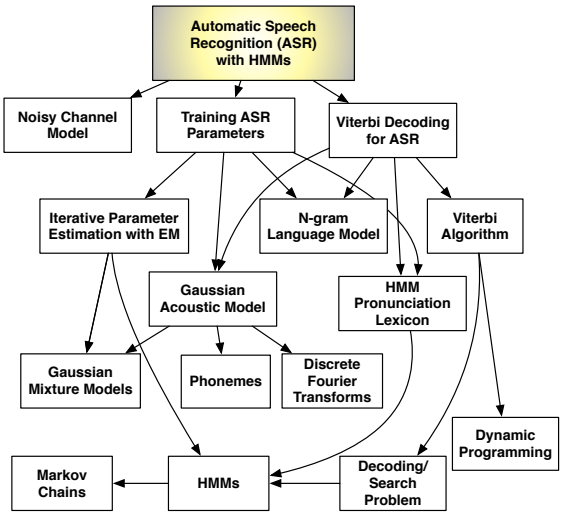


Figure 1: A human-authored concept graph excerpt, showing possible concepts related to automatic speech recognition and their concept dependencies.

Representing a scientific corpus in this way can improve tasks such as curriculum planning (Yang et al., 2015), automatic reading list generation (Jardine, 2014), and improving education quality (Rouly et al., 2015). Motivated by the importance of representing the content of a scientific corpus as a concept graph, the challenge we address in this work is to automatically infer the concepts and their dependency relations.

Towards this end, we first instantiate each concept as a topic from statistical topic modeling (Blei et al., 2003). To link concepts with directed depen-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 866–875,
Berlin, Germany, August 7-12, 2016.

dency edges, we propose the use of information-theoretic measures, which we compare against baseline methods of computing word similarity, hierarchical clustering, and citation prediction. We then gather human annotations of concept graph nodes and edges learned from the ACL Anthology, which we use to evaluate these methods.

The main contributions of this paper are:

1 We introduce the concept graph representation for modeling the technical concepts in a corpus and their relations.
2 We present information-theoretic approaches to infer concept dependence relations.
3 We perform the first human annotation of concept dependence for a technical corpus.
4 We release the human annotation data for use in future research.

In the following section, we contrast this problem with previous work. We then describe the concept graph framework (Section 3) and present automatic approaches for inferring concept graphs (Section 4). The details of human evaluation are presented in Section 5. We discuss some interesting open questions related to this work in Section 6 before concluding this work.

2 Related Work

There is a long history of work on identifying structure in the contents of a text corpus. Our approach is to link documents to concepts and to model relations among these concepts rather than to identify the specific claims (Schäfer et al., 2011) or empirical results (Choi et al., 2016) in each document. In this section, we first provide an overview of different relations between concepts, followed by discussion of some representative methods for inferring them. We briefly discuss the differences between these relations and the concept dependency relation we are interested in.

Similarity Concepts are similar to the extent that they share content. Grefenstette (1994) applied the Jaccard similarity measure to relate concepts to each other. White and Jose (2004) empirically studied 10 similarity metrics on a small sample of 10 pairs of topics, and the results suggested that correlation-based measures best match general subject perceptions of search topic similarity.

Hierarchy Previous work on linking concepts has usually been concerned with forming subsumption hierarchies from text (Woods, 1997; Sanderson and Croft, 1999; Cimiano et al., 2005) – e.g., *Machine translation* is part of *Natural language processing* – and more recent work does so for statistical topic models. Jonyer et al. (2002) applied graph-based hierarchical clustering to learn hierarchies from both structured and unstructured data. Ho et al. (2012) learn a topic taxonomy from the ACL Anthology and from Wikipedia with a method that scales linearly with the number of topics and the tree depth.

Other relations Every pair of concepts is statistically correlated with each other based on word co-occurrence (Blei and Lafferty, 2006) providing a simple baseline metric for comparison. For a topic modeling approach performed over document citation links rather than over words or n-grams, Wang et al. (2013) gave a topic A's dependence on another topic B as the probability of a document in A citing a document in B.

Our approach to studying *concept dependence* differs from the relations derived from similarity, hierarchy, correlation and citation mentioned above, but intuitively they are related. We thus adapt one representative method for the similarity (Grefenstette, 1994), hierarchy (Jonyer et al., 2002), and citation likelihood (Wang et al., 2013) relations as baselines for computing concept dependency relations in Section 4.2.3.

Concept dependence is also related to curriculum planning. Yang et al. (2015) and Talukdar and Cohen (2012) studied prerequisite relationships between course material documents based on external information from Wikipedia. They assumed that hyperlinks between Wikipedia pages and course material indicate a prerequisite relationship. With this assumption, Talukdar and Cohen (2012) use crowdsourcing approaches to obtain a subset of the prerequisite structure and train a maximum entropy–based classifier to identify the prerequisite structure. Yang et al. (2015) applied both classification and learning to rank approaches in order to classify or rank prerequisite structure.

3 Concept Graph Representation of a Text Corpus

We represent the scientific literature as a labeled graph, where nodes represent both documents and concepts – and, optionally, metadata (such as author, title, conference, year) and features (such as

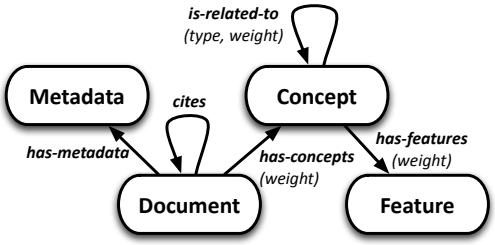


Figure 2: The Concept Graph Data Schema. Each node is a class and edges are named relations between classes (with associated attributes).

words, or n-grams) – and labeled edges represent the relations between nodes. Figure 2 shows an example schema for a concept graph representation for a scientific corpus.

Concepts are abstract and require a concrete representation. In this work, we use statistical topic modeling, where each topic – a multinomial distribution over a vocabulary of words – is taken as a single concept. Documents are linked to concepts by weighted edges, which can be derived from the topic model's document–topic composition distributions. Other approaches to identifying concepts are considered in Section 6.

Concepts exhibit various relations to other concepts, such as *hierarchy*, connecting more general and more specific concepts; *similarity*; and *correlation*. We model each concept as a node and concept-to-concept relations as directed, weighted, labeled edges. The label of an edge denotes the type of relation, such as "is similar to", "depends on", and "relates to", and the weights represent the strength of different relations.

In this work, we focus on *concept dependency*, which is the least studied of these relations and, intuitively, the most important for learners. We consider there to be a dependency relation between two concepts if understanding one concept would help you to understand the other. This notion forms the core of our human-annotated data set which demonstrates that this idea is meaningful and robust for expert annotators when asked to judge if there exists a dependency relation between two concepts defined by LDA topics (see Section 5.2).

4 Learning the Concept Graph

4.1 Identifying Concepts

The representation of concepts using topics is very general, and any effective topic modeling approach can be applied. These include probabilistic latent semantic indexing (PLSI) (Hofmann, 1999), latent Dirichlet allocation (LDA) (Blei et al., 2003), and non-negative matrix factorization (NMF) (Arora et al., 2012). In our experiments, we use the open-source tool Mallet (McCallum, 2002), which provides a highly scalable implementation of LDA; see Section 5.1 for more details.

4.2 Discovering Concept Dependency Relations

Identifying concept dependency relations between topics is the key step for building a useful concept graph. These relations add semantic structure to the contents of the text corpus, and they facilitate search and ordering in information retrieval. In this section, as a proof-of-concept, we propose two information-theoretic approaches to learn concept dependency relations: an approach based on cross entropy and another based on information flow.

4.2.1 Cross-entropy Approach

The intuition of the cross-entropy approach is simple: Given concepts c_i and c_j, if most of the instances of c_i can be explained by the occurrences of c_j, but not vice versa, it is likely that c_i depends on c_j. For example, if c_i is *Markov logic networks* (*MLNs*) and c_j is *Probability*, we might say that observing *MLNs* depends on seeing *Probability* since most of the times that we see *MLNs*, we also see *Probability*, but the opposite does not hold.

Given concepts c_i and c_j, the cross-entropy approach predicts that c_i depends on c_j if they satisfy these conditions:

1 The distribution of c_i is better approximated by that of c_j than the distribution of c_j is approximated by that of c_i.
2 The co-occurrence frequency of instances of c_i and c_j is relatively higher than that of a non-dependency pair.

Therefore, to predict the concept dependency relation, we need to examine whether the distribution of c_i could well approximate the distribution of c_j and the joint distribution of c_i and c_j. For this, we use cross entropy and joint entropy:

Cross entropy measures the difference between two distributions. Specifically, the cross entropy for the distributions X and Y over a given set is defined as:

$$H(X;Y) = H(X) + D_{KL}(X||Y) \qquad (1)$$

where $H(X)$ is the entropy of X, and $D_{KL}(X||Y)$ is the Kullback–Leibler divergence of an estimated distribution Y from true distribution X. Therefore, $H(X;Y)$ examines how well the distribution of Y approximates that of X.

Joint entropy measures the information we obtained when we observe both X and Y. The joint Shannon entropy of two variables X and Y is defined as:

$$H(X,Y) = \sum_X \sum_Y P(X,Y) \log_2 P(X,Y) \quad (2)$$

where $P(X,Y)$ is the joint probability of these values occurring together.

Based on the conditions listed above and these definitions, we say that c_i depends on c_j if and only if they satisfy the following constraints:

$$\begin{aligned} H(c_i;c_j) &> H(c_j;c_i) \\ H(c_i,c_j) &\leq \theta \end{aligned} \quad (3)$$

with θ as a threshold value, which can be interpreted as "the average joint entropy of any non-dependence concepts". The weight of the dependency is defined as:

$$D_{\mathrm{CE}}(c_i,c_j) = H(c_i;c_j)$$

The cross-entropy method is general and can be applied to different distributions used to model concepts, such as distributions of relevant words, of relevant documents, or of the documents that are cited by relevant documents.

4.2.2 Information-flow Approach

Now we consider predicting concept dependency relations from the perspective of navigating information. Imagine that we already have a perfect concept dependency graph. When we are at a concept node (e.g., reading a document about it), the navigation is more likely to continue to a concept it depends on than to other concepts that it doesn't depend on. To give a concrete example, if we are navigating from the concept *Page rank*, it is more likely for us to jump to *Eigenvalue* than to *Language model*. Therefore, if concept c_i depends on concept c_j, then c_j generally receives more navigation hits than c_i and has higher "information flow".

Based on this intuition, we can predict concept dependency relations using information flow: Given concepts c_i and c_j, c_i depends on c_j if they satisfy these conditions:

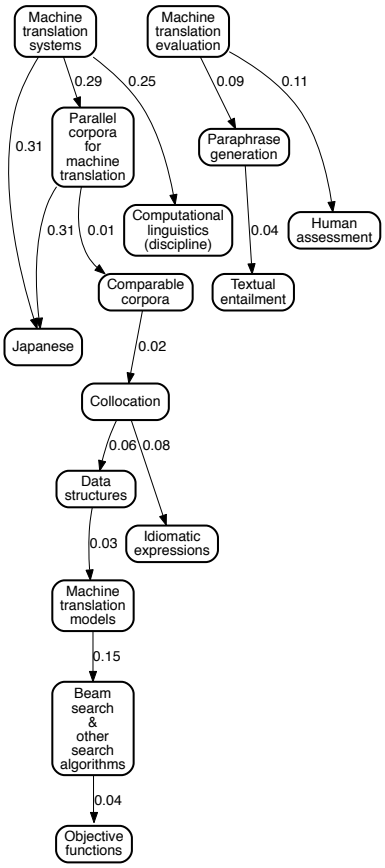


Figure 3: A concept graph excerpt related to machine translation, where concepts are linked based on cross entropy. Concepts are represented by manually chosen names, and links to documents are omitted.

1 The concept c_i receives relatively lower navigation hits than c_j.
2 The number of navigation traces from concept c_i to c_j is much stronger than that to another non-dependent concept c_k.

While we do not have data for human navigation between concepts, a natural way to simulate this is through information flow. As proposed by Rosvall and Bergstrom (2008), we use the probability flow of random walks on a network as a proxy for information flow in the real system. Given any observed graph G, the information score $I(v)$ of a node v, is defined as its steady state visit frequency. The information flow $I(u,v)$ from node u to node v, is consequently defined as the transition probability (or "exit probability") from u to v.

To this end, we construct a graph connecting concepts by their co-occurrences in documents, and we can use either Map Equation (Rosvall and

Bergstrom, 2008) or Content Map Equation (Smith et al., 2014) to compute the information flow network and the information score for each concept node. The details are outlined as follows:

1 Construct a concept graph G^{co} based on co-occurrence observations. We define weighted, undirected edges within the concept graph based on the number of documents in which the concepts co-occur. Formally, given concepts c_i and c_j and a threshold $0 \leq \tau \leq 1$, the weighted edge is calculated as:

$$w^{co}(c_i, c_j) = \begin{cases} \sum_d p(c_i|d)p(c_j|d) & \text{if } p(c|d) > \tau \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

2 Given the graph G^{co}, we compute the information score $I(c)$ for each concept node c and information flow $I(c_i, c_j)$ between a pair of nodes c_i and c_j. For the details of calculating $I(c)$ and $I(c_i, c_j)$, refer to Map Equation (Rosvall and Bergstrom, 2008) and Content Map Equation (Smith et al., 2014).

3 Given two concepts c_i and c_j, we link c_i to c_j with a directed edge if $I(c_i) > I(c_j)$ with weight:

$$D_{\text{IF}}(c_i, c_j) = I(c_i, c_j)$$

The information flow approach for inferring dependency can be further improved with a few true human navigation traces. As introduced earlier, the concept graph representation facilitates applications such as reading list generation, and document retrieval. Those applications enable the collection of human navigation traces, which can provide a better approximation of dependency relation.

4.2.3 Baseline Approaches

Similarity Relations Intuitively, concepts that are more similar (e.g., *Machine translation* and *Machine translation evaluation*) are more likely to be connected by concept dependency relations than less similar concepts are. As a baseline, we compute the Jaccard similarity coefficient based on the top 20 words or n-grams in the concept's topic word distributions.

Hierarchical Relations Previous work has looked at learning hierarchies that connect broader topics (acting as equivalent proxies for concepts in our work) to more specific subtopics (Cimiano et al., 2005; Sanderson and Croft, 1999). We compare against a method for doing so to see how close identifying hierarchical relations comes to our goal of identifying concept dependency relations. Specifically, we perform agglomerative clustering over the topic–topic co-occurrence graph G^{co} with weights defined in Eq. 4, in order to obtain the hierarchical representation for concepts.

Citation-based Given concepts c_i and c_j, if the documents that are highly related to c_j are cited by most of the instances of c_i, c_i may depend on c_j. Wang et al. (2013) used this approach in the context of CitationLDA topic modeling, where topics are learned from citation links rather than text. We adapt this for regular LDA so that the concept c_i depends on c_j with weight

$$\text{D}_{\text{Cite}}(c_i, c_j) = \sum_{d_1 \in D} \sum_{d_2 \in C_{d_1}} T_{1,i} T_{2,j} \quad (5)$$

where D is the set of all documents, C_d are the documents cited by d, and $T_{x,y}$ is the distribution of documents d_x composed of concepts c_y. For this method, we return a score of 0 if the concepts do not co-occur in at least three documents.

5 Evaluation of Concept Graphs

There are two main approaches to evaluating a concept graph: We can directly evaluate the graph, using human judgments to measure the quality of the concepts and the reliability of the links between them. Alternatively, we can evaluate the application of a concept graph to a task, such as ordering documents for a reading list or recommending documents to cite when writing a paper.

Our motivation to build a concept graph from a technical corpus is to improve performance at the task of reading list generation. However, an applied evaluation makes it harder to judge the quality of the concept graph itself. Each document contains a combination of concepts, which have different ordering restrictions, and other factors also affect the quality of a reading list, such as the classification of document difficulty and type (e.g., survey, tutorial, or experimental results). As such, we focus on a direct human evaluation of our proposed methods for building a concept graph and leave the measure of applied performance to future work.

5.1 Corpus and its Evaluation Concept Graphs

For this evaluation, the scientific corpus we use is the ACL Anthology. This consists of articles published in a variety of journals, conferences,

and workshops related to computational linguistics. Specifically, we use a modified copy of the plain text distributed for the ACL Anthology Network (AAN), release 2013 (Radev et al., 2013), which includes 23,261 documents from 1965 to 2013. The AAN includes plain text for documents, with OCR performed using PDFBox. We manually substituted OmniPage OCR output from the ACL Anthology Reference Corpus, version 1 (Bird et al., 2008) for documents where it was observed to be of higher quality. The text was processed to join words that were split across lines with hyphens. We manually removed documents that were not written in English or where text extraction failed, leaving 20,264 documents, though this filtering was not exhaustive.

The topic model we used was built using the Mallet (McCallum, 2002) implementation of LDA. It is composed of bigrams, filtered of typical English stop words before the generation of bigrams, so that, e.g., "word to word" yields the bigram "word word". We generated topic models consisting of between 20 and 400 topics and selected a 300-topic model based on manual inspection. Documents were linked to concepts based on the document's LDA topic composition. The concept nodes for each topic were linked in concept dependency relations using each of the methods described in Section 4, producing five concept graphs to evaluate. We applied the general cross-entropy method to the distribution of top-k bigrams for each concept. For all methods, the results we report are for $k = 20$. Changing this value shifts the precision–recall trade-off, but in our experiments, the relative performance of the methods are generally consistent for different values of k.

Since it is impractical to manually annotate all pairs of concept nodes from a 300-node graph, we selected a subset of edges for evaluation. Intuitively, the evaluation set should satisfy the following sampling criteria: (1) The evaluation set should cover the top weighted edges for a precision evaluation. (2) The evaluation set should cover the bottom-weighted edges for a recall evaluation. (3) The evaluation set should provide low-biased sampling. With respect to these requirements, we generated an evaluation edge set as the union of the following three sets:

1 Top-20 edges for each approach (including baseline approaches)
2 A random shuffle selection from the union of the top-50 and bottom-50 edges in terms of the baseline word similarity.[1]
3 A random shuffle section from the union of top-100 edges in terms of the proposed approaches.

Judges	All	Coherent	Related	Dependent
Non-NLP	0.407	0.446	0.305	0.329
NLP	0.526	0.610	0.448	0.395
All	0.467	0.529	0.354	0.357

Table 1: Inter-annotator agreement measured as Pearson correlation.

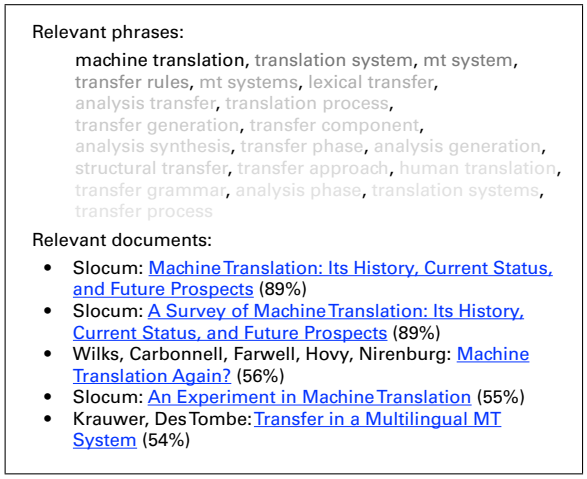


Figure 4: An example of the presentation of a topic for human evaluation.

5.2 Human Annotation

For annotation, we present pairs of topics followed by questions. Each topic is presented to a judge as a list of the most relevant bigrams in descending order of their topic-specific "collapsed" probabilities. These are presented in greyscale so that the most relevant items appear black, fading through grey to white as the strength of that item's association with the topic decreases. The evaluation interface also lists the documents that are most relevant to the topic, linked to the original PDFs. These documents can be used to clarify the occurrence of unfamiliar terms, such as author names or common examples that may show up in the topic representation. An example topic is shown in Figure 4.

For each topic, judges were asked:

1 How clear and coherent is Topic 1?
2 How clear and coherent is Topic 2?

[1] We observe that usually if the edge strength in terms of one of the information-theoretic methods is zero, the word similarity is zero as well, but if the word similarity is zero, the edge strength in terms of the proposed methods may be non-zero.

Edges	Top 20	Top 150			All scores > 0		
	Prec.	Prec.	Rec.	f_1	Prec.	Rec.	f_1
Cross entropy (D_{CE})	**0.851**	0.765	0.358	0.487	0.693	0.670	0.681
Information flow (D_{IF})	0.793	0.696	0.311	0.429	0.693	0.323	0.441
Word similarity (D_{Sim})	0.808	**0.768**	0.382	**0.511**	**0.768**	0.382	0.511
Hierarchy (D_{Hier})	0.680	0.692	0.297	0.416	0.686	0.638	0.661
Cite (D_{Cite})	0.693	0.718	0.343	0.465	0.693	0.670	0.681
Random	0.659	0.661	**0.580**	0.500	0.658	**1.000**	**0.794**

Table 2: Precision, recall, and f-scores (with different thresholds for which edges are included) for the methods of predicting dependency relations between concepts described in Section 4.2.

If both topics are at least somewhat clear:

3 How related are these topics?
4 Would understanding Topic 1 help you to understand Topic 2?
5 Would understanding Topic 2 help you to understand Topic 1?

For each question, they could answer "I don't know" or select from an ordinal scale:

1 Not at all
2 Somewhat
3 Very much

The evaluation was completed by eight judges with varying levels of familiarity with the technical domain. Four judges are NLP researchers: Three PhD students working in the area and one of the authors. Four judges are familiar with NLP but have less experience with NLP research: two MS students, an AI PhD student, and one of the authors. The full evaluation was divided into 10 sets taking a total of around 6–8 hours per person to annotate. Their overall inter-annotator agreement and the agreement for each question type is given in Table 1. Agreement is higher when we consider only judgments from NLP researchers, but in all cases is moderate, indicating the difficulty of interpreting statistical topics as concepts and judging the strength (if any) of the concept dependency relation between them.

The topic coherence judgments that were collected served to make each human judge consider how well she understood each topic before judging their dependence. The topic relatedness questions provided an opportunity to indicate that if the annotator recognized a relation between the topics without needing to say that their was a dependence.

5.3 Evaluation of Automatic Methods

To measure the quality of the concept dependency edges in our graphs, we compute the average precision for the strongest edges in each concept graph, up to three thresholds: the top 20 edges, the top 150, and all edges with strength > 0. These precision scores are in Table 2 as well as the corresponding recall, and f_1 scores for the larger thresholds. Despite the difference in inter-annotator agreement reported in Table 1, the ordering of methods by precision is the same whether we consider only the judgments of NLP experts, non-NLP judges, or everyone, so we only report the average across all annotators.

When we examine the results of precision at 20 – the strongest edges predicted by each method – we see that the cross-entropy method performs best. For comparison, we report the accuracy of a baseline of random numbers between 0 and 1. While all methods have better than chance precision, the random baseline has higher recall since it predicts a dependency relation of non-zero strength for all pairs. As we consider edges predicted with lower confidence, the word similarity approach shows the highest precision. A limitation of the word similarity baseline is that it is symmetric while concept dependence relations can be asymmetric.

Annotators marked many pairs of concepts as being at least somewhat co-dependent. E.g., understanding *Speech recognition* strongly helps you understand *Natural language processing*, but being familiar with this broader topic also somewhat helps you understand the narrower one. The precision scores we report count both annotations of concept dependence ("Somewhat" and "Very much") as positive predictions, but other evaluation metrics might show a greater benefit for methods like D_{CE} that can predict dependency with asymmetric strengths.

6 Discussion

Another natural evaluation of an automatically generated concept graph would be to compare it to a

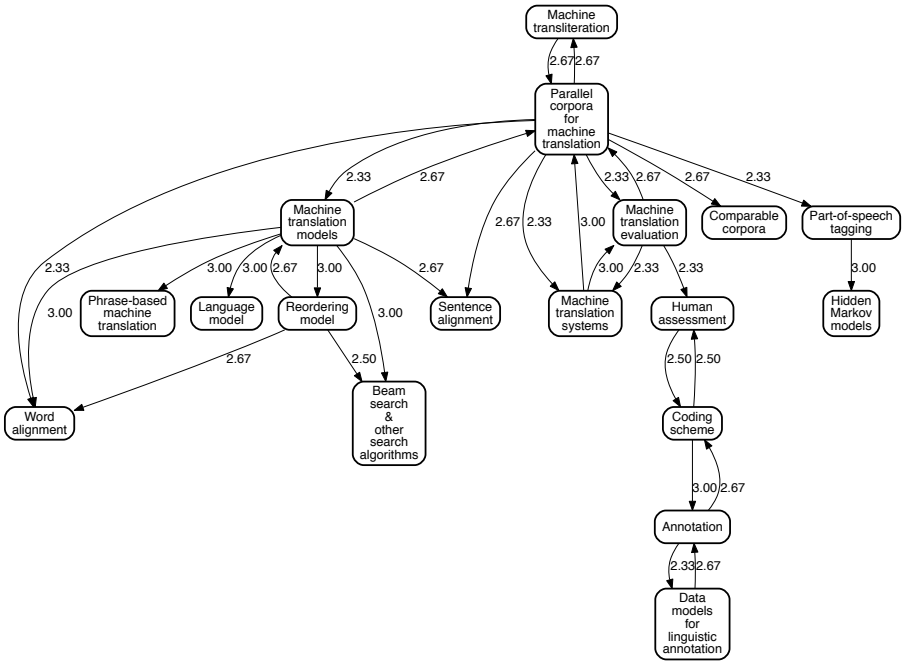


Figure 5: A concept graph excerpt related to machine translation, where concepts are joined based on the judgments of human annotators. Concepts are represented by manually chosen names, and links to documents are omitted.

human-generated gold standard, where an expert has created concept nodes at the optimal level of generality and linked these by her understanding of the conceptual dependencies among concepts in the domain. However, there are several difficulties with this approach: (1) It is quite labor-intensive to manually generate a concept graph; (2) we expect only moderate agreement between graphs produced by different experts, who have different ideas of what concepts are important and distinct and which concepts are important to understanding others; and (3) the concept graphs we learn from a collection of documents will differ significantly from those we imagine, without these differences necessarily being better or worse.

In this work, we assume that a topic model provides a reasonable proxy for the concepts a person might identify in a technical corpus. However, topic modeling approaches are better at finding general areas of research than at identifying fine-grained concepts like those shown in Figure 1. The concept graph formalism can be extended with the use of discrete entities, identified by a small set of names, e.g., (*First-order logic*, *FOL*). We have performed initial work on two approaches to extract entities:

1 We can use an external reference, Wikipedia, to help entity extraction. We count the occurrences of each article title in the scientific corpus, and we keep the high-frequency titles as entities. For example, in the ACL Anthology corpus, we obtain 56 thousand entities (page titles) that occurred at least once and 1,123 entities that occur at least 100 times.

2 We cannot assume that the important entities in every scientific or technical corpus will be well-represented on Wikipedia. In the absence of a suitable external reference source, we can use the open-source tool SKIMMR (Nováček and Burns, 2014) or the method proposed by Jardine (2014) to extract important noun phrases to use as entities. The importance of a potential entity can be computed based on the occurrence frequency and the sentence-level co-occurrence frequency with other phrases.

Another limitation of using a topic model like LDA as a proxy for concepts is that the topics are static, while a corpus may span decades of research. Studying how latent models might evolve or "drift" over time within a textual corpus describing a technical discipline is an important research question, and our approach could be extended to add or remove topics in a central model over time.

Despite its limitations, a topic model is useful for automatically discovering concepts in a corpus even if the concept is not explicitly mentioned in a document (e.g., the words "axiom" or "predi-

cate" might indicate discussion of logic) or has no canonical name. The concept graph representation allows for the introduction of additional or alternative features for concepts, making it suitable for new methods of identifying and linking concepts.

7 Conclusions

Problems such as reading list generation require a representation of the structure of the content of a scientific corpus. We have proposed the concept graph framework, which gives weighted links from documents to the concepts they discuss and links concepts to one another. The most important link in the graph is the concept dependency relation, which indicates that one concept helps a learner to understand another, e.g., *Markov logic networks* depends on *Probability*.

We have presented four approaches to predicting these relations. We propose information-theoretic measures based on cross entropy and on information flow. We also present baselines that compute the similarity of the word distributions associated with each concept, the likelihood of a citation connecting the concepts, and a hierarchical clustering approach. While word similarity proves a strong baseline, the strongest edges predicted by the cross-entropy approach are more precise. We are releasing human annotations of concept nodes and possible dependency edges learned from the ACL Anthology as well as implementations of the methods described in this paper to enable future research on modeling scientific corpora.[2]

Acknowledgments

The authors thank Yigal Arens, Emily Sheng, and Jon May for their valuable feedback on this work.

This work was supported by the Intelligence Advanced Research Projects Activity (IARPA) via the Air Force Research Laboratory. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

[2]The code and data associated with this work are available at http://techknacq.isi.edu

References

Sanjeev Arora, Rong Ge, and Ankur Moitra. 2012. Learning topic models – going beyond SVD. In *Proceedings of the 53rd Annual Symposium on Foundations of Computer Science*, pages 1–10. IEEE.

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. 2008. The ACL Anthology Reference Corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, Marrakech, Morocco, May. European Language Resources Association.

David Blei and John Lafferty. 2006. Correlated topic models. In *Advances in Neural Information Processing Systems*.

David M. Blei, Andew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Eunsol Choi, Matic Horvat, Jon May, Kevin Knight, and Daniel Marcu. 2016. Extracting structured scholarly information from the machine translation literature. In *Proceedings of the 10th International Conference on Language Resources and Evaluation*. European Language Resources Association.

Philipp Cimiano, Andreas Hotho, and Steffen Staab. 2005. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24(1):305–39, August.

Gregory Grefenstette. 1994. *Explorations in Automatic Thesaurus Discovery*. Kluwer Academic Publishers, Norwell, MA, USA.

Qirong Ho, Jacob Eisenstein, and Eric P. Xing. 2012. Document hierarchies from text and links. In *Proceedings of the International World Wide Web Conference*, April.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–7. ACM.

James G. Jardine. 2014. Automatically generating reading lists. Technical Report UCAM-CL-TR-848, University of Cambridge Computer Laboratory, February.

Istvan Jonyer, Diane J. Cook, and Lawrence B. Holder. 2002. Graph-based hierarchical conceptual clustering. *Journal of Machine Learning Research*, 2:19–43, March.

Andrew McCallum. 2002. MALLET: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Vít Nováček and Gully APC Burns. 2014. SKIMMR: Facilitating knowledge discovery in life sciences by machine-aided skim reading. *PeerJ*, 2:e483.

Dragomir R. Radev, Pradeep Muthukrishnan, Vahed Qazvinian, and Amjad Abu-Jbara. 2013. The ACL Anthology Network Corpus. *Language Resources and Evaluation*, pages 1–26.

Martin Rosvall and Carl T. Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–23.

Jean Michel Rouly, Huzefa Rangwala, and Aditya Johri. 2015. What are we teaching?: Automated evaluation of CS curricula content using topic modeling. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, pages 189–197.

Mark Sanderson and Bruce Croft. 1999. Deriving concept hierarchies from text. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 206–13, New York, NY, USA. ACM.

Ulrich Schäfer, Bernd Kiefer, Christian Spurk, Jörg Steffen, and Rui Wang. 2011. The ACL Anthology Searchbench. In *Proceedings of the ACL-HLT 2011 System Demonstrations*, pages 7–13.

Laura M. Smith, Linhong Zhu, Kristina Lerman, and Allon G. Percus. 2014. Partitioning networks with node attributes by compressing information flow. *arXiv preprint arXiv:1405.4332*.

Partha Pratim Talukdar and William W. Cohen. 2012. Crowdsourced comprehension: Predicting prerequisite structure in Wikipedia. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 307–15. Association for Computational Linguistics.

Xiaolong Wang, Chengxiang Zhai, and Dan Roth. 2013. Understanding evolution of research themes: A probabilistic generative model for citations. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1115–23, New York, NY, USA. ACM.

Ryen W. White and Joemon M. Jose. 2004. A study of topic similarity measures. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and development in Information Retrieval*, pages 520–1. ACM.

William A. Woods. 1997. Conceptual indexing: A better way to organize knowledge. Technical report, Sun Microsystems, Inc., Mountain View, CA, USA.

Yiming Yang, Hanxiao Liu, Jaime Carbonell, and Wanli Ma. 2015. Concept graph learning from educational data. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 159–68. ACM.

Normalized Log-Linear Interpolation of Backoff Language Models is Efficient

Kenneth Heafield University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB, United Kingdom, kheafiel@inf.ed.ac.uk

Chase Geigle **Sean Massung** **Lane Schwartz** University of Illinois at Urbana-Champaign, 707 S. Mathews Ave., Urbana, IL 61801, United States, {geigle1,massung1,lanes}@illinois.edu

Abstract

We prove that log-linearly interpolated backoff language models can be efficiently and exactly collapsed into a single normalized backoff model, contradicting Hsu (2007). While prior work reported that log-linear interpolation yields lower perplexity than linear interpolation, normalizing at query time was impractical. We normalize the model offline in advance, which is efficient due to a recurrence relationship between the normalizing factors. To tune interpolation weights, we apply Newton's method to this convex problem and show that the derivatives can be computed efficiently in a batch process. These findings are combined in new open-source interpolation tool, which is distributed with KenLM. With 21 out-of-domain corpora, log-linear interpolation yields 72.58 perplexity on TED talks, compared to 75.91 for linear interpolation.

1 Introduction

Log-linearly interpolated backoff language models yielded better perplexity than linearly interpolated models (Klakow, 1998; Gutkin, 2000), but experiments and adoption were limited due the impractically high cost of querying. This cost is due to normalizing to form a probability distribution by brute-force summing over the entire vocabulary for each query. Instead, we prove that the log-linearly interpolated model can be normalized offline in advance and exactly expressed as an ordinary backoff language model. This contradicts Hsu (2007), who claimed that log-linearly interpolated models "cannot be efficiently represented as a backoff n–gram model."

We show that offline normalization is efficient due to a recurrence relationship between the normalizing factors (Whittaker and Klakow, 2002). This forms the basis for our open-source implementation, which is part of KenLM: `https://kheafield.com/code/kenlm/`.

Linear interpolation (Jelinek and Mercer, 1980), combines several language models p_i into a single model p_{L}

$$p_{\text{L}}(w_n \mid w_1^{n-1}) = \sum_i \lambda_i p_i(w_n \mid w_1^{n-1})$$

where λ_i are weights and w_1^n are words. Because each component model p_i is a probability distribution and the non-negative weights λ_i sum to 1, the interpolated model p_{L} is also a probability distribution. This presumes that the models have the same vocabulary, an issue we discuss in §3.1.

A log-linearly interpolated model p_{LL} uses the weights λ_i as powers (Klakow, 1998).

$$p_{\text{LL}}(w_n \mid w_1^{n-1}) \propto \prod_i p_i(w_n \mid w_1^{n-1})^{\lambda_i}$$

The weights λ_i are unconstrained real numbers, allowing parameters to soften or sharpen distributions. Negative weights can be used to divide a mixed-domain model by an out-of-domain model. To form a probability distribution, the product is normalized

$$p_{\text{LL}}(w_n \mid w_1^{n-1}) = \frac{\prod_i p_i(w_n \mid w_1^{n-1})^{\lambda_i}}{Z(w_1^{n-1})}$$

where normalizing factor Z is given by

$$Z(w_1^{n-1}) = \sum_x \prod_i p_i(x \mid w_1^{n-1})^{\lambda_i}$$

The sum is taken over all words x in the combined vocabulary of the underlying models, which can number in the millions or even billions. Computing Z efficiently is a key contribution in this work.

Our proofs assume the component models p_i are backoff language models (Katz, 1987) that memorize probability for seen n–grams and charge a

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 876–886, Berlin, Germany, August 7-12, 2016.

backoff penalty b_i for unseen n–grams.

$$p_i(w_n \mid w_1^{n-1}) = \begin{cases} p_i(w_n \mid w_1^{n-1}) \text{ if } w_1^n \text{ is seen} \\ p_i(w_n \mid w_2^{n-1}) b_i(w_1^{n-1}) \text{ o.w.} \end{cases}$$

While linearly or log-linearly interpolated models can be queried online by querying the component models (Stolcke, 2002; Federico et al., 2008), doing so costs RAM to store duplicated n–grams and CPU time to perform lookups. Log-linear interpolation is particularly slow due to normalizing over the entire vocabulary. Instead, it is preferable to combine the models offline into a single backoff model containing the union of n–grams. Doing so is impossible for linear interpolation (§3.2); SRILM (Stolcke, 2002) and MITLM (Hsu and Glass, 2008) implement an approximation. In contrast, we prove that offline log-linear interpolation requires no such approximation.

2 Related Work

Instead of building separate models then weighting, Zhang and Chiang (2014) show how to train Kneser-Ney models (Kneser and Ney, 1995) on weighted data. Their work relied on prescriptive weights from domain adaptation techniques rather than tuning weights, as we do here.

Our exact normalization approach relies on the backoff structure of component models. Several approximations support general models: ignoring normalization (Chen et al., 1998), noise-contrastive estimation (Vaswani et al., 2013), and self-normalization (Andreas and Klein, 2015). In future work, we plan to exploit the structure of other features in high-quality unnormalized log-linear language models (Sethy et al., 2014).

Ignoring normalization is particularly common in speech recognition and machine translation. This is one of our baselines. Unnormalized models can also be compiled into a single model by multiplying the weighted probabilities and backoffs.[1] Many use unnormalized models because weights can be jointly tuned along with other feature weights. However, Haddow (2013) showed that linear interpolation weights can be jointly tuned by pairwise ranked optimization (Hopkins and May, 2011). In theory, normalized log-linear interpolation weights can be jointly tuned in the same way.

[1]Missing probabilities are found from the backoff algorithm and missing backoffs are implicitly one.

Dynamic interpolation weights (Weintraub et al., 1996) give more weight to models familiar with a given query. Typically the weights are a function of the contexts that appear in the combined language model, which is compatible with our approach. However, normalizing factors would need to be calculated in each context.

3 Linear Interpolation

To motivate log-linear interpolation, we examine two issues with linear interpolation: normalization when component models have different vocabularies and offline interpolation.

3.1 Vocabulary Differences

Language models are normalized with respect to their vocabulary, including the unknown word.

$$\sum_{x \in \text{vocab}(p_1)} p_1(x) = 1$$

If two models have different vocabularies, then the combined vocabulary is larger and the sum is taken over more words. Component models assign their unknown word probability to these new words, leading to an interpolated model that sums to more than one. An example is shown in Table 1.

	p_1	p_2	p_{L}	Zero
<unk>	0.4	0.2	0.3	0.3
A	0.6		0.4	0.3
B		0.8	0.6	0.4
Sum	1	1	1.3	1

Table 1: Linearly interpolating two models p_1 and p_2 with equal weight yields an unnormalized model p_{L}. If gaps are filled with zeros instead, the model is normalized.

To work around this problem, SRILM (Stolcke, 2002) uses zero probability instead of the unknown word probability for new words. This produces a model that sums to one, but differs from what users might expect.

IRSTLM (Federico et al., 2008) asks the user to specify a common large vocabulary size. The unknown word probability is downweighted so that all models sum to one over the large vocabulary.

A component model can also be renormalized with respect to a larger vocabulary. For unigrams, the extra mass is the number of new words times the unknown word probability. For longer contexts, if we assume the typical case where the

unknown word appears only as a unigram, then queries for new words will back off to unigrams. The total mass in context w_1^{n-1} is

$$1 + |new| p(\text{<unk>}) \prod_{i=1}^{n-1} b(w_i^{n-1})$$

where new is the set of new words. This is efficient to compute online or offline. While there are tools to renormalize models, we are not aware of a tool that does this for linear interpolation.

Log-linear interpolation is normalized by construction. Nonetheless, in our experiments we extend IRSTLM's approach by training models with a common vocabulary size, rather than retrofitting it at query time.

3.2 Offline Linear Interpolation

Given an interpolated model, offline interpolation seeks a combined model meeting three criteria: (i) encoding the same probability distribution, (ii) being a backoff model, and (iii) containing the union of n–grams from component models.

Theorem 1. *The three offline criteria cannot be satisfied for general linearly interpolated backoff models.*

Proof. By counterexample. Consider the models given in Table 2 interpolated with equal weight.

	p_1	p_2	p_L
$p(\text{A})$	0.4	0.2	0.3
$p(\text{B})$	0.3	0.3	0.3
$p(\text{C})$	0.3	0.5	0.4
$p(\text{C} \mid \text{A})$	0.4	0.8	0.6
b(A)	$\frac{6}{7} \approx 0.857$	0.4	$\frac{2}{3} \approx 0.667$

Table 2: Counterexample models.

The probabilities shown for p_L result from encoding the same distribution. Taking the union of n–grams implies that p_L only has entries for A, B, C, and A C. Since the models have the same vocabulary, they are all normalized to one.

$$p(\text{A} \mid \text{A}) + p(\text{B} \mid \text{A}) + p(\text{C} \mid \text{A}) = 1$$

Since all models have backoff structure,

$$p(\text{A})b(\text{A}) + p(\text{B})b(\text{A}) + p(\text{C} \mid \text{A}) = 1$$

which when solved for backoff $b(\text{A})$ gives the values shown in Table 2. We then query $p_\text{L}(\text{B} \mid \text{A})$ online and offline. Online interpolation yields

$$\begin{aligned} p_\text{L}(\text{B} \mid \text{A}) &= \frac{1}{2} p_1(\text{B} \mid \text{A}) + \frac{1}{2} p_2(\text{B} \mid \text{A}) \\ &= \frac{1}{2} p_1(\text{B}) b_1(\text{A}) + \frac{1}{2} p_2(\text{B}) b_2(\text{A}) = \frac{33}{175} \end{aligned}$$

Offline interpolation yields

$$p_\text{L}(\text{B} \mid \text{A}) = p_\text{L}(\text{B}) b_\text{L}(\text{A}) = 0.2 \neq \frac{33}{175} \approx 0.189$$

□

The same problem happens with real language models. To understand why, we attempt to solve for the backoff $b_\text{L}(w_1^{n-1})$. Supposing w_1^n is not in either model, we query $p_\text{L}(w_n \mid w_1^{n-1})$ offline

$$\begin{aligned} & p_\text{L}(w_n | w_1^{n-1}) \\ =& p_\text{L}(w_n | w_2^{n-1}) b_\text{L}(w_1^{n-1}) \\ =& (\lambda_1 p_1(w_n | w_2^{n-1}) + \lambda_2 p_2(w_n | w_2^{n-1})) b_\text{L}(w_1^{n-1}) \end{aligned}$$

while online interpolation yields

$$\begin{aligned} & p_\text{L}(w_n | w_1^{n-1}) \\ =& \lambda_1 p_1(w_n | w_1^{n-1}) + \lambda_2 p_2(w_n | w_1^{n-1}) \\ =& \lambda_1 p_1(w_n | w_2^{n-1}) b_1(w_1^{n-1}) + \lambda_1 p_2(w_n | w_2^{n-1}) b_2(w_1^{n-1}) \end{aligned}$$

Solving for $b_\text{L}(w_1^{n-1})$ we obtain

$$\frac{\lambda_1 p_1(w_n | w_2^{n-1}) b_1(w_1^{n-1}) + \lambda_2 p_2(w_n | w_2^{n-1}) b_2(w_1^{n-1})}{\lambda_1 p_1(w_n | w_2^{n-1}) + \lambda_2 p_2(w_n | w_2^{n-1})}$$

which is a weighted average of the backoff weights $b_1(w_1^{n-1})$ and $b_2(w_1^{n-1})$. The weights depend on w_n, so b_L is no longer a function of w_1^{n-1}.

In the SRILM approximation (Stolcke, 2002), probabilities for n–grams that exist in the model are computed exactly. The backoff weights are chosen to produce a model that sums to one. However, newer versions of SRILM (Stolcke et al., 2011) interpolate by ingesting one component model at a time. For example, the first two models are approximately interpolated before adding a third model. An n–gram appearing only in the third model will have an approximate probability. Therefore, the output depends on the order in which users specify models. Moreover, weights were optimized for correct linear interpolation, not the approximation.

Stolcke (2002) find that the approximation actually decreases perplexity, which we also see in the experiments (§6). However, approximation only happens when the model backs off, which is less likely to happen in fluent sentences used for perplexity scoring.

4 Offline Log-Linear Interpolation

Log-linearly interpolated backoff models p_i can be collapsed into a single offline model p_{LL}. The combined model takes the union of n–grams in component models.[2] For those n–grams, it memorizes correct probability p_{LL}.

$$p_{\text{LL}}(w_n \mid w_1^{n-1}) = \frac{\prod_i p_i(w_n \mid w_1^{n-1})^{\lambda_i}}{Z(w_1^{n-1})} \quad (1)$$

When w_1^n does not appear, the backoff $b_{\text{LL}}(w_1^{n-1})$ modifies $p_{\text{LL}}(w_n \mid w_2^{n-1})$ to make an appropriately normalized probability. To do so, it cancels out the shorter query's normalization term $Z(w_2^{n-1})$ then applies the correct term $Z(w_1^{n-1})$. It also applies the component backoff terms.

$$b_{\text{LL}}(w_1^{n-1}) = \frac{Z(w_2^{n-1})}{Z(w_1^{n-1})} \prod_i b_i(w_1^{n-1})^{\lambda_i} \quad (2)$$

Almost by construction, the model satisfies two of our criteria (§3.2): being a backoff model and containing the union of n–grams. However, backoff models require that the backoff weight of an unseen n–gram be implicitly 1.

Lemma 1. *If w_1^{n-1} is unseen in the combined model, then the backoff weight $b_{LL}(w_1^{n-1}) = 1$.*

Proof. Because we have taken the union of entries, w_1^{n-1} is unseen in component models. These components are backoff models, so implicitly $b_i(w_1^{n-1}) = 1 \; \forall i$. Focusing on the normalization term $Z(w_1^{n-1})$,

$$\begin{aligned} Z(w_1^{n-1}) &= \sum_x \prod_i p_i(x \mid w_1^{n-1})^{\lambda_i} \\ &= \sum_x \prod_i p_i(x \mid w_2^{n-1})^{\lambda_i} b_i(w_1^{n-1})^{\lambda_i} \\ &= \sum_x \prod_i p_i(x \mid w_2^{n-1})^{\lambda_i} \\ &= Z(w_2^{n-1}) \end{aligned}$$

All of the models back off because $w_1^{n-1}x$ is unseen, being a superstring of w_1^{n-1}. Relevant backoff weights $b_i(w_1^{n-1}) = 1$ as noted earlier. Recalling the definition of $b_{\text{LL}}(w_1^{n-1})$,

$$\frac{Z(w_2^{n-1})}{Z(w_1^{n-1})} \prod_i b_i(w_1^{n-1})^{\lambda_i} = \frac{Z(w_2^{n-1})}{Z(w_1^{n-1})} = 1 \quad \square$$

[2] We further assume that every substring of a seen n–gram is also seen. This follows from estimating on text, except in the case of adjusted count pruning by SRILM. In such cases, we add the missing entries to component models, with no additional memory cost in trie data structures.

We now have a backoff model containing the union of n–grams. It remains to show that the offline model produces correct probabilities.

Theorem 2. *The proposed offline model agrees with online log-linear interpolation.*

Proof. By induction on the number of words backed off in offline interpolation. To disambiguate, we will use p_{on} to refer to online interpolation and p_{off} to refer to offline interpolation.
Base case: the queried n–gram is in the offline model and we have memorized the online probability by construction.
Inductive case: Let $p_{\text{off}}(w_n \mid w_1^{n-1})$ be a query that backs off. In online interpolation,

$$p_{\text{on}}(w_n \mid w_1^{n-1}) = \frac{\prod_i p_i(w_n \mid w_1^{n-1})^{\lambda_i}}{Z(w_1^{n-1})}$$

Because w_1^n is unseen in the offline model and we took the union, it is unseen in every model p_i.

$$\begin{aligned} &= \frac{\prod_i p_i(w_n \mid w_2^{n-1})^{\lambda_i} b_i(w_1^{n-1})^{\lambda_i}}{Z(w_1^{n-1})} \\ &= \frac{\left(\prod_i p_i(w_n \mid w_2^{n-1})^{\lambda_i}\right) \prod_i b_i(w_1^{n-1})^{\lambda_i}}{Z(w_1^{n-1})} \end{aligned}$$

Recognizing the unnormalized probability $Z(w_2^{n-1})p_{\text{on}}(w_n \mid w_2^{n-1})$,

$$\begin{aligned} &= \frac{Z(w_2^{n-1})p_{\text{on}}(w_n \mid w_2^{n-1}) \prod_i b_i(w_1^{n-1})^{\lambda_i}}{Z(w_1^{n-1})} \\ &= p_{\text{on}}(w_n \mid w_2^{n-1}) \frac{Z(w_2^{n-1})}{Z(w_1^{n-1})} \prod_i b_i(w_1^{n-1})^{\lambda_i} \\ &= p_{\text{on}}(w_n \mid w_2^{n-1}) b_{\text{off}}(w_1^{n-1}) \end{aligned}$$

The last equality follows from the definition of b_{off} and Lemma 1, which extended the domain of b_{off} to any w_1^{n-1}. By the inductive hypothesis, $p_{\text{on}}(w_n \mid w_2^{n-1}) = p_{\text{off}}(w_n \mid w_2^{n-1})$ because it backs off one less time.

$$= p_{\text{off}}(w_n \mid w_2^{n-1}) b_{\text{off}}(w_1^{n-1}) = p_{\text{off}}(w_n \mid w_1^{n-1})$$

The offline model $p_{\text{off}}(w_n \mid w_1^{n-1})$ backs off because that is the case we are considering. Combining our chain of equalities,

$$p_{\text{on}}(w_n \mid w_1^{n-1}) = p_{\text{off}}(w_n \mid w_1^{n-1})$$

By induction, the claim holds for all w_1^n. $\square$

4.1 Normalizing Efficiently

In order to build the offline model, the normalization factor Z needs to be computed in every seen context. To do so, we extend the tree-structure method of Whittaker and Klakow (2002), which they used to compute and cache normalization factors on the fly. It exploits the sparsity of language models: when summing over the vocabulary, most queries will back off. Formally, we define $s(w_1^n)$ to be the set of words x where $p_i(x \mid w_1^n)$ does not back off in some model.

$$s(w_1^n) = \{x : w_1^n x \text{ is seen in any model}\}$$

To exploit this, we use the normalizing factor $Z(w_2^n)$ from a lower order and patch it up by summing over $s(w_1^n)$.

Theorem 3. *The normalization factors Z obey a recurrence relationship:*

$$Z(w_1^n) = \sum_{x \in s(w_1^n)} \prod_i p_i(x \mid w_1^n)^{\lambda_i} + \left(Z(w_2^n) - \sum_{x \in s(w_1^n)} \prod_i p_i(x \mid w_2^n)^{\lambda_i} \right) \prod_i b_i(w_1^n)^{\lambda_i}$$

Proof. The first term handles seen n–grams while the second term handles unseen n–grams. The definition of Z

$$Z(w_1^n) = \sum_{x \in \text{vocab}} \prod_i p_i(x \mid w_1^n)^{\lambda_i}$$

can be partitioned by cases.

$$\sum_{x \in s(w_1^n)} \prod_i p_i(x \mid w_1^n)^{\lambda_i} + \sum_{x \notin s(w_1^n)} \prod_i p_i(x \mid w_1^n)^{\lambda_i}$$

The first term agrees with the claim, so we focus on the case where $x \notin s(w_1^n)$. By definition of s, all models back off.

$$\begin{aligned}
&\sum_{x \notin s(w_1^n)} \prod_i p_i(x \mid w_1^n)^{\lambda_i} \\
&= \sum_{x \notin s(w_1^n)} \prod_i p_i(x \mid w_2^n)^{\lambda_i} b_i(w_1^n)^{\lambda_i} \\
&= \left(\sum_{x \notin s(w_1^n)} \prod_i p_i(x \mid w_2^n)^{\lambda_i} \right) \prod_i b_i(w_1^n)^{\lambda_i} \\
&= \left(Z(w_2^n) - \sum_{x \in s(w_1^n)} \prod_i p_i(x \mid w_2^n)^{\lambda_i} \right) \prod_i b_i(w_1^n)^{\lambda_i}
\end{aligned}$$

This is the second term of the claim. □

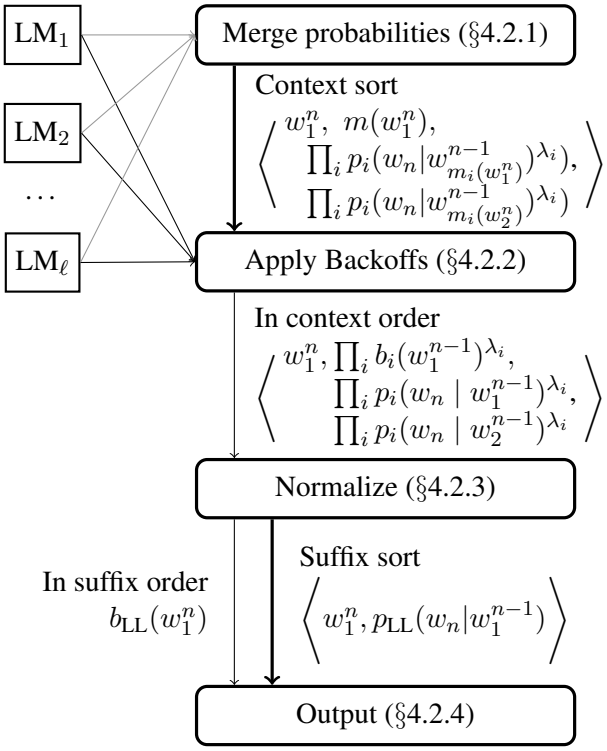


Figure 1: Multi-stage streaming pipeline for offline log-linear interpolation. Bold arrows indicate sorting is performed.

The recurrence structure of the normalization factors suggests a computational strategy: compute $Z(\epsilon)$ by summing over the unigrams, $Z(w_n)$ by summing over bigrams $w_n x$, $Z(w_{n-1}^n)$ by summing over trigrams $w_{n-1}^n x$, and so on.

4.2 Streaming Computation

Part of the point of offline interpolation is that there may not be enough RAM to fit all the component models. Moreover, with compression techniques that rely on immutable models (Whittaker and Raj, 2001; Talbot and Osborne, 2007), a mutable version of the combined model may not fit in RAM. Instead, we construct the offline model with disk-based streaming algorithms, using the framework we designed for language model estimation (Heafield et al., 2013). Our pipeline (Figure 1) has four conceptual steps: merge probabilities, apply backoffs, normalize, and output. Applying backoffs and normalization are performed in the same pass, so there are three total passes.

4.2.1 Merge Probabilities

This step takes the union of n–grams and multiplies probabilities from component models. We

assume that the component models are sorted in suffix order (Figure 4), which is true of models produced by `lmplz` (Heafield et al., 2013) or stored in a reverse trie. Moreover, despite having different word indices, the models are consistently sorted using the string word, or a hash thereof.

3	**2**	**1**
		A
	A	A
A	A	A
B	A	A
		B

Table 3: Merging probabilities processes n–grams in lexicographic order by suffix. Column headings indicate precedence.

The algorithm processes n–grams in lexicographic (depth-first) order by suffix (Table 3). In this way, the algorithm processes $p_i(A)$ before it might be used as a backoff point for $p_i(A \mid B)$ in one of the models. It jointly streams through all models, so that $p_1(A \mid B)$ and $p_2(A \mid B)$ are available at the same time. Ideally, we would compute unnormalized probabilities.

$$\prod_i p_i(w_n \mid w_1^{n-1})^{\lambda_i}$$

However, these queries back off when models contain different n–grams. The appropriate backoff weights $b_i(w_1^{n-1})$ are not available in a streaming fashion. Instead, we proceed without charging backoffs

$$\prod_i p_i(w_n \mid w_{m_i(w_1^n)}^{n-1})^{\lambda_i}$$

where $m_i(w_1^n)$ records what backoffs should be charged later.

The normalization step (§4.2.3) also uses lower-order probabilities

$$\prod_i p_i(w_n \mid w_2^{n-1})^{\lambda_i}$$

and needs to access them in a streaming fashion, so we also output

$$\prod_i p_i(w_n \mid w_{m_i(w_2^n)}^{n-1})^{\lambda_i}$$

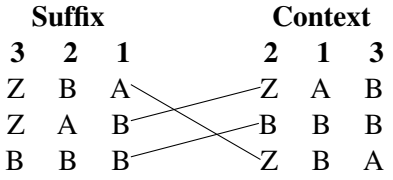


Table 4: Sorting orders (Heafield et al., 2013). In suffix order, the last word is primary. In context order, the penultimate word is primary. Column headings indicate precedence.

Each output tuple has the form

$$\left\langle w_1^n, m(w_1^n), \prod_i p_i(w_n|w_{m_i(w_1^n)}^{n-1})^{\lambda_i}, \prod_i p_i(w_n|w_{m_i(w_2^n)}^{n-1})^{\lambda_i} \right\rangle$$

where $m(w_1^n)$ is a vector of backoff requests, from which $m(w_2^n)$ can be computed.

4.2.2 Apply Backoffs

This step fulfills the backoff requests from merging probabilities. The merged probabilities are sorted in context order (Table 4) so that n–grams w_1^n sharing the same context w_1^{n-1} are consecutive. Moreover, contexts w_1^{n-1} appear in suffix order. We use this property to stream through the component models again in their native suffix order, this time reading backoff weights $b_i(w_1^{n-1}), b_i(w_2^{n-1}), \ldots, b_i(w_{n-1})$. Multiplying the appropriate backoff weights by $\prod_i p_i(w_n|w_{m_i(w_1^n)}^{n-1})^{\lambda_i}$ yields unnormalized probability

$$\prod_i p_i(w_n|w_1^{n-1})^{\lambda_i}$$

The same applies to the lower order.

$$\prod_i p_i(w_n|w_2^{n-1})^{\lambda_i}$$

This step also merges backoffs from component models, with output still in context order.

$$\left\langle w_1^n, \prod_i b_i(w_1^{n-1})^{\lambda_i}, \prod_i p_i(w_n|w_1^{n-1})^{\lambda_i} \prod_i p_i(w_n|w_2^{n-1})^{\lambda_i} \right\rangle$$

The implementation is combined with normalization, so the tuple is only conceptual.

4.2.3 Normalize

This step computes normalization factor Z for all contexts, which it applies to produce p_{LL} and b_{LL}. Recalling §4.1, $Z(w_1^{n-1})$ is efficient to compute in a batch process by processing suffixes $Z(\epsilon), Z(w_n), \ldots Z(w_2^{n-1})$ first. In order to minimize memory consumption, we chose to evaluate the contexts in depth-first order by suffix, so that $Z(\text{A})$ is computed immediately before it is needed to compute $Z(\text{A A})$ and forgotten at $Z(\text{B})$.

Computing $Z(w_1^{n-1})$ by applying Theorem 3 requires the sum

$$\sum_{x \in s(w_1^{n-1})} \prod_i p_i(x \mid w_1^{n-1})^{\lambda_i}$$

where $s(w_1^{n-1})$ restricts to seen n–grams. For this, we stream through the output of the apply backoffs step in context order, which makes the various values of x consecutive. Theorem 3 also requires a sum over the lower-order unnormalized probabilities

$$\sum_{x \in s(w_1^{n-1})} \prod_i p_i(x \mid w_2^{n-1})^{\lambda_i}$$

We placed these terms in the input tuple for $w_1^{n-1}x$. Otherwise, it would be hard to access these values while streaming in context order.

While we have shown how to compute $Z(w_1^{n-1})$, we still need to normalize the probabilities. Unfortunately, $Z(w_1^{n-1})$ is only known after streaming through all records of the form $w_1^{n-1}x$, which are the very same records to normalize. We therefore buffer up to the vocabulary size for each order in memory to allow rewinding. Processing context w_1^{n-1} thus yields normalized probabilities $p_{\text{LL}}(x \mid w_1^{n-1})$ for all seen $w_1^{n-1}x$.

$$\left\langle w_1^n, p_{\text{LL}}(x \mid w_1^{n-1}) \right\rangle$$

These records are generated in context order, the same order as the input.

The normalization step also computes backoffs.

$$b_{\text{LL}}(w_1^{n-1}) = \frac{Z(w_2^{n-1})}{Z(w_1^{n-1})} \prod_i b_i(w_1^{n-1})^{\lambda_i}$$

Normalization $Z(w_1^{n-1})$ is computed by this step, numerator $Z(w_2^{n-1})$ is available due to depth-first search, and the backoff terms $\prod_i b_i(w_1^{n-1})^{\lambda_i}$ are present in the input. The backoffs b_{LL} are generated in suffix order, since each context produces a backoff value. These are written to a sidechannel stream as bare values without keys.

4.2.4 Output

Language model toolkits store probability $p_{\text{LL}}(w_n \mid w_1^{n-1})$ and backoff $b_{\text{LL}}(w_1^n)$ together as values for the key w_1^n. To reunify them, we sort $\langle w_1^n, p_{\text{LL}}(w_n \mid w_1^{n-1}) \rangle$ in suffix order and merge with the backoff sidechannel from normalization, which is already in suffix order. Suffix order is also preferable because toolkits can easily build a reverse trie data structure.

5 Tuning

Weights are tuned to maximize the log probability of held-out data. This is a convex optimization problem (Klakow, 1998). Iterations are expensive due to the need to normalize over the vocabulary at least once. However, the number of weights is small, which makes the Hessian matrix cheap to store and invert. We therefore selected Newton's method.[3]

The log probability of tuning data w is

$$\log \prod_n p_{\text{LL}}(w_n \mid w_1^{n-1})$$

which expands according to the definition of p_{LL}

$$\sum_n \left(\sum_i \lambda_i \log p_i(w_n \mid w_1^{n-1}) \right) - \log Z(w_1^{n-1})$$

The gradient with respect to λ_i has a compact form

$$\sum_n \log p_i(w_n \mid w_1^{n-1}) + \text{CH}(p_{\text{LL}}, p_i \mid w_1^{n-1})$$

where CH is cross entropy. However, computing the cross entropy directly would entail a sum over the vocabulary for every word in the tuning data. Instead, we apply Theorem 3 to express $Z(w_1^{n-1})$ in terms of $Z(w_2^{n-1})$ before taking the derivative. This allows us to perform the same depth-first computation as before (§4.2.3), only this time $\frac{\partial}{\partial \lambda_i} Z(w_1^{n-1})$ is computed in terms of $\frac{\partial}{\partial \lambda_i} Z(w_2^{n-1})$.

The same argument applies when taking the Hessian with respect to λ_i and λ_j. Rather than compute it directly in the form

$$\sum_n -\sum_x p_{\text{LL}}(x|w_1^{n-1}) \log p_i(x|w_1^{n-1}) \log p_j(x|w_1^{n-1}) + \text{CH}(p_{\text{LL}}, p_i \mid w_1^{n-1})\text{CH}(p_{\text{LL}}, p_j \mid w_1^{n-1})$$

we apply Theorem 3 to compute the Hessian for w_1^n in terms of the Hessian for w_2^n.

[3] We also considered minibatches, though grouping tuning data to reduce normalization cost would introduce bias.

6 Experiments

We perform experiments for perplexity, query speed, memory consumption, and effectiveness in a machine translation system.

Individual language models were trained on English corpora from the WMT 2016 news translation shared task (Bojar et al., 2016). This includes the seven newswires (afp, apw, cna, ltw, nyt, wpb, xin) from English Gigaword Fifth Edition (Parker et al., 2011); the 2007–2015 news crawls;[4] News discussion; News commmentary v11; English from Europarl v8 (Koehn, 2005); the English side of the French-English parallel corpus (Bojar et al., 2013); and the English side of SETIMES2 (Tiedemann, 2009). We additionally built one language model trained on the concatenation of all of the above corpora. All corpora were preprocessed using the standard Moses (Koehn et al., 2007) scripts to perform normalization, tokenization, and truecasing. To prevent SRILM from running out of RAM, we excluded the large monolingual CommonCrawl data, but included English from the parallel CommonCrawl data.

All language models are 5-gram backoff language models trained with modified Kneser-Ney smoothing (Chen and Goodman, 1998) using `lmplz` (Heafield et al., 2013). Also to prevent SRILM from running out of RAM, we pruned singleton trigrams and above.

For linear interpolation, we tuned weights using IRSTLM. To work around SRILM's limitation of ten models, we interpolated the first ten then carried the combined model and added nine more component models, repeating this last step as necessary. Weights were normalized within batches to achieve the correct final weighting. This simply extends the way SRILM internally carries a combined model and adds one model at a time.

6.1 Perplexity experiments

We experiment with two domains: TED talks, which is out of domain, and news, which is in-domain for some corpora. For TED, we tuned on the IWSLT 2010 English dev set and test on the 2010 test set. For news, we tuned on the English side of the WMT 2015 Russian–English evaluation set and test on the WMT 2014 Russian–English evaluation set. To measure generalization, we also evaluated news on models tuned for TED and vice-versa. Results are shown in Table 5.

[4]For News Crawl 2014, we used version 2.

	Component Models	
Component	**TED test**	**News test**
Gigaword afp	163.48	221.57
Gigaword apw	140.65	206.85
Gigaword cna	299.93	448.56
Gigaword ltw	106.28	243.35
Gigaword nyt	97.21	211.75
Gigaword wpb	151.81	341.48
Gigaword xin	204.60	246.32
News 07	127.66	243.53
News 08	112.48	202.86
News 09	111.43	197.32
News 10	114.40	209.31
News 11	107.69	187.65
News 12	105.74	180.28
News 13	104.09	155.89
News 14 v2	101.85	139.94
News 15	101.13	141.13
News discussion	99.88	249.63
News commentary v11	236.23	495.27
Europarl v8	268.41	574.74
CommonCrawl fr-en.en	149.10	343.20
SETIMES2 ro-en.en	331.37	521.19
All concatenated	80.69	96.15

	TED weights	
Interpolation	**TED test**	**News test**
Offline linear	75.91	100.43
Online linear	76.93	152.37
Log-linear	72.58	112.31

	News weights	
Interpolation	**TED test**	**News test**
Offline linear	83.34	107.69
Online linear	83.94	110.95
Log-linear	89.62	124.63

Table 5: Test set perplexities. In the middle table, weights are optimized for TED and include a model trained on all concatenated text. In the bottom table, weights are optimized for news and exclude the model trained on all concatenated text.

LM	Tuning		Compiling		Querying	
All concatenated	N/A	N/A	N/A	N/A	0.186μs	46.7G
Offline linear	0.876m	60.2G	641m	123G	0.186μs	46.8G
Online linear	0.876m	60.2G	N/A	N/A	5.67μs	89.1G
Log-linear	600m	63.9G	89.8m	63.9G	0.186μs	46.8G

Table 6: Speed and memory consumption of LM combination methods. Interpolated models include the concatenated model. Tuning and compiling times are in minutes, memory consumption in gigabytes, and query time in microseconds per query (on 1G of held-out Common Crawl monolingual data).

Log-linear interpolation performs better on TED (72.58 perplexity versus 75.91 for offline linear interpolation). However, it performs worse on news. In future work, we plan to investigate whether log-linear wins when all corpora are out-of-domain since it favors agreement by all models.

Table 6 compares the speed and memory performance of the competing methods. While the log-linear tuning is much slower, its compilation is faster compared to the offline linear model's long run time. Since the model formats are the same for the concatenation and log-linear, they share the fastest query speeds. Query speed was measured using KenLM's (Heafield, 2011) faster probing data structure.[5]

6.2 MT experiments

We trained a statistical phrase-based machine translation system for Romanian-English on the Romanian-English parallel corpora released as part of the 2016 WMT news translation shared task. We trained three variants of this MT system. The first used a single language model trained on the concatenation of the 21 individual LM training corpora. The second used 22 language models, with each LM presented to Moses as a separate feature. The third used a single language model which is an interpolation of all 22 models. This variant was run with offline linear, online linear, and log-linear interpolation. All MT system variants were optimized using IWSLT 2011 Romanian-English TED test as the development set, and were evaluated using the IWSLT 2012 Romanian-English TED test set.

As shown in Table 7, no significant difference in MT quality as measured by BLEU was observed; the best BLEU score came from separate features at 18.40 while log-linear scored 18.15. We leave jointly tuned normalized log-linear interpolation to future work.

[5]KenLM does not natively implement online linear interpolation, so we wrote a custom wrapper, which is faster than querying IRSTLM.

LM	BLEU	BLEU-c
22 separate LMs	18.40	17.91
All concatenated	18.02	17.55
Offline linear	18.00	17.53
Online linear	18.27	17.82
Log-linear	18.15	17.70

Table 7: Machine translation performance comparison in an end-to-end system.

7 Conclusion

Normalized log-linear interpolation is now a tractable alternative to linear interpolation for backoff language models. Contrary to Hsu (2007), we proved that these models can be exactly collapsed into a single backoff language model. This solves the query speed problem. Empirically, compiling the log-linear model is faster than SRILM can collapse its approximate offline linear model. In future work, we plan to improve performace of feature weight tuning and investigate more general features.

Acknowledgments

Thanks to João Sedoc, Grant Erdmann, Jeremy Gwinnup, Marcin Junczys-Dowmunt, Chris Dyer, Jon Clark, and MT Marathon attendees for discussions. Partial funding was provided by the Amazon Academic Research Awards program. This material is based upon work supported by the NSF GRFP under Grant Number DGE-1144245.

References

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *NAACL 2015*.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 workshop on statistical machine translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Rajen Chatterjee, Christian Federmann, Liane Guillou, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Aurélie Névéol, Mariana Neves, Pavel Pecina, Martin Popel, Philipp Koehn, Christof Monz, Matteo Negri, Matt Post, Lucia Specia, Karin Verspoor, Jörg Tiedemann, and Marco Turchi. 2016. Findings of the 2016 Conference on Machine Translation. In *Proceedings of the First Conference on Machine Translation (WMT'16)*, Berlin, Germany, August.

Stanley Chen and Joshua Goodman. 1998. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, August.

Stanley F. Chen, Kristie Seymore, and Ronald Rosenfeld. 1998. Topic adaptation for language modeling using unnormalized exponential models. In *Acoustics, Speech and Signal Processing, 1998. Proceedings of the 1998 IEEE International Conference on*, volume 2, pages 681–684. IEEE.

Marcello Federico, Nicola Bertoldi, and Mauro Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech*, Brisbane, Australia.

Alexander Gutkin. 2000. Log-linear interpolation of language models. Master's thesis, University of Cambridge, November.

Barry Haddow. 2013. Applying pairwise ranked optimisation to improve the interpolation of translation models. In *Proceedings of NAACL*.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, August.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, Edinburgh, UK, July. Association for Computational Linguistics.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352—-1362, Edinburgh, Scotland, July.

Bo-June Hsu and James Glass. 2008. Iterative language model estimation: Efficient data structure & algorithms. In *Proceedings of Interspeech*, Brisbane, Australia.

Bo-June Hsu. 2007. Generalized linear interpolation of language models. In *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*, pages 136–140. IEEE.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, May.

Slava Katz. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-35(3):400–401, March.

Dietrich Klakow. 1998. Log-linear interpolation of language models. In *Proceedings of 5th International Conference on Spoken Language Processing*, pages 1695–1699, Sydney, November.

Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 181–184.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, June.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit*.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English gigaword fifth edition, June. LDC2011T07.

Abhinav Sethy, Stanley Chen, Bhuvana Ramabhadran, and Paul Vozila. 2014. Static interpolation of exponential n–gram models using features of features. In *2014 IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP)*.

Andreas Stolcke, Jing Zheng, Wen Wang, and Victor Abrash. 2011. SRILM at sixteen: Update and outlook. In *Proc. 2011 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, Waikoloa, Hawaii, USA.

Andreas Stolcke. 2002. SRILM - an extensible language modeling toolkit. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 901–904.

David Talbot and Miles Osborne. 2007. Randomised language modelling for statistical machine translation. In *Proceedings of ACL*, pages 512–519, Prague, Czech Republic.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of EMNLP*.

Mitch Weintraub, Yaman Aksu, Satya Dharanipragada, Sanjeev Khudanpur, Hermann Ney, John Prange, Andreas Stolcke, Fred Jelinek, and Liz Shriberg. 1996. LM95 project report: Fast training and portability. Research Note 1, Center for Language and Speech Processing, Johns Hopkins University, February.

Edward D. W. Whittaker and Dietrich Klakow. 2002. Efficient construction of long-range language models using log-linear interpolation. In *7th International Conference on Spoken Language Processing*, pages 905–908.

Edward Whittaker and Bhiksha Raj. 2001. Quantization-based language model compression. In *Proceedings of Eurospeech*, pages 33–36, Aalborg, Denmark, December.

Hui Zhang and David Chiang. 2014. Kneser-Ney smoothing on expected counts. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 765–774. ACL.

How Well Do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation

Danqing Huang[1]*, **Shuming Shi**[2], **Chin-Yew Lin**[2], **Jian Yin**[1] and **Wei-Ying Ma**[2]
[1] Sun Yat-sen University
{huangdq2@mail2,issjyin@mail}.sysu.edu.cn
[2] Microsoft Research
{shumings,cyl,wyma}@microsoft.com

Abstract

Recently a few systems for automatically solving math word problems have reported promising results. However, the datasets used for evaluation have limitations in both scale and diversity. In this paper, we build a large-scale dataset which is more than 9 times the size of previous ones, and contains many more problem types. Problems in the dataset are semi-automatically obtained from community question-answering (CQA) web pages. A ranking SVM model is trained to automatically extract problem answers from the answer text provided by CQA users, which significantly reduces human annotation cost. Experiments conducted on the new dataset lead to interesting and surprising results.

1 Introduction

Designing computer systems for automatically solving math word problems is a challenging research topic that dates back to the 1960s (Bobrow, 1964a; Briars and Larkin, 1984; Fletcher, 1985). As early proposals seldom report empirical evaluation results, it is unclear how well they perform. Recently, promising results have been reported on both statistical learning approaches (Kushman et al., 2014; Hosseini et al., 2014; Koncel-Kedziorski et al., 2015; Zhou et al., 2015; Roy and Roth, 2015) and semantic parsing methods (Shi et al., 2015). However, we observe two limitations on the datasets used by these previous works. First, the datasets are small. The most frequently used dataset (referred to as Alg514 hereafter) only contains 514 algebra problems. The Dolphin1878 dataset (Shi et al., 2015), the largest collection among them, contains 1878 problems. Second, the diversity of problems in the datasets is low. The Alg514 collection contains linear algebra problems of 28 types (determined by 28 unique equation systems), with each problem type corresponding to at least 6 problems. Although the Dolphin1878 collection has over 1,000 problem types, only number word problems (i.e., math word problems about the operations and relationship of numbers) are contained in the collection.

Due to the above two limitations, observations and conclusions based on existing datasets may not be representative. Therefore it is hard to give a convincing answer to the following question: How well do state-of-the-art computer algorithms perform in solving math word problems?

To answer this question, we need to re-evaluate state-of-the-art approaches on a larger and more diverse data set. It is not hard to collect a large set of problems from the web. The real challenge comes from attaching annotations to the problems. Important annotation types include equation systems (required by most statistical learning methods for model training) and gold answers (for testing algorithm performance). Manually adding equation systems and gold answers is extremely time-consuming[1].

In this paper, we build a large-scale and diverse dataset called Dolphin18K [2], which contains over 18,000 annotated math word problems. It is constructed by semi-automatically extracting problems, equation systems and answers from community question-answering (CQA) web pages. The source data we leverage are the (question, answer text) pairs in the math category of Yahoo! An-

*Work done while this author was an intern at Microsoft Research.

[1] According to our experience, the speed is about 10-15 problems per hour for a person with good math skills.

[2] Available from http://research.microsoft.com/en-us/projects/dolphin/.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 887–896,
Berlin, Germany, August 7-12, 2016.

swers[3]. Please note that the answer text provided by CQA users cannot be used directly in evaluation as gold answers, because answer numbers and other numbers are often mixed together in answer text (refer to Figure 1 of Section 3). We train a ranking SVM model to identify (structured) problem answers from unstructured answer text.

We then conduct experiments to test the performance of some recent math problem solving systems on the dataset. We make the following main observations,

1. All systems evaluated on the Dolphin18K dataset perform much worse than on their original small and less diverse datasets.

2. On the large dataset, a simple similarity-based method performs as well as more sophisticated statistical learning approaches.

3. System performance improves sub-linearly as more training data is used. This suggests that we need to develop algorithms which can utilize data more effectively.

Our experiments indicate that the problem of automatic math word problem solving is still far from being solved. Good results obtained on small datasets may not be good indicators of high performance on larger and diverse datasets. For current methods, simply adding more training data is not an effective way to improve performance. New methodologies are required for this topic.

2 Related Work

2.1 Math Word Problem Solving

Previous work on automatic math word problem solving falls into two categories: symbolic approaches and statistical learning methods. In symbolic approaches (Bobrow, 1964a; Bobrow, 1964b; Charniak, 1968; Charniak, 1969; Bakman, 2007; Liguda and Pfeiffer, 2012; Shi et al., 2015), math problem sentences are transformed to certain structures (usually trees) by pattern matching, verb categorization, or semantic parsing. Math equations are then derived from the structured representation. Addition/subtraction problems are studied most in early research (Briars and Larkin, 1984; Fletcher, 1985; Dellarosa, 1986; Bakman, 2007; Yuhui et al., 2010). Please refer to Mukherjee and Garain (2008) for a review of symbolic approaches before 2008.

[3]https://answers.yahoo.com/

Statistical machine learning methods have been proposed to solve math word problems since 2014. Hosseini et al. (2014) solve single step or multi-step homogeneous addition and subtraction problems by learning verb categories from the training data. Kushman et al. (2014) and Zhou et al. (2015) solve a wide range of algebra word problems, given that systems of linear equations are attached to problems in the training set. Seo et al. (2015) focuses on SAT geometry questions with text and diagram provided. Koncel-Kedziorski et al. (2015) and Roy and Roth (2015) target math problems that can be solved by one single linear equation.

No empirical evaluation results are reported in most early publications on this topic. Although promising empirical results are reported in recent work, the datasets employed in their evaluation are small and lack diversity. For example, the Alg514 dataset used in Kushman et al. (2014) and Zhou et al. (2015) only contains 514 problems of 28 types. Please refer to Section 3.4 for more details about the datasets.

Recently, a framework was presented in Koncel-Kedziorsk et al. (2016) for building an online repository of math word problems. The framework is initialized by including previous public available datasets. The largest dataset among them contains 1,155 problems.

2.2 Answer Extraction in CQA

Our work on automatic answer and equation extraction is related to the recent CQA extraction work (Agichtein et al., 2008; Cong et al., 2008; Ding et al., 2008). Most of them aim to discover high-quality (question, answer text) pairs from CQA posts. We are different because we extract structured data (i.e., numbers and equation systems) *inside* the pieces of answer text.

3 Dataset Construction

Our goal is to construct a large and diverse problem collection of elementary mathematics (i.e., math topics frequently taught at the primary or secondary school levels). We build our dataset by automatically extracting problems and their annotations from the mathematics category of the Yahoo! Answers web site. A math problem post on Yahoo! Answers consists of the raw problem text and one or multiple pieces of answer text provided by its answerers (refer to Figure 1).

Please note that posts cannot be used directly as our dataset entries. For example, for training statistical models, we have to extract equation systems from the unstructured text of user answers. We also need to extract numbers (56,000 and 21,000 in Figure 1) from the raw answer text as gold answers. We perform the following actions to the posts,

- Removing the posts that do not contain a math problem of our scope (Section 3.1)
- Cleaning problem text (Section 3.1)
- Extracting gold answers (Section 3.2)
- Extracting equation systems (Section 3.3)

In Section 3.4, we report some statistics of our dataset and compare them with previous ones.

3.1 Preprocessing

We crawl over one million posts from the mathematics categories of Yahoo! Answers. They are part of the posts submitted and answered by users since year 2008. By examining some examples, we soon find that many of them do not contain math problems of our scope. We discard or ignore the posts with the following types:

1. Containing a general math-related question but not a typical math problem. For example, "*Can anyone teach me how to set up two equations for one problem, and then how to solve it after?*".
2. College-level math
3. Containing multiple math problems in a single post. They are discarded for simplifying our answer and equation system extraction process.

As the size of a set of one million problems is large for human annotation and many of them belong to the above three types, we need a way to automatically filter out undesired problems. We manually annotate 6,000 posts with the speed of about 150 posts per hour per person. Then a logistic regression classifier of posts is trained with a precision of 80% and a recall of 70%. The post collection after classification is reduced to 120,000 posts.

Then we randomly sample 46,000 posts from the reduced post collection to perform two actions manually: post classification and problem text cleaning. Please note that, since the precision of the automatic classifier is only 80%, we rely on manual classification to remove the remaining 20% undesired posts. Problem text cleaning is for removing sentences like "please help" and "Son's 6th grade math" (refer to Figure 1). The problem text after cleaning is just like that appearing in a formal math test in an elementary or secondary school.

Question part:
Son's 6th grade math? The number of cans produced in one day by two companies A and B were in ratio 8:3 and their difference was 35,000. How many cans did each company produce that day?

Answer part:
Answer 1: Let can produced by 1 company be 8x and the other 3x. so 8x - 3x = 35000. 5x = 35000. x = 7000. So the first company produced 8 x 7000 = 56,000 cans, and the other produced 3 x 7000 = 21,000 cans.
Answer 2: From the ratio: 3A=8B or A=(8/3)B. From the difference: A-B=35000. By substituting for A, we get (8/3)B-B=35000 and further to B = 21000. From the difference: A = 21000+35000=56000.
Answer 3: It's 56000 and 21000.
Answer 4: what the hell thats not 6th grade math!!!

Figure 1: An example post from Yahoo! answers

Eight annotators participated in the manual post classification and problem text cleaning, at an average speed of about 80 posts per hour per person.

3.2 Automatic Answer Extraction

Compared to post classification and problem text cleaning, it is much more time consuming to manually assign gold answers and equation systems to a problem (10-15 problems per hour per person vs. 80 posts per hour per person). In addition, the latter has higher requirements of the math skills of annotators. Since manually annotating all problems exceeds our budget, we choose to train a high precision model to automatically extract numbers as gold answers from the answer part of a post.

In our dataset, the gold answer to a problem is one or a set of numbers acting as the solution to the problem. We define *answer dimension* as the

count of numbers required in the gold answer. For example, the gold answer to the problem in Figure 1 is $\{56000, 21000\}$, with dimension 2.

Extracting gold answers from the answer part of a post is nontrivial. We tried an intuitive approach called last-number-majority-voting, where the last number in each answer of the post is chosen as a candidate and then majority voting is performed among all the candidates. We got a low accuracy of 47% on our annotated data. Thus, we turn to a machine learning model for better utilizing more features in the posts.

Notations: Let χ denote the set of training problems. For each problem x_i in χ, $N_{ij} = \{n_{ij}^1, n_{ij}^2, \ldots, n_{ij}^m\}$ denote the set of all unique numbers given the *j*th answer, where *m* represents the size of N_{ij}. For each N_{ij}, we generate possible subsets of numbers as candidate answers to the problems. Please pay attention that the gold answer to a problem may contain multiple numbers (in the case that the answer dimension is larger than 1). We use Y_i to denote all the candidate answers in problem x_i.

Model: We define the conditional probability of $y_{ik} \in Y_i$ given x_i:

$$p(y_{ik}|x_i;\nu) = \frac{\exp(\nu \cdot f(x_i, y_{ik}))}{\sum_{y'_{ik} \in Y_i} \exp(\nu \cdot f(x_i, y'_{ik}))}$$

where ν is a parameter vector of the model and $f(x_i, y_{ik})$ is the feature vector. We apply the Ranking SVM (Herbrich et al., 2000) to maximize the margin between the correct instances and the negative ones. Constructing the SVM model is equivalent to solving the following Quadratic Optimization problem:

$$\min_{\nu} M(\nu) = \frac{1}{2}\|\nu\|^2 + C\sum_i \xi_i$$

$$\textit{s.t. } \xi_i \geq 0, \nu \cdot \langle f(x_i, y_{ik})^+ - f(x_i, y_{il})^- \rangle \geq 1 - \xi_i$$

where subscript "+" indicates the correct instance and "-" indicates the false ones.

Features: Features are extracted from the answer part of each post for model training. We design features based on the following observations. In Yahoo! answers, users tend to write down correct answers at the beginning of the answer text, or at the end after providing the solving procedure. Surrounding words also give hints for finding correct solutions. For example, numbers that are close to the word *"answer"* are more likely to be in the gold answer. Given a post, numbers appearing in the answer text of more users are more likely to be the correct solution. Some words in the question sentence help determine answer dimension. For example, *"How far does Tom run?"* requires a one-dimension answer while *"How much do they each earn?"* indicates multiple dimensions. Main features are listed in table 1.

Table 1: Features for automatic answer extraction

Local context features
Relative position in the procedure
On right side of the symbol "="?
On left side of the symbol "="?
Close to *"ans", "answer", "result", or "therefore"*
Global features
Is in the text of the first answer (the first answer is often marked as the best answer in Yahoo! answers)?
Is in problem text?
Frequency in the text of all answers for this problem
Frequency in the first position of all answers
Frequency in the last position of all answers
Number value features
Is positive?
Is an integer?
Its value is between 0 to 1?
Equals to the predicted solution in automatic equation extraction?
Number set features
Are numbers at same line of answer text?
Are numbers at consecutive lines of answer text?
Frequency of the numbers at same line in all answers
Frequency of the numbers at consecutive lines in all answers
Dimension features
Has singular verb in question?
Has plural noun in question?
Has special words (e.g., *and, both, each, all*) in question?

Inference: After we train the model to get parameter vector ν, the predicated gold answer is selected from the candidate number subsets by maximizing $\nu \cdot f(x_i, y_{ik})$. Formally, the predicated gold

answer is,

$$\arg\max_{y_{ik} \in Y_i} \nu \cdot f(x_i, y_{ik})$$

About 3,000 problems are manually annotated with answers and equations by the human annotators we hire. Then we train and evaluate our model using 5-fold cross validation. The extractor's performance is shown in Figure 2. To preserve an accuracy rate of 90%, we use score = 3 as the threshold and only keep problems with predicted confidence score >= 3.

Please note that precision is more important than recall in our scenario. We need to guarantee that most extracted answers are correct. Lower recall can be tackled by processing more posts.

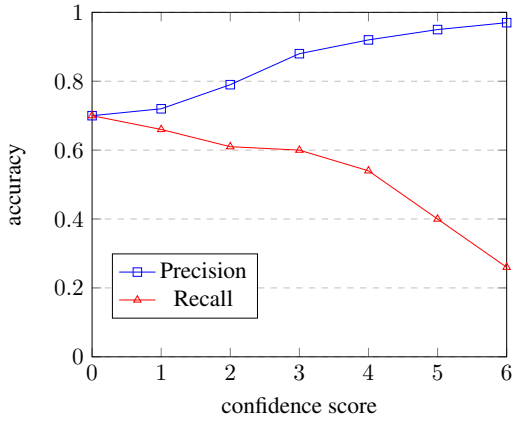


Figure 2: Accuracy of answer extraction

3.3 Automatic Equation Annotation

Now we illustrate how to extract equation systems automatically from the unstructured answer text of a post. The input is the answer text of n answers:

$$T = \{T_1, T_2, \ldots, T_n\}$$

For example in Figure 1, there are four answers, each corresponding to a piece of answer text T_i.

The task is not easy, because variables and equations may not be in standard formats in answer text. In addition, equations may be duplicate (like those in Answer 1 of Figure 1). Our algorithm is a two-phase procedure:

Candidate extraction: We extract an equation system from each piece of answer text T_i. In processing T_i, we first extract a list of equations by regular expression matching. Then the equations are added to the equation system by the order of their occurrences in the text. Before adding an equation, we check whether it can be induced by the already-added equations. If so, we skip it. Duplicate equations are effectively reduced in this way.

Voting by solution: We solve each equation system obtained from the first phase and build a (equation system, solution) bipartite graph. We then choose the equation system that has the maximum degree as our output. For example, if three equation systems return the solution $\{24\}$ and the fourth returns $\{-1\}$, we will choose one from the first three equation systems. To improve precision, we do not return any equation system if the maximal degree is less than 2.

We evaluate our equation extractor on 3,000 manually annotated problems [4]. For an equation system extracted for a problem, we say it is correct if the annotated gold answer is a subset of the solutions to the equation system. For example, if the gold answer is $\{16, 34\}$ and the solution to the equation system is $\{16, 34, 100\}$, then the equation system is considered correct. Evaluation results show a precision of **91.4%** and a recall of **64.7%**.

3.4 Datasets Summary

Below are a list of previous benchmark datasets for math word problem solving.

Alg514 is introduced in Kushman et al. (2014) and also used in Zhou et al. (2015) for evaluation. It consists of 514 algebra word problems from algebra.com[5], with each problem annotated with linear equations. The *template* (explained later) of each problem has to appear at least six times in the whole set.

Verb395 (Hosseini et al., 2014): A collection of addition/subtraction problems.

Dolphin1878: A collection built by Shi et al. (2015), containing 1,878 number word problems obtained from algebra.com and Yahoo! answers.

DRAW (Upadhyay and Chang, 2015): Containing 1,000 algebra word problems from algebra.com, each annotated with linear equations.

SingleEQ: By Koncel-Kedziorski et al. (2015), containing 508 problems, each of which corresponds to one single equation.

Before comparing the datasets, let's first introduce the concept of *equation system templates*, which are first introduced in Kushman et al. (2014)

[4] the same set of problems as we used in training and evaluating answer extraction

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

Table 2: Comparison of different datasets

Dataset	# Problems	# Templates	# Sentences	# Words	Problems types
Verb395	395	3	1.13k	12.4k	homogeneous addition or subtraction problems
Alg514	514	28	1.62k	19.3k	algebra, linear
Dolphin1878	1,878	1,183	3.30k	41.4k	number word problems
DRAW	1,000	232	2.67k	35.3k	algebra, linear
SingleEQ	508	31	1.38k	13.8k	single equation, linear
Dolphin18K	18,460	5,871	49.9k	604k	linear + nonlinear

for math word problem solving. A *template* is a unique form of equation system. For example, the following is a template of two equations:

$$n_1 \cdot x_1 + x_2 = n_2$$
$$x_1 + n_3 \cdot x_2 = n_4$$

The following equation system corresponds to the above template,

$$3 \cdot x_1 + x_2 = 5$$
$$x_1 + 7 \cdot x_2 = 15$$

Table 2 shows some statistical information of our dataset and previous ones. It can be seen that our dataset has a much larger scale (about 10 times the size of the Dolphin1878 collection and more than 17 times larger than the others) and higher diversity (in terms of both problem types and the number of templates contained).

We split our dataset into a development set and an evaluation set. The development set is used for algorithm design and debugging, while the evaluation set is for training and testing. Any problem in the evaluation set should be invisible to the people who design an automatic math problem solving system. Statistics on our dataset are shown in Table 3, where *dev* and *eval* represent the development set and the evaluation set respectively. Most problems are assigned with both equation systems and gold answers. Some of them are annotated with answers only, either because annotators feel it is hard to do so, or because our equation extraction algorithm returns empty results.

As most previous systems only handle linear equation systems, we summarize, in Table 4, the distribution of linear problems in the evaluation set by template size. In the table, the size of a template is defined as the number of problems corresponding to this template. Between the two numbers in each cell, the first one is the number of problems, and the second number (or the one in parentheses) is the number of templates in this category. For example, in the automatically annotated evaluation set, 166 templates have size 6 or larger. They correspond to 4,826 problems.

Table 3: Annotation statistics for our dataset

		Equations + answer	Answer only	Sum
dev	Manual	909	67	976
	Auto	2,245	507	2,752
	All	3,154	574	3,728
eval	Manual	3,605	321	3,926
	Auto	8,754	2,052	10,806
	All	12,359	2,373	14,732

4 Experiments

4.1 Systems for evaluation

We report the performance of several state-of-the-art systems on our new dataset.

KAZB: A template-based statistical learning method introduced in Kushman et al. (2014). It maps a problem to one equation template defined in the training set by reasoning across problem sentences. KAZB reports an accuracy of 68.7% on the Alg514 dataset.

ZDC: Proposed in Zhou et al. (2015) as an improved version of KAZB. It reduces the search space by not modeling alignment between noun phrases and variables. It achieves an accuracy of 79.7% on Alg514.

SIM is a simple similarity-based method implemented by us. To solve a problem, it calculates the lexical similarity between the problem and each problem in the training set. Then the equation system of the most similar problem is ap-

Table 4: Problem distribution by template size (for linear problems only)

Template size	Manual	Auto	All
(all linear templates)	2,675 (876)	7,969 (2,609)	10,644 (3,158)
>=2	2,036 (237)	5,956 (596)	8,229 (743)
>=5	1,678 (98)	4,979 (196)	7,081 (254)
>=6	1,578 (78)	4,826 (166)	6,827 (216)
>=10	1,337 (43)	4,329 (96)	6,216 (130)
>=20	1,039 (22)	3,673 (48)	5,392 (68)
>=50	634 7	2,684 18	4,281 30

plied to the new problem. In a little more details, a test problem P_T is solved in two steps: template selection, and template slot filling. In the first step, each problem is modeled as a vector of word TF-IDF scores. The similarity between two problems is calculated by the weighted Jaccard similarity between their corresponding vectors. We choose, from the training data, problem P_1 that has the maximal similarity with P_T and use the equation template T of P_1 as the template of problem P_T. In the second step, the numbers appearing in problem P_T are mapped to the number slots of template T (which has been identified in the first step). The mapping is implemented by selecting one problem P_2 from all the training problems corresponding to template T so that it has the minimum word-level edit-distance to P_T. Then the number mapping of P_2 is borrowed as the number mapping of P_T. For example, for the following test problem,

An overnight mail service charges $3.60 for the first six ounces and $0.45 for each additional ounce or fraction of an ounce. Find the number of ounces in a package that cost $7.65 to deliver.

Assuming that a problem P_1 has maximum Jaccard similarity with the above problem and its corresponding equation template is as follows, this template will be identified in the first step,

$$n_1 + n_2 * (x - n_3) = n_4$$

Assume that P_2 has the minimum edit-distance to P_T among all the training problems corresponding to template T. Suppose the numbers in P_2 are (by their order in the problem text),

$$3.5, 5, 0.5, 6.5$$

Also suppose P_2 is annotated with the following equation system,

$$3.5 + 0.5 * (x - 5) = 6.5$$

Then we will choose P_2 and borrow its number mapping. So the mapping from numbers in the above test problem to template slots will be,

$$3.60/n_1; 6/n_3; 0.45/n_2; 7.65/n_4$$

In implementing SIM, we do not use any POS tagging or syntactic parsing features for similarity calculation. This method gets an accuracy of 71.2% on Alg514 and 49.0% on SingleEQ.

Systems not included for evaluation: Although the system of Shi et al. (2015) achieves very high performance on number word problems, we do not include it in our evaluation because it is unknown how to extend it to other problem types. The system of Hosseini et al. (2014) is not included in our evaluation because it only handles homogeneous addition/subtraction problems. The systems of Koncel-Kedziorski et al. (2015) and Roy and Roth (2015) are also not included because so far they only supports problems with one single linear equation.

4.2 Overall Evaluation Results

Table 5 shows the accuracy of various systems on different subsets of our dataset. In the table, Manual.Linear contains all the manually annotated problems with linear equation systems. It contains 2,675 problems and 876 templates (as shown in Table 4). Auto.LinearT6 (containing 4,826 problems) is the set of all the automatically annotated problems with a template size larger than or equal to 6. Similarly, LinearT2 means the subset of problems with template size ≥ 2. For each system on each subset, experiments are conducted using 5-fold cross-validation with 80% problems randomly selected as training data and the remaining 20% for test.

In the table, "-" means that the system does not complete running on the dataset in three days. Since KAZB and ZDC only handle linear equation systems, they are not applicable to the datasets

	Systems		
Dataset	KAZB	ZDC	SIM
Manual.Linear	10.7%	11.1%	13.3%
Manual.LinearT2	12.8%	13.9%	17.3%
Manual.LinearT6	17.6%	17.1%	18.8%
Auto.Linear	-	17.2%	17.4%
Auto.LinearT2	-	20.1%	19.2%
Auto.LinearT6	-	19.2%	18.4%
All.Linear	-	17.9%	18.4%
All.LinearT2	-	20.6%	20.3%
All.LinearT6	-	21.7%	20.2%
All (Dolphin18K)	n/a	n/a	16.7%
Alg514	68.7%	79.7%	71.2%

Table 5: Overall evaluation results

containing nonlinear problems. An "n/a" is filled in the corresponding cell in this case.

The results show that all three systems (KAZB, ZDC, and SIM) have extremely low performance on our new datasets. Surprisingly, no system achieves an accuracy rate of over 25%. Such results indicate that automatic math word problem solving is still a very challenging task.

Another surprising observation is that KAZB and ZDC do not perform better than SIM, a simple similarity-based method which runs much faster than the two statistical learning systems.

By comparing the results obtained from the manual version of the datasets with their corresponding auto version (for example, Manuall.Linear vs. Auto.Linear), we can see larger accuracy scores on the auto versions [6]. This demonstrates the usefulness of the automatically annotated data. Considering the huge cost of manually assigning equation systems and gold answers, automatic annotation has good potential in constructing larger datasets.

4.3 Why Different from Previous Results

The last line of Table 5 displays the results on Alg514. All three systems perform well on Alg514 but poorly on Dolphin18K. To study the reason of such a large gap, we derive two small datasets from All.Linear by referring to the equation templates in Alg514.

Small.01: The set of all problems in All.Linear that correspond to one of the 28 templates in Alg514. The dataset contains 2,021 problems.

[6]Please note that the auto versions are more than 2 times larger.

Small.02: A subset of Small.01, constructed by randomly removing problems from Small.01 so that each template contains similar number of problems as in Alg514. In other words, Small.02 and Alg514 have similar problem distribution among templates.

	Small.01	Small.02
KAZB	29.9%	50.0%
ZDC	30.1%	52.7%
SIM	33.7%	43.0%

Table 6: The case of fewer number of templates

We still use 5-fold cross validation to test and compare system performance on the two small datasets. Evaluation results are displayed in Table 6. We now obtain higher accuracy scores for each system, but there is a big difference between the results on Small.01 and Small.02. As mentioned in (Upadhyay and Chang, 2015), Alg514 has a skewed problem distribution, with a few templates covering almost 50% problems. This may be the main reason why all three systems achieve high accuracy on this dataset and on Small.02. From all of the above results, we see at least two factors which affect system performance: number of templates in the dataset, and the distribution of problems among the templates. For a small dataset, the distribution of problems among templates have a huge impact on evaluation results.

4.4 Effect of Training Data Size

Now we investigate the performance change of various systems when the size of training data changes. The goal is to check whether the accuracy increases quickly when more training data are added. This is important: If it is the case, we can tackle this task by simply adding more training data, either manually or automatically. Otherwise, we have to discover new approaches.

We conduct experiments in two settings: fixed-test-set, and increasing-test-set. In the first setting, we randomly choose 1/2 of the problems from the Manual.Linear subset to form a fixed test-set (with size 1330). Then the other problems in All.Linear forms a candidate training collection (containing 9314 elements). We construct training sets of different scales by doing random sampling from the candidate training collection.

In the second setting (i.e., increasing-test-set), we construct datasets (training set plus test set) of

Training data source	All.LinearT6					All.Linear				
Training data size	138	434	1024	2940	5771	500	1000	2000	5000	9000
Test set size	1330	1330	1330	1330	1330	1330	1330	1330	1330	1330
KAZB accuracy (%)	6.7	7.2	-	-	-	7.1	8.3	-	-	-
ZDC accuracy (%)	6.1	7.5	8.6	11.4	12.6	5.5	9.2	10.5	12.5	13.1
SIM accuracy (%)	5.5	8.7	11.0	13.7	15.9	6.5	10.8	12.2	14.9	18.4

Table 7: System performance with different training data size (setting: fixed-test-set)

Training data size	400	800	1600	4000	8516
Test set size	100	200	400	1000	2128
KAZB	5.4%	6.7%	11.7%	-	-
ZDC	5.8%	7.6%	12.9%	17.0%	17.9%
SIM	7.4%	10.0%	13.3%	16.9%	18.4%

Table 8: System performance with different training data size (setting: increasing-test-set)

different scales by doing random sampling from All.Linear, and then conduct 5-fold cross validation on each dataset. In each fold, 80% problems are chosen at random for training, and the other 20% for testing.

The results in the two settings are reported in Tables 7 and 8 respectively. Both tables show that the accuracy of all the three systems improves steadily but slowly along with the increasing of training data size. So it is not very effective to improve accuracy by simply adding more training data.

4.5 Results Summary

In summary, the following observations are made from the experiments on our new dataset. First, all systems evaluated on the Dolphin18K dataset perform much worse than on the small and less diverse datasets. Second, the two statistical learning methods do not perform better than a simple similarity-based method. Third, it seems not promising for the current methods to achieve much better results by simply adding more training data. Automatic math word problem solving is still a very challenging task so far.

5 Conclusion

We have constructed Dolphin18K, a large dataset for training and evaluating automatic math word problem solving systems. The new dataset is almost one order of magnitude larger than most of previous ones, and has a much higher level of diversity in term of problem types. We reduce human annotation cost by automatically extracting gold answers and equation systems from the unstructured answer text of CQA posts.

We have also conducted experiments on our dataset to evaluate state-of-the-art systems. Interesting and surprising observations are made from the experimental results.

Acknowledgments

We would like to thank the annotators for their efforts in annotating the math problems in our dataset. Thanks to the anonymous reviewers for their helpful comments and suggestions.

References

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding high-quality content in social media. In *First ACM International Conference on Web Search and Data Mining (WSDM'08)*.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. http://arxiv.org/abs/math/0701393.

Daniel G. Bobrow. 1964a. Natural language input for a computer problem solving system. Technical report, Cambridge, MA, USA.

Daniel G. Bobrow. 1964b. Natural language input for a computer problem solving system. Ph.D. Thesis.

Diane J. Briars and Jill H. Larkin. 1984. An integrated model of skill in solving elementary word problems. *Cognition and Instruction*, 1(3):245–296.

Eugene Charniak. 1968. Carps, a program which solves calculus word problems. Technical report.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 303–316.

Gao Cong, Long Wang, Chin-Yew Lin, Young-In Song, and Yueheng Sun. 2008. Finding question-answer pairs from online forums. In *Proceedings of 31st International ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08)*, pages 467–474.

Denise Dellarosa. 1986. A computer simulation of children's arithmetic word-problem solving. *Behavior Research Methods, Instruments, & Computers*, 18(2):147–154.

Shilin Ding, Gao Cong, Chin-Yew Lin, and Xiaoyan Zhu. 2008. Using conditional random fields to extract context and answers of questions from online forums. In *Proceedings of the 46th Annual Meeting of the ACL: HLT (ACL 2008)*, pages 710–718, Columbus, USA.

Charles R. Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer, 2000. *Large Margin Rank Boundaries for Ordinal Regression*, chapter 7, pages 115–132.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, October.

Rik Koncel-Kedziorsk, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In *Natural Language Processing and Information Systems. International Conference on Applications of Natural Language to Information Systems (NLDB-2012)*, pages 247–252.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752. The Association for Computational Linguistics.

Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm. 2015. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Shyam Upadhyay and Ming-Wei Chang. 2015. Draw: A challenging and diverse algebra word problem set. Number MSR-TR-2015-78, October.

Ma Yuhui, Zhou Ying, Cui Guangzuo, Ren Yun, and Huang Ronghuai. 2010. Frame-based calculus of solving arithmetic multistep addition and subtraction word problems. *Education Technology and Computer Science, International Workshop*, 2:476–479.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Embeddings for Word Sense Disambiguation: An Evaluation Study

Ignacio Iacobacci[1], **Mohammad Taher Pilehvar**[2] **and Roberto Navigli**[1]
[1]Department of Computer Science, Sapienza University of Rome, Italy
[2]Language Technology Lab, University of Cambridge, UK
{iacobacci,navigli}@di.uniroma1.it
mp792@cam.ac.uk

Abstract

Recent years have seen a dramatic growth in the popularity of word embeddings mainly owing to their ability to capture semantic information from massive amounts of textual content. As a result, many tasks in Natural Language Processing have tried to take advantage of the potential of these distributional models. In this work, we study how word embeddings can be used in Word Sense Disambiguation, one of the oldest tasks in Natural Language Processing and Artificial Intelligence. We propose different methods through which word embeddings can be leveraged in a state-of-the-art supervised WSD system architecture, and perform a deep analysis of how different parameters affect performance. We show how a WSD system that makes use of word embeddings alone, if designed properly, can provide significant performance improvement over a state-of-the-art WSD system that incorporates several standard WSD features.

1 Introduction

Embeddings represent words, or concepts in a low-dimensional continuous space. These vectors capture useful syntactic and semantic information, such as regularities in language, where relationships are characterized by a relation-specific vector offset. The ability of embeddings to capture knowledge has been exploited in several tasks, such as Machine Translation (Mikolov et al., 2013, MT), Sentiment Analysis (Socher et al., 2013), Word Sense Disambiguation (Chen et al., 2014, WSD) and Language Understanding (Mesnil et al., 2013). Supervised WSD is based on the hypothesis that contextual information provides a good approximation to word meaning, as suggested by Miller and Charles (1991): semantically similar words tend to have similar contextual distributions.

Recently, there have been efforts on leveraging embeddings for improving supervised WSD systems. Taghipour and Ng (2015) showed that the performance of conventional supervised WSD systems can be increased by taking advantage of embeddings as new features. In the same direction, Rothe and Schütze (2015) trained embeddings by mixing words, lexemes and synsets, and introducing a set of features based on calculations on the resulting representations. However, none of these techniques takes full advantage of the semantic information contained in embeddings. As a result, they generally fail in providing substantial improvements in WSD performance.

In this paper, we provide for the first time a study of different techniques for taking advantage of the combination of embeddings with standard WSD features. We also propose an effective approach for leveraging embeddings in WSD, and show that this can provide significant improvement on multiple standard benchmarks.

2 Word Embeddings

An embedding is a representation of a topological object, such as a manifold, graph, or field, in a certain space in such a way that its connectivity or algebraic properties are preserved (Insall et al., 2015). Presented originally by Bengio et al. (2003), word embeddings aim at representing, i.e., embedding, the ideal semantic space of words in a real-valued continuous vector space. In contrast to traditional distributional techniques, such as Latent Semantic Analysis (Landauer and Dutnais, 1997, LSA) and Latent Dirichlet Allocation (Blei et al., 2003, LDA), Bengio et al. (2003) designed a

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 897–907,
Berlin, Germany, August 7-12, 2016.

feed-forward neural network capable of predicting a word given the words preceding (i.e., leading up to) that word. Collobert and Weston (2008) presented a much deeper model consisting of several layers for feature extraction, with the objective of building a general architecture for NLP tasks. A major breakthrough occurred when Mikolov et al. (2013) put forward an efficient algorithm for training embeddings, known as Word2vec. A similar model to Word2vec was presented by Pennington et al. (2014, GloVe), but instead of using latent features for representing words, it makes an explicit representation produced from statistical calculation on word countings.

Numerous efforts have been made to improve different aspects of word embeddings. One way to enhance embeddings is to represent more fine-grained semantic items, such as word senses or concepts, given that conventional embeddings conflate different meanings of a word into a single representation. Several research studies have investigated the representation of word senses, instead of words (Reisinger and Mooney, 2010; Huang et al., 2012; Camacho-Collados et al., 2015b; Iacobacci et al., 2015; Rothe and Schütze, 2015). Another path of research is aimed at refining word embeddings on the basis of additional information from other knowledge resources (Faruqui et al., 2015; Yu and Dredze, 2014). A good example of this latter approach is that proposed by Faruqui et al. (2015), which improves pre-trained word embeddings by exploiting the semantic knowledge from resources such as PPDB[1] (Ganitkevitch et al., 2013), WordNet (Miller, 1995) and FrameNet (Baker et al., 1998). In the following section we discuss how embeddings can be integrated into an important lexical semantic task, i.e., Word Sense Disambiguation.

3 Word Sense Disambiguation

Natural language is inherently ambiguous. Most commonly-used words have several meanings. In order to identify the intended meaning of a word one has to analyze the context in which it appears by directly exploiting information from raw texts. The task of automatically assigning predefined meanings to words in contexts, known as Word Sense Disambiguation, is a fundamental task in computational lexical semantics (Navigli, 2009). There are four conventional approaches to WSD which we briefly explain in the following.

3.1 Supervised methods

These methods make use of manually sense-annotated data, which are curated by human experts. They are based on the assumption that a word's context can provide enough evidence for its disambiguation. Since manual sense annotation is a difficult and time-consuming process, something known as the "knowledge acquisition bottleneck" (Pilehvar and Navigli, 2014), supervised methods are not scalable and they require repetition of a comparable effort for each new language. Currently, the best performing WSD systems are those based on supervised learning. It Makes Sense (Zhong and Ng, 2010, IMS) and the system of Shen et al. (2013) are good representatives for this category of systems. We provide more information on IMS in Section 4.1.

3.2 Unsupervised methods

These methods create their own annotated corpus. The underlying assumption is that similar senses occur in similar contexts, therefore it is possible to group word usages according to their shared meaning and induce senses. These methods lead to the difficulty of mapping their induced senses into a sense inventory and they still require manual intervention in order to perform such mapping. Examples of this approach were studied by Agirre et al. (2006), Brody and Lapata (2009), Manandhar et al. (2010), Van de Cruys and Apidianaki (2011) and Di Marco and Navigli (2013).

3.3 Semi-supervised methods

Other methods, called semi-supervised, take a middle-ground approach. Here, a small manually-annotated corpus is usually used as a seed for bootstrapping a larger annotated corpus. Examples of these approaches were presented by Mihalcea and Faruque (2004). A second option is to use a word-aligned bilingual corpus approach, based on the assumption that an ambiguous word in one language could be unambiguous in the context of a second language, hence helping to annotate the sense in the first language (Ng and Lee, 1996).

3.4 Knowledge-based methods

These methods are based on existing lexical resources, such as knowledge bases, semantic networks, dictionaries and thesauri. Their main feature is their coverage, since they function indepen-

[1] www.paraphrase.org/#/download

dently of annotated data and can exploit the graph structure of semantic networks to identify the most suitable meanings. These methods are able to obtain wide coverage and good performance using structured knowledge, rivaling supervised methods (Patwardhan and Pedersen, 2006; Mohammad and Hirst, 2006; Agirre et al., 2010; Guo and Diab, 2010; Ponzetto and Navigli, 2010; Miller et al., 2012; Agirre et al., 2014; Moro et al., 2014; Chen et al., 2014; Camacho-Collados et al., 2015a).

3.5 Standard WSD features

As was analyzed by Lee and Ng (2002), conventional WSD systems usually make use of a fixed set of features to model the context of a word. The first feature is based on the words in the surroundings of the target word. The feature usually represents the local context as a binary array, where each position represents the occurrence of a particular word. Part-of-speech (POS) tags of the neighboring words have also been used extensively as a WSD feature. Local collocations represent another standard feature that captures the ordered sequences of words which tend to appear around the target word (Firth, 1957). Though not very popular, syntactic relations have also been studied as a possible feature (Stetina et al., 1998) in WSD.

More sophisticated features have also been studied. Examples are distributional semantic models, such as Latent Semantic Analysis (Van de Cruys and Apidianaki, 2011) and Latent Dirichlet Allocation (Cai et al., 2007). Inasmuch as they are the dominant distributional semantic model, word embeddings have also been applied as features to WSD systems. In this paper we study different methods through which word embeddings can be used as WSD features.

3.6 Word Embeddings as WSD features

Word embeddings have become a prominent technique in distributional semantics. These methods leverage neural networks in order to model the contexts in which a word is expected to appear. Thanks to their ability in efficiently learning the semantics of words, word embeddings have been applied to a wide range of NLP applications. Several studies have also investigated their integration into the Word Sense Disambiguation setting. These include the works of Zhong and Ng (2010), Taghipour and Ng (2015), Rothe and Schütze (2015), and Chen et al. (2014), which leverage embeddings for supervised (the former three) and knowledge-based (the latter) WSD. However, to our knowledge, no previous work has investigated methods for integrating word embeddings in WSD and the role that different training parameters can play. In this paper, we put forward a framework for a comprehensive evaluation of different methods of leveraging word embeddings as WSD features in a supervised WSD system. We provide an analysis of the impact of different parameters in the training of embeddings on the WSD performance. We consider four different strategies for integrating a pre-trained word embedding in a supervised WSD system, discussed in what follows.

3.6.1 Concatenation

Concatenation is our first strategy, which is inspired by the model of Bengio et al. (2003). This method consists of concatenating the vectors of the words surrounding a target word into a larger vector that has a size equal to the aggregated dimensions of all the individual embeddings. Let w_{ij} be the weight associated with the i^{th} dimension of the vector of the j^{th} word in the sentence, let D be the dimensionality of this vector, and W be the window size which is defined as the number of words on a single side. We are interested in representing the context of the I^{th} word in the sentence. The i^{th} dimension of the concatenation feature vector, which has a size of $2WD$, is computed as follows:

$$e_i = \begin{cases} w_{i \bmod D,\, I-W+\lfloor \frac{i}{D} \rfloor} & \text{if } \lfloor \frac{i}{D} \rfloor < W \\ w_{i \bmod D,\, I-W+1+\lfloor \frac{i}{D} \rfloor} & \text{otherwise} \end{cases}$$

where *mod* is the modulo operation, i.e., the remainder after division.

3.6.2 Average

As its name indicates, the average strategy computes the centroid of the embeddings of all the surrounding words. The formula divides each dimension by $2W$ since the number of context words is twice the window size:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} \frac{w_{ij}}{2W}$$

3.6.3 Fractional decay

Our third strategy for constructing a feature vector on the basis of the context word embeddings is inspired by the way Word2vec combines the words in the context. Here, the importance of a word

for our representation is assumed to be inversely proportional to its distance from the target word. Hence, surrounding words are weighted based on their distance from the target word:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} w_{ij} \frac{W - |I - j|}{W}$$

3.6.4 Exponential decay

Exponential decay functions similarly to the fractional decay, which gives more importance to the close context, but in this case the weighting in the former is performed exponentially:

$$e_i = \sum_{\substack{j=I-W \\ j \neq I}}^{I+W} w_{ij} (1 - \alpha)^{|I-j|-1}$$

where $\alpha = 1 - 0.1^{(W-1)^{-1}}$ is the decay parameter. We choose the parameter in such a way that the immediate surrounding words contribute 10 times more than the last words on both sides of the window.

4 Framework

Our goal was to experiment with a state-of-the-art conventional supervised WSD system and a varied set of word embedding techniques. In this section we discuss the WSD system as well as the word embeddings used in our experiments.

4.1 WSD System

We selected It Makes Sense (Zhong and Ng, 2010, IMS) as our underlying framework for supervised WSD. IMS provides an extensible and flexible platform for supervised WSD by allowing the verification of different WSD features and classification techniques. By default, IMS makes use of three sets of features: (1) POS tags of the surrounding words, with a window of three words on each side, restricted by the sentence boundary, (2) the set of words that appear in the context of the target word after stopword removal, and (3) local collocations which consist of 11 features around the target word. IMS uses a linear support vector machine (SVM) as its classifier.

4.2 Embedding Features

We take the real-valued word embeddings as new features of IMS and introduce them into the system without performing any further modifications. We carried out experiments with three different embeddings:

- **Word2vec** (Mikolov et al., 2013): We used the Word2vec toolkit[2] to learn 400 dimensional vectors on the September-2014 dump of the English Wikipedia which comprises around three billion tokens. We chose the Skip-gram architecture with the negative sampling set to 10. The sub-sampling of frequent words was set to 10^{-3} and the window size to 10 words.

- **C&W** (Collobert and Weston, 2008): These 50 dimensional embeddings were learnt using a neural network model, consisting of several layers for feature extraction. The vectors were trained on a subset of the English Wikipedia.[3]

- **Retrofitting**: Finally, we used the approach of Faruqui et al. (2015) to retrofit our Word2vec vectors. We used the Paraphrase Database (Ganitkevitch et al., 2013, PPDB) as external knowledge base for retrofitting and set the number of iterations to 10.

5 Experiments

We evaluated the performance of our embedding-based WSD system on two standard WSD tasks: lexical sample and all-words. In all the experiments in this section we used the exponential decay strategy (cf. Section 3.6) and a window size of ten words on each side of the target word.

5.1 Lexical Sample WSD Experiment

The lexical sample WSD tasks provide training datasets in which different occurrences of a small set of words are sense annotated. The goal is for a WSD system to analyze the contexts of the individual senses of these words and to capture clues that can be used for distinguishing different senses of a word from each other at the test phase.

Datasets. As our benchmark for the lexical sample WSD, we chose the Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 (Mihalcea et al., 2004), and SemEval-2007 (Pradhan et al., 2007) English Lexical Sample WSD tasks. The former two cover nouns, verbs and adjectives in their datasets whereas the latter task focuses on nouns and verbs

[2] code.google.com/archive/p/word2vec/
[3] http://ronan.collobert.com/senna/

Task	Training			Test		
	noun	verb	adjective	noun	verb	adjective
Senseval-2 (SE2)	4851	3566	755	1740	1806	375
Senseval-3 (SE3)	3593	3953	314	1807	1978	159
SemEval-07 (SE7)	13287	8987	–	2559	2292	–

Table 1: The number of sentences per part of speech in the datasets of the English lexical sample tasks we considered for our experiments.

System	SE2	SE3	SE7
IMS (2010)	65.3	72.9	87.9
Taghipour and Ng (2015)	66.2	73.4	–
AutoExtend (2015)	66.5	73.6	–
IMS + C&W	64.3	70.1	88.0
IMS + Word2vec	**69.9**	**75.2**	**89.4**
IMS + Retrofitting	65.9	72.8	88.3
C&W feature only	55.0	61.6	83.4
Word2vec feature only	65.6	69.4	87.0
Retrofitting feature only	67.2	72.7	88.0

Table 2: F1 performance on the three English lexical sample datasets. IMS + X denotes the improved IMS system when the X set of word representations were used as additional features. We also show in the last three rows the results for the IMS system when word representations were used as the only features.

only. Table 1 shows the number of sentences per part of speech for the training and test datasets of each of these tasks.

Comparison systems. In addition to the vanilla IMS system in its default setting we compared our system against two recent approaches that also modify the IMS system so that it can benefit from the additional knowledge derived from word embeddings for improved WSD performance: (1) the system of Taghipour and Ng (2015), which combines word embeddings of Collobert and Weston (2008) using the concatenation strategy (cf. Section 3.6) and introduces the combined embeddings as a new feature in addition to the standard WSD features in IMS; and (2) AutoExtend (Rothe and Schütze, 2015), which constructs a whole new set of features based on vectors made from words, senses and synsets of WordNet and incorporates them in IMS.

5.1.1 Lexical sample WSD results

Table 2 shows the F1 performance of the different systems on the three lexical sample datasets. As can be seen, the IMS + Word2vec system improves over all comparison systems including those that combine standard WSD and embedding features (i.e., the system of Taghipour and Ng (2015) and AutoExtend) across all the datasets. This shows that our proposed strategy for introducing word embeddings into the IMS system on the basis of exponential decay was beneficial. In the last three rows of the table, we also report the performance of the WSD systems that leverage only word embeddings as their features and do not incorporate any standard WSD feature. It can be seen that word embeddings, in isolation, provide competitive performance, which proves their capability in obtaining the information captured by standard WSD features. Among different embeddings, the retrofitted vectors provide the best performance when used in isolation.

5.2 All-Words WSD Experiments

The goal in this task is to disambiguate all the content words in a given text. In order to learn models for disambiguating a large set of content words, a high-coverage sense-annotated corpus is required. Since all-words tasks do not usually provide any training data, the challenge here is not only to learn accurate disambiguation models from the training data, as is the case in the lexical sample task, but also to gather high-coverage training data and to learn disambiguation models for as many words as possible.

Training corpus. As our training corpus we opted for two available resources: SemCor and OMSTI. SemCor (Miller et al., 1994) is a manually sense-tagged corpus created by the WordNet project team at Princeton University. The dataset is a subset of the English Brown Corpus and comprises around 360,000 words, providing annotations for more than 200K content words.[4] OM-

[4]We used automatic mappings to WordNet 3.0 provided in `web.eecs.umich.edu/~mihalcea/downloads.html`.

STI[5] (One Million Sense-Tagged for Word Sense Disambiguation and Induction) was constructed based on the DSO corpus (Ng and Lee, 1996) and provides annotations for around 42K different nouns, verbs, adjectives, and adverbs.

Datasets. As benchmark for this experiment, we considered the Senseval-2 (Edmonds and Cotton, 2001), Senseval-3 (Snyder and Palmer, 2004), and SemEval-2007 (Pradhan et al., 2007) English all-words tasks. There are 2474, 2041, and 465 words for which at least one of the occurrences has been sense annotated in the Senseval-2, Senseval-3 and SemEval-2007 datasets, respectively.

Experimental setup. Similarly to the lexical sample experiment, in the all-words setting we used the exponential decay strategy (cf. Section. 4.2) in order to incorporate word embeddings as new features in IMS. For this experiment, we only report the results for the best-performing word embeddings in the lexical sample experiment, i.e., Word2vec (see Table 2).

Comparison systems. We benchmarked the performance of our system against five other systems. Similarly to our lexical sample experiment, we compared against the vanilla IMS system and the work of Taghipour and Ng (2015). In addition, we performed experiments on the nouns subsets of the datasets in order to be able to provide comparisons against two other WSD approaches: Babelfy (Moro et al., 2014) and Muffin (Camacho-Collados et al., 2015a). Babelfy is a multilingual knowledge-based WSD and Entity Linking algorithm based on the semantic network of BabelNet. Muffin is a multilingual sense representation technique that combines the structural knowledge derived from semantic networks with the distributional statistics obtained from text corpora. The system uses sense-based representations for performing WSD. Camacho-Collados et al. (2015a) also proposed a hybrid system that averages the disambiguation scores of IMS with theirs (shown as "Muffin + IMS" in our tables). We also report the results for UKB w2w (Agirre and Soroa, 2009), another knowledge-based WSD approach based on Personalized PageRank (Haveliwala, 2002). Finally, we also carried out experiments with the pre-trained models[6] that are provided with the IMS toolkit, as well as IMS trained on our two training corpora, i.e., SemCor and OMSTI.

[5] www.comp.nus.edu.sg/~nlp/corpora.html

[6] www.comp.nus.edu.sg/~nlp/sw/models.tar.gz

System	SE2	SE3	SE7
MFS baseline	60.1	62.3	51.4
IMS (Zhong and Ng, 2010)	68.2	67.6	58.3
Taghipour and Ng (2015)	–	**68.2**	–
IMS (pre-trained models)	67.7	67.5	58.0
IMS (SemCor)	62.5	65.0	56.5
IMS (OMSTI)	67.0	66.4	57.6
IMS + Word2vec (SemCor)	63.4	65.3	57.8
IMS + Word2vec (OMSTI)	**68.3**	**68.2**	**59.1**

Table 3: F1 performance on different English all-words WSD datasets.

System	SE2	SE3	SE7
MFS baseline	71.6	70.3	65.8
Babelfy	–	68.3	62.7
Muffin	–	–	66.0
Muffin + IMS	–	–	68.5
UBK w2w	–	65.3	56.0
IMS (pre-trained models)	77.5	74.0	66.5
IMS (SemCor)	73.0	70.8	64.2
IMS (OMSTI)	76.6	73.3	67.7
IMS + Word2vec (SemCor)	74.2	70.1	68.6
IMS + Word2vec (OMSTI)	**77.7**	**74.1**	**71.5**

Table 4: F1 performance in the nouns subsets of different all-words WSD datasets.

5.2.1 All-words WSD results

Tables 3 and 4 list the performance of different systems on, respectively, the whole and the noun-subset datasets of the three all-words WSD tasks. Similarly to our lexical sample experiment, the IMS + Word2vec system provided the best performance across datasets and benchmarks. The coupling of Word2vec embeddings to the IMS system proved to be consistently helpful. Among the two training corpora, as expected, OMSTI provided a better performance owing to its considerably larger size and higher coverage. Another point to be noted here is the difference between results of the IMS with the pre-trained models and those trained on the OMSTI corpus. Since we used the same system configuration across the two runs, we conclude that the OMSTI corpus is either substantially smaller or less representative than the corpus used by Zhong and Ng (2010) for building

the pre-trained models of IMS. Despite this fact, the IMS + Word2vec system can consistently improve the performance of IMS (pre-trained models) across the three datasets. This shows that a proper introduction of word embeddings into a supervised WSD system can compensate the negative effect of using lower quality training data.

6 Analysis

We carried out a series of experiments in order to check the impact of different system parameters on the final WSD performance. We were particularly interested in observing the role that various training parameters of embeddings as well as WSD features have in the WSD performance. We used the Senseval-2 English Lexical Sample task as our benchmark for this analysis.

6.1 The effect of different parameters

Table 5 shows F1 performance of different configurations of our system on the task's dataset. We studied five different parameters: the type (i.e., w2v or Retrofitting) and dimensionality (200, 400, or 800) of the embeddings, combination strategy (concatenation, average, fractional or exponential decay), window size (5, 10, 20 and words), and WSD features (collocations, POS tags, surrounding words, all of these or none). All the embeddings in this experiment were trained on the same training data and, unless specified, with the same configuration as described in Section 4.2. As baseline we show in the table the performance of the vanilla WSD system, i.e., IMS. For better readability, we report the differences between the performances of our system and the baseline.

We observe that the addition of Word2vec word embeddings to IMS (+w2v in the table) was beneficial in all settings. Among combination strategies, concatenation and average produced the smallest gain and did not benefit from embeddings of higher dimensionality. However, the other two strategies, i.e., fractional and exponential decay, showed improved performance with the increase in the size of the employed embeddings, irrespective of the WSD features. The window size showed a peak of performance when 10 words were taken in the case of standard word embeddings. For retrofitting, a larger window seems to have been beneficial, except when no standard WSD features were taken. Another point to note here is that, among the three WSD features, POS proved to be the most effective one while due to the nature of the embeddings, the exclusion of the Surroundings features in addition to the inclusion of the embeddings was largely beneficial in all the configurations. Furthermore, we found that the best configurations for this task were the ones that excluded Surroundings, and included w2v embeddings with a window of 10 and 800 dimensions with exponential decay strategy (70.2% of F1 performance) as well as the configuration used in our experiments, with all the standard features, and w2v embeddings with 400 dimensions, a window of 10 and exponential decay strategy (69.9% of F1 performance).

The retrofitted embeddings provided lower performance improvement when added on top of standard WSD features. However, when they were used in isolation (shown in the right-most column), the retrofitted embeddings interestingly provided the best performance, improving the vanilla WSD system with standard features by 2.8 percentage points (window size 5, dimensionality 800). In fact, the standard features had a destructive role in this setting as the overall performance was reduced when they were combined with the retrofitted embeddings. Finally, we point out the missing values in the configuration with 800 dimensions and a window size of 20. Due to the nature of the concatenation strategy, this configuration greatly increased the number of features from embeddings only, reaching 32000 (800 x 2 x 20) features. Not only was the concatenation strategy unable to take advantage of the increased dimensionality, but also it was not able to scale.

These results show that a state-of-the-art supervised WSD system can be constructed without incorporating any of the conventional WSD features, which in turn demonstrates the potential of retrofitted word embeddings for WSD. This finding is interesting, because it provides the basis for further studies on how synonymy-based semantic knowledge introduced by retrofitting might play a role in effective WSD, and how retrofitting might be optimized for improved WSD. Indeed, such studies may provide the basis for re-designing the standard WSD features.

6.2 Comparison of embedding types

We were also interested in comparing different types of embeddings in our WSD framework. We tested for seven sets of embeddings with dif-

Collocations			✓					✓			✓					
POS						✓			✓			✓				
Surroundings			✓			✓					✓					
Dimensionality		200	400	800	200	400	800	200	400	800	200	400	800	200	400	800
System / Strategy / Window																
IMS			62.4			63.7			62.0			65.2			–	
+ w2v	Con	5	+0.1	+0.4	+0.1	-0.1	+0.3	+0.2	+0.1	+0.5	+0.1	-0.2	+0.1	+0.1	46.9	48.7	44.2
		10	-0.1	+0.5	+0.3	-0.1	+0.5	0.0	+0.6	+1.0	+0.5	-0.1	+0.1	-0.1	48.6	51.1	49.7
		20	-0.2	+0.4	—	-0.3	+0.3	—	+0.7	+1.5	—	-0.5	+0.4	—	52.5	54.1	—
+ w2v	Avg	5	+0.8	+1.0	+1.0	+1.3	+1.3	+1.4	+3.9	+4.2	+4.1	+1.7	+1.4	+1.6	58.3	59.9	61.3
		10	+0.8	+0.9	+0.9	+0.6	+0.7	+0.8	+3.6	+3.7	+3.9	+0.6	+0.6	+0.7	63.7	64.1	64.7
		20	+0.3	+0.3	+0.3	+0.5	+0.3	+0.4	+2.4	+2.3	+2.3	+0.2	+0.2	+0.2	62.7	63.1	63.5
+ w2v	Frac	5	+3.9	+4.9	+5.2	+4.2	+4.6	+5.3	+6.3	+6.6	+6.8	+3.0	+3.6	+3.8	61.2	63.1	64.8
		10	+4.9	+5.8	+5.7	+4.6	+5.2	+5.1	+5.9	+7.0	+7.4	+3.6	+4.3	+4.0	61.3	63.8	65.2
		20	+4.4	+4.5	+4.7	+3.7	+4.0	+4.3	+4.8	+6.1	+5.4	+3.2	+3.3	+3.4	61.2	63.4	63.9
+ w2v	Exp	5	+4.1	+5.0	+5.2	+4.1	+4.7	+5.0	+6.1	+6.1	+6.4	+2.9	+3.5	+3.7	62.3	64.7	64.9
		10	+5.4	**+6.6**	+6.4	+4.9	+5.8	**+6.0**	+7.2	+7.7	**+8.2**	+4.1	**+4.7**	+4.6	63.2	65.6	66.9
		20	+5.2	+5.6	+5.9	+4.4	+5.1	+4.9	+6.1	+7.0	+6.8	+3.9	+4.3	+4.2	61.9	64.4	65.2
+ Ret	Con	5	-0.1	-0.1	-0.1	-0.1	-0.1	0.0	+0.1	+0.1	-0.1	-0.1	+0.1	+0.1	50.7	53.5	50.9
		10	+0.1	0.0	0.0	-0.3	0.0	0.0	+0.1	+0.2	+0.1	0.0	0.0	0.0	52.1	54.2	53.4
		20	0.0	0.0	—	-0.2	0.0	—	+0.7	+0.3	—	0.0	-0.1	—	53.7	54.8	—
+ Ret	Avg	5	+0.1	0.0	-0.1	+0.1	0.0	-0.1	+0.8	+0.8	+0.7	+0.1	0.0	+0.1	60.7	60.3	60.5
		10	-0.2	-0.1	0.0	-0.2	-0.3	0.0	+0.7	+0.7	+0.5	0.0	+0.1	+0.1	58.9	58.4	58.2
		20	-0.1	+0.1	+0.1	-0.2	-0.2	-0.2	+0.5	+0.4	+0.4	0.0	0.0	0.0	56.5	56.0	55.5
+ Ret	Frac	5	+1.4	+1.3	+1.2	+1.2	+1.0	+0.9	+3.3	+3.1	+2.9	+0.5	+0.3	+0.3	66.5	67.3	67.7
		10	+1.7	+1.4	+1.2	+1.5	+1.4	+1.2	+5.2	+4.7	+4.5	+0.7	+0.8	+0.6	64.4	66.2	66.1
		20	+2.2	+2.2	+1.8	+2.2	+1.8	+2.0	+6.7	+6.4	+5.9	+1.3	+1.2	+1.0	64.0	64.2	64.7
+ Ret	Exp	5	+1.1	+1.1	+1.1	+0.8	+0.8	+0.7	+2.7	+2.6	+2.2	+0.3	+0.3	+0.3	66.8	67.7	**68.0**
		10	+1.5	+1.3	+1.0	+1.2	+1.1	+1.0	+4.4	+4.2	+3.8	+0.7	+0.7	+0.3	65.9	67.2	67.5
		20	+1.8	+1.7	+1.5	+1.7	+1.5	+1.5	+6.3	+5.9	+5.4	+1.1	+0.8	+0.7	65.1	65.8	66.5

Table 5: F1 performance of different models on the Senseval-2 English Lexical Sample task. We show results for varied dimensionality (200, 400, and 800), window size (5, 10 and 20 words) and combination strategy, i.e., Concatenation (Con), Averaging (Avg), Fractional decay (Frac), and Exponential decay (Exp). To make the table easier to read, we highlight each cell according to the relative performance gain in comparison to the IMS baseline (top row in the table).

ferent dimensionalities and learning techniques: Word2vec embeddings trained on Wikipedia, with the Skip-gram model for dimensionalities 50, 300 and 500 (for comparison reasons) and CBOW with 300 dimensions, Word2vec trained on the Google News corpus with 300 dimensions and the Skip-gram model, the 300 dimensional embeddings of GloVe, and the 50 dimensional C&W embeddings. Additionally we include experiments on a non-embedding model, a PMI-SVD vector space model trained by Baroni et al. (2014).

Table 6 lists the performance of our system with different word representations in vector space on the Senseval-2 English Lexical Sample task. The results corroborate the findings of Levy et al. (2015) that Skip-gram is more efficient in captur-

Word representations	Dim.	Combination strategy			
		Concatenation	Average	Fractional	Exponential
Skip-gram - GoogleNews	300	65.5	65.5	69.4	**69.6**
GloVe	300	61.7	66.3	66.7	**68.3**
CBOW - Wiki	300	65.1	65.4	**68.9**	68.8
Skip-gram - Wiki	300	65.2	65.6	68.9	**69.7**
PMI - SVD - Wiki	500	65.5	65.3	**67.3**	66.8
Skip-gram - Wiki	500	65.1	65.6	69.1	**69.9**
Collobert & Weston	50	58.6	**67.3**	62.9	64.3
Skip-gram - Wiki	50	65.0	65.7	68.3	**68.6**

Table 6: F1 percentage performance on the Senseval-2 English Lexical Sample dataset with different word representations models, vector dimensionalities (Dim.) and combination strategies.

ing the semantics than CBOW and GloVe. Additionally, the use of embeddings with decay fares well, independently of the type of embedding. The only exception is the C&W embeddings, for which the average strategy works best. We attribute this behavior to the nature of these embeddings, rather than to their dimensionality. This is shown in our comparison against the 50-dimensional Skip-gram embeddings trained on the Wikipedia corpus (bottom of Table 6), which performs well with both decay strategies, outperforming C&W embeddings.

7 Conclusions

In this paper we studied different ways of integrating the semantic knowledge of word embeddings in the framework of WSD. We carried out a deep analysis of different parameters and strategies across several WSD tasks. We draw three main findings. First, word embeddings can be used as new features to improve a state-of-the-art supervised WSD that only uses standard features. Second, integrating embeddings on the basis of an exponential decay strategy proves to be more consistent in producing high performance than the other conventional strategies, such as vector concatenation and centroid. Third, the retrofitted embeddings that take advantage of the knowledge derived from semi-structured resources, when used as the only feature for WSD can outperform state-of-the-art supervised models which use standard WSD features. However, the best performance is obtained when standard WSD features are augmented with the additional knowledge from Word2vec vectors on the basis of a decay function strategy. Our hope is that this work will serve as the first step for further studies on re-designing standard WSD features. We release at `https://github.com/iiacobac/ims_wsd_emb` all the codes and resources used in our experiments in order to provide a framework for research on the evaluation of new VSM models in the WSD framework. As future work, we plan to investigate the possibility of designing word representations that best suit the WSD framework.

Acknowledgments

The authors gratefully acknowledge the support of the ERC Starting Grant MultiJEDI No. 259234.

References

Eneko Agirre and Aitor Soroa. 2009. Personalizing PageRank For Word Sense Disambiguation. In *Proceedings of the 12th Conference of the EACL*, pages 33–41, Athens, Greece.

Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. 2006. Two graph-based algorithms for state-of-the-art wsd. In *Proceedings of the 2006 EMNLP*, pages 585–593, Sydney, Australia.

Eneko Agirre, Aitor Soroa, and Mark Stevenson. 2010. Graph-based word sense disambiguation of biomedical documents. *Bioinformatics*, 26(22):2889–2896.

Eneko Agirre, Oier López de Lacalle, and Aitor Soroa. 2014. Random Walks for Knowledge-based Word Sense Disambiguation. *Comp. Ling.*, 40(1):57–84.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 36th ACL*, pages 86–90, Montreal, Quebec, Canada.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A

systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd ACL*, volume 1, pages 238–247, Baltimore, Maryland.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Samuel Brody and Mirella Lapata. 2009. Bayesian word sense induction. In *Proceedings of the 12th Conference of the EACL*, pages 103–111, Athens, Greece.

Jun Fu Cai, Wee Sun Lee, and Yee Whye Teh. 2007. NUS-ML:Improving Word Sense Disambiguation Using Topic Features. In *Proceedings of the SemEval-2007*, pages 249–252, Prague, Czech Republic.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015a. A Unified Multilingual Semantic Representation of Concepts. In *Proceedings of the 53rd ACL*, volume 1, pages 741–751, Beijing, China.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015b. NASARI: a novel approach to a semantically-aware representation of items. In *Proceedings of the 2015 NAACL*, pages 567–577, Denver, Colorado.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 EMNLP*, pages 1025–1035, Doha, Qatar.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th ICML*, pages 160–167, Helsinki, Finland.

Antonio Di Marco and Roberto Navigli. 2013. Clustering and diversifying web search results with graph-based word sense induction. *Comp. Ling.*, 39(3):709–754.

Philip Edmonds and Scott Cotton. 2001. Senseval-2: Overview. In *The Proceedings of the 2nd International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France.

Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2015. Retrofitting Word Vectors to Semantic Lexicons. In *Proceedings of the 2015 NAACL*, pages 1606–1615, Denver, Colorado.

J. R. Firth. 1957. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The Paraphrase Database. In *Proceedings 2013 NAACL*, pages 758–764, Atlanta, Georgia.

Weiwei Guo and Mona Diab. 2010. Combining Orthogonal Monolingual and Multilingual Sources of Evidence for All Words WSD. In *Proceedings of the 48th ACL*, pages 1542–1551, Uppsala, Sweden.

Taher H. Haveliwala. 2002. Topic-sensitive PageRank. In *Proceedings of the 11th international conference on World Wide Web*, pages 517–526, Honolulu, Hawai.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving Word Representations Via Global Context And Multiple Word Prototypes. In *Proceedings of 50th ACL*, volume 1, pages 873–882, Jeju Island, South Korea.

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. SensEmbed: Learning Sense Embeddings for Word and Relational Similarity. In *Proceedings of the 53rd ACL*, volume 1, pages 95–105, Beijing, China.

Matt Insall, Todd Rowland, and Eric W. Weisstein. 2015. "Embedding". From MathWorld–A Wolfram Web Resource (access Sep 11, 2015) `http://mathworld.wolfram.com/Embedding.html`.

Thomas K. Landauer and Susan T. Dutnais. 1997. A Solution to Platos Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104(2):211–240.

Yoong Keok Lee and Hwee Tou Ng. 2002. An Empirical Evaluation of Knowledge Sources and Learning Algorithms for Word Sense Disambiguation. In *Proceedings of the 2002 EMNLP*, volume 10, pages 41–48, Philadelphia, Pennsylvania.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer S. Pradhan. 2010. Semeval-2010 task 14: Word sense induction & disambiguation. In *Proceedings of SemEval-2010*, pages 63–68, Uppsala, Sweden.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of Recurrent-neural-network Architectures and Learning Methods for Spoken Language Understanding. In *INTERSPEECH*, pages 3771–3775, Lyon, France.

Rada Mihalcea and Ehsanul Faruque. 2004. Senselearner: Minimally Supervised Word Sense Disambiguation for All Words in Open Text. In *Proceedings of ACL/SIGLEX Senseval-3*, volume 3, pages 155–158, Barcelona, Spain.

Rada Mihalcea, Timothy Chklovski, and Adam Kilgarriff. 2004. The Senseval-3 English Lexical Sample Task. In *Proceedings of ACL/SIGLEX Senseval-3*, pages 25–28, Barcelona, Spain.

Tomas Mikolov, Quoc V. Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

George A. Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. 1994. Using a Semantic Concordance for Sense Identification. In *Proceedings of the Workshop on HLT*, pages 240–243, Plainsboro, New Jersey.

Tristan Miller, Chris Biemann, Torsten Zesch, and Iryna Gurevych. 2012. Using Distributional Similarity for Lexical Expansion in Knowledge-based Word Sense Disambiguation. In *COLING*, pages 1781–1796, Mumbai, India.

George A. Miller. 1995. WordNet: A Lexical Database for English. *Comm. ACM*, 38(11):39–41.

Saif Mohammad and Graeme Hirst. 2006. Determining Word Sense Dominance Using a Thesaurus. In *Proceedings of the 11th Conference of EACL*, pages 121–128, Trento, Italy.

Andrea Moro, Alessandro Raganato, and Roberto Navigli. 2014. Entity Linking meets Word Sense Disambiguation: a Unified Approach. *Transactions of the ACL*, 2:231–244.

Roberto Navigli. 2009. Word sense disambiguation: a survey. *ACM COMPUTING SURVEYS*, 41(2):1–69.

Hwee Tou Ng and Hian Beng Lee. 1996. Integrating Multiple Knowledge Sources to Disambiguate Word Sense: An Exemplar-based Approach. In *Proceedings of the 34th Meeting on ACL*, pages 40–47, Santa Cruz, California.

Siddharth Patwardhan and Ted Pedersen. 2006. Using WordNet-based Context Vectors to Estimate the Semantic Relatedness of Concepts. In *Proceedings of the EACL 2006 Workshop Making Sense of Sense*, volume 1501, pages 1–8, Trento, Italy.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 EMNLP*, pages 1532–1543, Doha, Qatar.

Mohammad Taher Pilehvar and Roberto Navigli. 2014. A Large-scale Pseudoword-based Evaluation Framework for State-of-the-Art Word Sense Disambiguation. *Computational Linguistics*, 40(4):837–881.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich word sense disambiguation rivaling supervised systems. In *Proceedings of the 48th ACL*, pages 1522–1531, Uppsala, Sweden.

Sameer S. Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007. SemEval-2007 Task 17: English Lexical Sample, SRL and All Words. In *Proceedings of the SemEval-2007*, pages 87–92, Prague, Czech Republic.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-Prototype Vector-Space Models of Word Meaning. In *Proceedings of the 2010 Annual Conference of the NAACL*, pages 109–117, Los Angeles, California.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd ACL*, volume 1, pages 1793–1803, Beijing, China.

Hui Shen, Razvan Bunescu, and Rada Mihalcea. 2013. Coarse to Fine Grained Sense Disambiguation in Wikipedia. In **SEM 2013: The Secound Joint Conference on Lexical and Computational Semantics*, pages 22–31, Atlanta, Georgia.

Benjamin Snyder and Martha Palmer. 2004. The Senseval-3 English All-Words Task. In *Proceedings of ACL/SIGLEX Senseval-3*, pages 41–43, Barcelona, Spain.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 EMNLP*, pages 1631–1642, Seattle, USA.

Jiri Stetina, Sadao Kurohashi, and Makoto Nagao. 1998. General Word Sense Disambiguation Method Based on a Full Sentential Context. In *Usage of WordNet in Natural Language Processing, Proceedings of COLING-ACL Workshop*, Montreal, Quebec, Canada.

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-Supervised Word Sense Disambiguation Using Word Embeddings in General and Specific Domains. In *Proceedings of the 2015 Annual Conference of the NAACL*, pages 314–323, Denver, Colorado.

Tim Van de Cruys and Marianna Apidianaki. 2011. Latent Semantic Word Sense Induction and Disambiguation. In *Proceedings of the 49th ACL*, volume 1, pages 1476–1485, Portland, Oregon.

Mo Yu and Mark Dredze. 2014. Improving Lexical Embeddings with Semantic Knowledge. In *Proceedings of the 52nd ACL*, volume 2, pages 545–550, Baltimore, Maryland.

Zhi Zhong and Hwee Tou Ng. 2010. It Makes Sense: A Wide-coverage Word Sense Disambiguation System for Free Text. In *Proceedings of the 48th ACL*, pages 78–83, Uppsala, Sweden.

Text Understanding with the Attention Sum Reader Network

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar & Jan Kleindienst
IBM Watson
V Parku 4, Prague, Czech Republic
{rudolf_kadlec,martin.schmid,obajgar,jankle}@cz.ibm.com

Abstract

Several large cloze-style context-question-answer datasets have been introduced recently: the CNN and Daily Mail news data and the Children's Book Test. Thanks to the size of these datasets, the associated text comprehension task is well suited for deep-learning techniques that currently seem to outperform all alternative approaches. We present a new, simple model that uses attention to directly pick the answer from the context as opposed to computing the answer using a blended representation of words in the document as is usual in similar models. This makes the model particularly suitable for question-answering problems where the answer is a single word from the document. Ensemble of our models sets new state of the art on all evaluated datasets.

1 Introduction

Most of the information humanity has gathered up to this point is stored in the form of plain text. Hence the task of teaching machines how to understand this data is of utmost importance in the field of Artificial Intelligence. One way of testing the level of text understanding is simply to ask the system questions for which the answer can be inferred from the text. A well-known example of a system that could make use of a huge collection of unstructured documents to answer questions is for instance IBM's Watson system used for the Jeopardy challenge (Ferrucci et al., 2010).

Cloze style questions (Taylor, 1953), i.e. questions formed by removing a phrase from a sentence, are an appealing form of such questions (for example see Figure 1). While the task is easy to evaluate, one can vary the context, the question sentence or the specific phrase missing in the question to dramatically change the task structure and difficulty.

Document: What was supposed to be a fantasy sports car ride at Walt Disney World Speedway turned deadly when a Lamborghini crashed into a guardrail. The crash took place Sunday at the Exotic Driving Experience, which bills itself as a chance to drive your dream car on a racetrack. The Lamborghini's passenger, 36-year-old Gary Terry of Davenport, Florida, died at the scene, Florida Highway Patrol said. The driver of the Lamborghini, 24-year-old Tavon Watson of Kissimmee, Florida, lost control of the vehicle, the Highway Patrol said. (...)
Question: Officials say the driver, 24-year-old Tavon Watson, lost control of a _____
Answer candidates: Tavon Watson, Walt Disney World Speedway, Highway Patrol, Lamborghini, Florida, (...)
Answer: Lamborghini

Figure 1: Each example consists of a context document, question, answer cadidates and, in the training data, the correct answer. This example was taken from the CNN dataset (Hermann et al., 2015). Anonymization of this example that makes the task harder is shown in Table 3.

One way of altering the task difficulty is to vary the word type being replaced, as in (Hill et al., 2015). The complexity of such variation comes from the fact that the level of context understanding needed in order to correctly predict different types of words varies greatly. While predicting prepositions can easily be done using relatively simple models with very little context knowledge, predicting named entities requires a deeper understanding of the context.

Also, as opposed to selecting a random sentence from a text (as done in (Hill et al., 2015)), the questions can be formed from a specific part of a document, such as a short summary or a list of

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 908–918,
Berlin, Germany, August 7-12, 2016.

	CNN			Daily Mail			CBT CN			CBT NE		
	train	valid	test	train	valid	test	train	valid	test	train	valid	test
# queries	380,298	3,924	3,198	879,450	64,835	53,182	120,769	2,000	2,500	108,719	2,000	2,500
Max # options	527	187	396	371	232	245	10	10	10	10	10	10
Avg # options	26.4	26.5	24.5	26.5	25.5	26.0	10	10	10	10	10	10
Avg # tokens	762	763	716	813	774	780	470	448	461	433	412	424
Vocab. size	118,497			208,045			53,185			53,063		

Table 1: Statistics on the 4 data sets used to evaluate the model. CBT CN stands for CBT Common Nouns and CBT NE stands for CBT Named Entites. Statistics were taken from (Hermann et al., 2015) and the statistics provided with the CBT data set.

tags. Since such sentences often paraphrase in a condensed form what was said in the text, they are particularly suitable for testing text comprehension (Hermann et al., 2015).

An important property of cloze style questions is that a large amount of such questions can be automatically generated from real world documents. This opens the task to data-hungry techniques such as deep learning. This is an advantage compared to smaller machine understanding datasets like MCTest (Richardson et al., 2013) that have only hundreds of training examples and therefore the best performing systems usually rely on handcrafted features (Sachan et al., 2015; Narasimhan and Barzilay, 2015).

In the first part of this article we introduce the task at hand and the main aspects of the relevant datasets. Then we present our own model to tackle the problem. Subsequently we compare the model to previously proposed architectures and finally describe the experimental results on the performance of our model.

2 Task and datasets

In this section we introduce the task that we are seeking to solve and relevant large-scale datasets that have recently been introduced for this task.

2.1 Formal Task Description

The task consists of answering a cloze style question, the answer to which is dependent on the understanding of a context document provided with the question. The model is also provided with a set of possible answers from which the correct one is to be selected. This can be formalized as follows:

The training data consist of tuples $(\mathbf{q}, \mathbf{d}, a, A)$, where $\mathbf{q}$ is a question, $\mathbf{d}$ is a document that contains the answer to question $\mathbf{q}$, A is a set of possible answers and $a \in A$ is the ground truth answer. Both $\mathbf{q}$ and $\mathbf{d}$ are sequences of words from vocabulary V. We also assume that all possible answers are words from the vocabulary, that is $A \subseteq V$, and that the ground truth answer a appears in the document, that is $a \in \mathbf{d}$.

2.2 Datasets

We will now briefly summarize important features of the datasets.

2.2.1 News Articles — CNN and Daily Mail

The first two datasets[1] (Hermann et al., 2015) were constructed from a large number of news articles from the CNN and Daily Mail websites. The main body of each article forms a context, while the cloze style question is formed from one of short highlight sentences, appearing at the top of each article page. Specifically, the question is created by replacing a named entity from the summary sentence (*e.g. "Producer X will not press charges against Jeremy Clarkson, his lawyer says."*).

Furthermore the named entities in the whole dataset were replaced by anonymous tokens which were further shuffled for each example so that the model cannot build up any world knowledge about the entities and hence has to genuinely rely on the context document to search for an answer to the question.

Qualitative analysis of reasoning patterns needed to answer questions in the CNN dataset together with human performance on this task are provided in (Chen et al., 2016).

[1]The CNN and Daily Mail datasets are available at `https://github.com/deepmind/rc-data`

2.2.2 Children's Book Test

The third dataset[2], the Children's Book Test (CBT) (Hill et al., 2015), is built from books that are freely available thanks to Project Gutenberg[3]. Each context document is formed by 20 consecutive sentences taken from a children's book story. Due to the lack of summary, the cloze style question is then constructed from the subsequent (21^{st}) sentence.

One can also see how the task complexity varies with the type of the omitted word (named entity, common noun, verb, preposition). (Hill et al., 2015) have shown that while standard LSTM language models have human level performance on predicting verbs and prepositions, they lack behind on named entities and common nouns. In this article we therefore focus only on predicting the first two word types.

Basic statistics about the CNN, Daily Mail and CBT datasets are summarized in Table 1.

3 Our Model — Attention Sum Reader

Our model called the *Attention Sum Reader (AS Reader)*[4] is tailor-made to leverage the fact that the answer is a word from the context document. This is a double-edged sword. While it achieves state-of-the-art results on all of the mentioned datasets (where this assumption holds true), it cannot produce an answer which is not contained in the document. Intuitively, our model is structured as follows:

1. We compute a vector embedding of the query.
2. We compute a vector embedding of each individual word in the context of the whole document (*contextual embedding*).
3. Using a dot product between the question embedding and the contextual embedding of each occurrence of a candidate answer in the document, we select the most likely answer.

3.1 Formal Description

Our model uses one word embedding function and two encoder functions. The word embedding function e translates words into vector representations. The first encoder function is a document encoder f that encodes every word from the document $\mathbf{d}$ in the context of the whole document. We call this the *contextual embedding*. For convenience we will denote the contextual embedding of the i-th word in $\mathbf{d}$ as $f_i(\mathbf{d})$. The second encoder g is used to translate the query $\mathbf{q}$ into a fixed length representation of the same dimensionality as each $f_i(\mathbf{d})$. Both encoders use word embeddings computed by e as their input. Then we compute a weight for every word in the document as the dot product of its contextual embedding and the query embedding. This weight might be viewed as an attention over the document $\mathbf{d}$.

To form a proper probability distribution over the words in the document, we normalize the weights using the *softmax* function. This way we model probability s_i that the answer to query $\mathbf{q}$ appears at position i in the document $\mathbf{d}$. In a functional form this is:

$$s_i \propto \exp\left(f_i(\mathbf{d}) \cdot g(\mathbf{q})\right) \tag{1}$$

Finally we compute the probability that word w is a correct answer as:

$$P(w|\mathbf{q},\mathbf{d}) \propto \sum_{i \in I(w,\mathbf{d})} s_i \tag{2}$$

where $I(w,\mathbf{d})$ is a set of positions where w appears in the document $\mathbf{d}$. We call this mechanism *pointer sum attention* since we use attention as a pointer over discrete tokens in the context document and then we directly sum the word's attention across all the occurrences. This differs from the usual use of attention in sequence-to-sequence models (Bahdanau et al., 2015) where attention is used to blend representations of words into a new embedding vector. Our use of attention was inspired by Pointer Networks (Ptr-Nets) (Vinyals et al., 2015).

A high level structure of our model is shown in Figure 2.

3.2 Model instance details

In our model the document encoder f is implemented as a bidirectional Gated Recurrent Unit (GRU) network (Cho et al., 2014; Chung et al., 2014) whose hidden states form the contextual word embeddings, that is $f_i(\mathbf{d}) = \overrightarrow{f_i}(\mathbf{d}) \mid\mid \overleftarrow{f_i}(\mathbf{d})$, where $\mid\mid$ denotes vector concatenation and $\overrightarrow{f_i}$ and

[2]The CBT dataset is available at `http://www.thespermwhale.com/jaseweston/babi/CBTest.tgz`

[3]`https://www.gutenberg.org/`

[4]An implementation of AS Reader is available at `https://github.com/rkadlec/asreader`

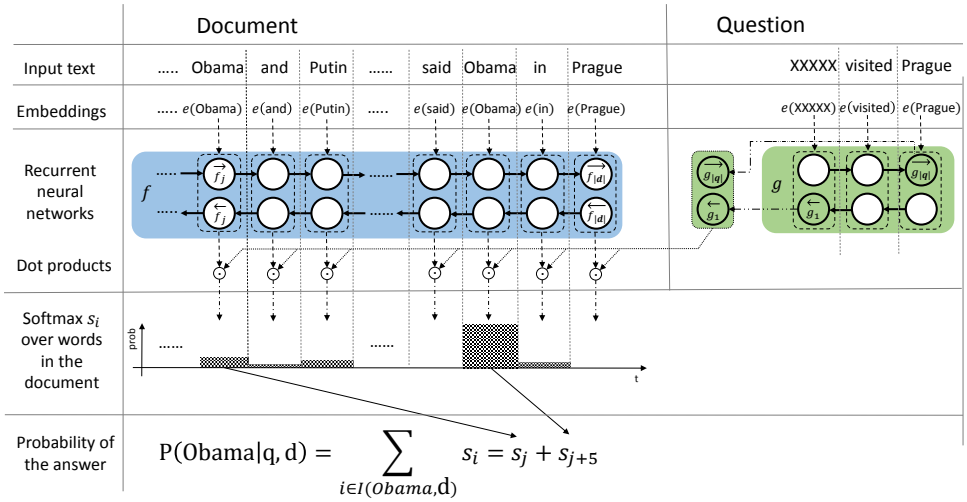


Figure 2: Structure of the model.

...

what was supposed to be a fantasy sports car ride at @entity3 turned deadly when a @entity4 crashed into a guardrail . the crash took place sunday at the @entity8 , which bills itself as a chance to drive your dream car on a racetrack . the @entity4 's passenger , 36 - year - old @entity14 of @entity15 , @entity16 , died at the scene , @entity13 said . the driver of the @entity4 , 24 - year - old @entity18 of @entity19 , @entity16 , lost control of the vehicle , the @entity13 said .

...

0 1

officials say the driver , 24 - year - old @entity18 , lost control of a _____

Figure 3: Attention in an example with anonymized entities where our system selected the correct answer. Note that the attention is focused only on named entities.

...

@entity11 film critic @entity29 writes in his review that "anyone nostalgic for childhood dreams of transformation will find something to enjoy in an uplifting movie that invests warm sentiment in universal themes of loss and resilience , experience and maturity . " more : the best and worst adaptations of "@entity" @entity43, @entity44 and @entity46 star in director @entity48's crime film about a hit man trying to save his estranged son from a revenge plot. @entity11 chief film critic @entity52 writes in his review that the film

...

0 1

_____ stars in crime film about hit man trying to save his estranged son

Figure 4: Attention over an example where our system failed to select the correct answer (entity43). The system was probably mislead by the co-occurring word 'film'. Namely, entity11 occurs 7 times in the whole document and 6 times it is together with the word 'film'. On the other hand, the correct answer occurs only 3 times in total and only once together with 'film'.

$\overleftarrow{f_i}$ denote forward and backward contextual embeddings from the respective recurrent networks. The query encoder g is implemented by another bidirectional GRU network. This time the last hidden state of the forward network is concatenated with the last hidden state of the backward network to form the query embedding, that is $g(\mathbf{q}) = \overrightarrow{g_{|\mathbf{q}|}}(\mathbf{q}) \mid\mid \overleftarrow{g_1}(\mathbf{q})$. The word embedding function e is implemented in a usual way as a look-up table $\mathbf{V}$. $\mathbf{V}$ is a matrix whose rows can be indexed by words from the vocabulary, that is $e(w) = V_w, w \in V$. Therefore, each row of $\mathbf{V}$ contains embedding of one word from the vocabulary. During training we jointly optimize parameters of f, g and e.

4 Related Work

Several recent deep neural network architectures (Hermann et al., 2015; Hill et al., 2015; Chen et al., 2016; Kobayashi et al., 2016) were applied to the task of text comprehension. The last two architectures were developed independently at the same time as our work. All of these architectures use an attention mechanism that allows them to highlight places in the document that might be relevant to answering the question. We will now briefly describe these architectures and compare

them to our approach.

4.1 Attentive and Impatient Readers

Attentive and *Impatient Readers* were proposed in (Hermann et al., 2015). The simpler Attentive Reader is very similar to our architecture. It also uses bidirectional document and query encoders to compute an attention in a similar way we do. The more complex Impatient Reader computes attention over the document after reading every word of the query. However, empirical evaluation has shown that both models perform almost identically on the CNN and Daily Mail datasets.

The key difference between the Attentive Reader and our model is that the Attentive Reader uses attention to compute a fixed length representation r of the document $\mathbf{d}$ that is equal to a weighted sum of contextual embeddings of words in $\mathbf{d}$, that is $r = \sum_i s_i f_i(\mathbf{d})$. A joint query and document embedding m is then a non-linear function of r and the query embedding $g(\mathbf{q})$. This joint embedding m is in the end compared against all candidate answers $a' \in A$ using the dot product $e(a') \cdot m$, in the end the scores are normalized by softmax. That is: $P(a'|\mathbf{q}, \mathbf{d}) \propto \exp(e(a') \cdot m)$.

In contrast to the Attentive Reader, we select the answer from the context directly using the computed attention rather than using such attention for a weighted sum of the individual representations (see Eq. 2). The motivation for such simplification is the following.

Consider a context "*A UFO was observed above our city in January and again in March.*" and question "*An observer has spotted a UFO in ___ .*"

Since both January and March are equally good candidates, the attention mechanism might put the same attention on both these candidates in the context. The blending mechanism described above would compute a vector between the representations of these two words and propose the closest word as the answer - this may well happen to be February (it is indeed the case for Word2Vec trained on Google News). By contrast, our model would correctly propose January or March.

4.2 Chen et al. 2016

A model presented in (Chen et al., 2016) is inspired by the Attentive Reader. One difference is that the attention weights are computed with a bilinear term instead of simple dot-product, that is: $s_i \propto \exp(f_i(\mathbf{d})^\intercal W\, g(\mathbf{q}))$. The document embedding r is computed using a weighted sum as in the Attentive Reader: $r = \sum_i s_i f_i(\mathbf{d})$. In the end $P(a'|\mathbf{q}, \mathbf{d}) \propto \exp(e'(a') \cdot r)$, where e' is a new embedding function.

Even though it is a simplification of the Attentive Reader this model performs significantly better than the original.

4.3 Memory Networks

MemNNs (Weston et al., 2014) were applied to the task of text comprehension in (Hill et al., 2015).

The best performing memory networks model setup - window memory - uses windows of fixed length (8) centered around the candidate words as memory cells. Due to this limited context window, the model is unable to capture dependencies out of scope of this window. Furthermore, the representation within such window is computed simply as the sum of embeddings of words in that window. By contrast, in our model the representation of each individual word is computed using a recurrent network, which not only allows it to capture context from the entire document but also the embedding computation is much more flexible than a simple sum.

To improve on the initial accuracy, a heuristic approach called *self supervision* is used in (Hill et al., 2015) to help the network to select the right supporting "memories" using an attention mechanism showing similarities to the ours. Plain MemNNs without this heuristic are not competitive on these machine reading tasks. Our model does not need any similar heuristics.

4.4 Dynamic Entity Representation

The Dynamic Entity Representation model (Kobayashi et al., 2016) has a complex architecture also based on the weighted attention mechanism and max pooling over contextual embeddings of vectors for each named entity.

4.5 Pointer Networks

Our model architecture was inspired by Ptr-Nets (Vinyals et al., 2015) in using an attention mechanism to select the answer in the context rather than to blend words from the context into an answer representation. While a Ptr-Net consists of an encoder as well as a decoder, which uses the attention to select the output at each step, our model outputs the answer in a single step. Furthermore,

the pointer networks assume that no input in the sequence appears more than once, which is not the case in our settings.

4.6 Summary

Our model combines the best features of the architectures mentioned above. We use recurrent networks to "read" the document and the query as done in (Hermann et al., 2015; Chen et al., 2016; Kobayashi et al., 2016) and we use attention in a way similar to Ptr-Nets. We also use summation of attention weights in a way similar to MemNNs (Hill et al., 2015).

From a high level perspective we simplify all the discussed text comprehension models by removing all transformations past the attention step. Instead we use the attention directly to compute the answer probability.

5 Evaluation

In this section we evaluate our model on the CNN, Daily Mail and CBT datasets. We show that despite the model's simplicity its ensembles achieve state-of-the-art performance on each of these datasets.

5.1 Training Details

To train the model we used stochastic gradient descent with the ADAM update rule (Kingma and Ba, 2015) and learning rate of 0.001 or 0.0005. During training we minimized the following negative log-likelihood with respect to θ:

$$-logP_\theta(a|\mathbf{q},\mathbf{d}) \tag{3}$$

where a is the correct answer for query $\mathbf{q}$ and document $\mathbf{d}$, and θ represents parameters of the encoder functions f and g and of the word embedding function e. The optimized probability distribution $P(a|\mathbf{q},\mathbf{d})$ is defined in Eq. 2.

The initial weights in the word embedding matrix were drawn randomly uniformly from the interval $[-0.1, 0.1]$. Weights in the GRU networks were initialized by random orthogonal matrices (Saxe et al., 2014) and biases were initialized to zero. We also used a gradient clipping (Pascanu et al., 2012) threshold of 10 and batches of size 32.

During training we randomly shuffled all examples in each epoch. To speedup training, we always pre-fetched 10 batches worth of examples and sorted them according to document length. Hence each batch contained documents of roughly the same length.

For each batch of the CNN and Daily Mail datasets we randomly reshuffled the assignment of named entities to the corresponding word embedding vectors to match the procedure proposed in (Hermann et al., 2015). This guaranteed that word embeddings of named entities were used only as semantically meaningless labels not encoding any intrinsic features of the represented entities. This forced the model to truly deduce the answer from the single context document associated with the question. We also do not use pre-trained word embeddings to make our training procedure comparable to (Hermann et al., 2015).

We did not perform any text pre-processing since the original datasets were already tokenized.

We do not use any regularization since in our experience it leads to longer training times of single models, however, performance of a model ensemble is usually the same. This way we can train the whole ensemble faster when using multiple GPUs for parallel training.

For Additional details about the training procedure see Appendix A.

5.2 Evaluation Method

We evaluated the proposed model both as a single model and using ensemble averaging. Although the model computes attention for every word in the document we restrict the model to select an answer from a list of candidate answers associated with each question-document pair.

For single models we are reporting results for the best model as well as the average of accuracies for the best 20% of models with best performance on validation data since single models display considerable variation of results due to random weight initialization[5], even for identical hyperparameter values. Single model performance may consequently prove difficult to reproduce.

What concerns ensembles, we used simple averaging of the answer probabilities predicted by ensemble members. For ensembling we used 14, 16, 84 and 53 models for CNN, Daily Mail and CBT CN and NE respectively. The ensemble models were chosen either as the top 70% of all trained models, we call this *avg ensemble*. Alternatively we use the following algorithm: We started with

[5]The standard deviation for models with the same hyperparameters was between 0.6-2.5% in absolute test accuracy.

	CNN		Daily Mail	
	valid	test	valid	test
Attentive Reader †	61.6	63.0	70.5	69.0
Impatient Reader †	61.8	63.8	69.0	68.0
MemNNs (single model) ‡	63.4	66.8	NA	NA
MemNNs (ensemble) ‡	66.2	69.4	NA	NA
Dynamic Entity Repres. (max-pool) ♯	71.2	70.7	NA	NA
Dynamic Entity Repres. (max-pool + byway)♯	70.8	72.0	NA	NA
Dynamic Entity Repres. + w2v ♯	71.3	72.9	NA	NA
Chen et al. (2016) (single model)	72.4	72.4	76.9	75.8
AS Reader (single model)	68.6	69.5	75.0	73.9
AS Reader (avg for top 20%)	68.4	69.9	74.5	73.5
AS Reader (avg ensemble)	73.9	**75.4**	78.1	77.1
AS Reader (greedy ensemble)	74.5	74.8	78.7	**77.7**

Table 2: Results of our AS Reader on the CNN and Daily Mail datasets. Results for models marked with † are taken from (Hermann et al., 2015), results of models marked with ‡ are taken from (Hill et al., 2015) and results marked with ♯ are taken from (Kobayashi et al., 2016). Performance of ‡ and ♯ models was evaluated only on CNN dataset.

	Named entity		Common noun	
	valid	test	valid	test
Humans (query) (*)	NA	52.0	NA	64.4
Humans (context+query) (*)	NA	***81.6***	NA	***81.6***
LSTMs (context+query) ‡	51.2	41.8	62.6	56.0
MemNNs (window memory + self-sup.) ‡	70.4	66.6	64.2	63.0
AS Reader (single model)	73.8	68.6	68.8	63.4
AS Reader (avg for top 20%)	73.3	68.4	67.7	63.2
AS Reader (avg ensemble)	74.5	70.6	71.1	**68.9**
AS Reader (greedy ensemble)	76.2	**71.0**	72.4	67.5

Table 3: Results of our AS Reader on the CBT datasets. Results marked with ‡ are taken from (Hill et al., 2015). (*)Human results were collected on 10% of the test set.

the best performing model according to validation performance. Then in each step we tried adding the best performing model that had not been previously tried. We kept it in the ensemble if it did improve its validation performance and discarded it otherwise. This way we gradually tried each model once. We call the resulting model a *greedy ensemble*.

5.3 Results

Performance of our models on the CNN and Daily Mail datasets is summarized in Table 2, Table 3 shows results on the CBT dataset. The tables also list performance of other published models that were evaluated on these datasets. Ensembles of our models set the new state-of-the-art results on all evaluated datasets.

Table 4 then measures accuracy as the proportion of test cases where the ground truth was among the top k answers proposed by the greedy ensemble model for $k = 1, 2, 5$.

CNN and Daily Mail. The CNN dataset is the most widely used dataset for evaluation of text comprehension systems published so far. Perfor-

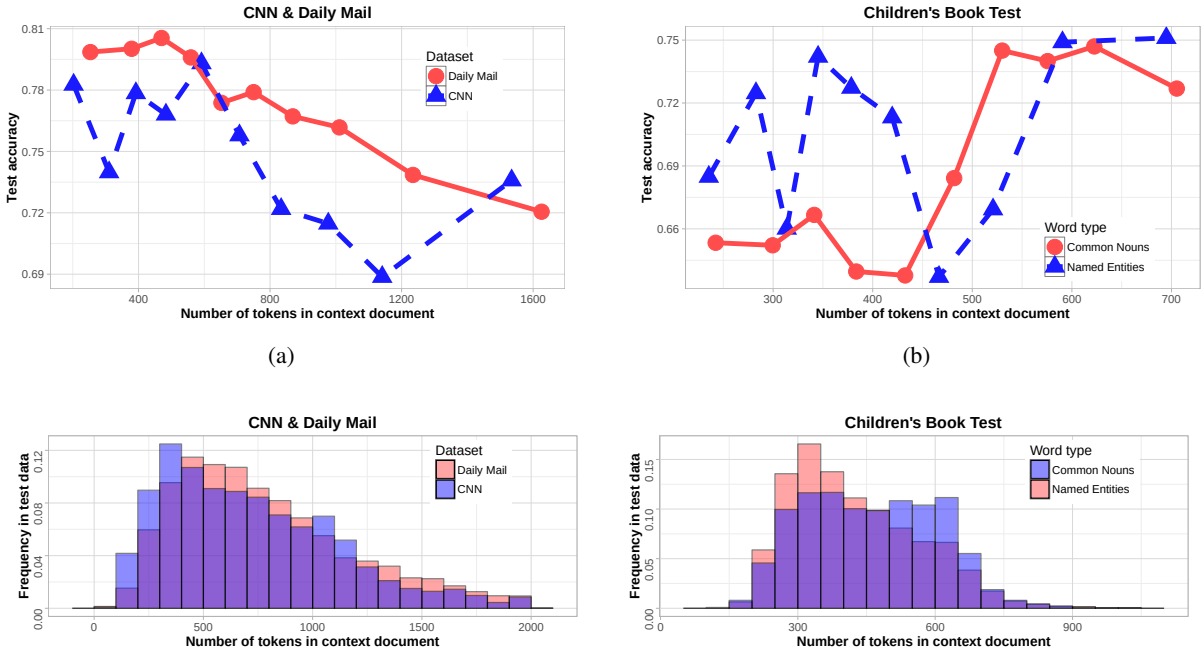


Figure 5: Sub-figures (a) and (b) plot the test accuracy against the length of the context document. The examples were split into ten buckets of equal size by their context length. Averages for each bucket are plotted on each axis. Sub-figures (c) and (d) show distributions of context lengths in the four datasets.

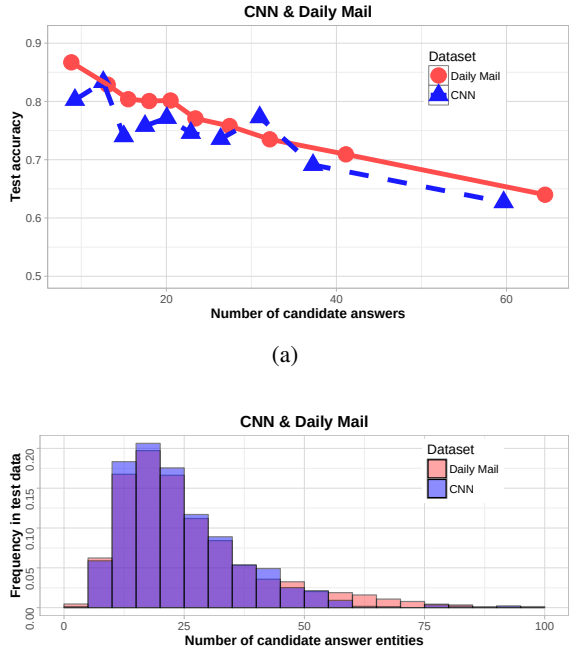


Figure 6: Subfigure (a) illustrates how the model accuracy decreases with an increasing number of candidate named entities. Subfigure (b) shows the overall distribution of the number of candidate answers in the news datasets.

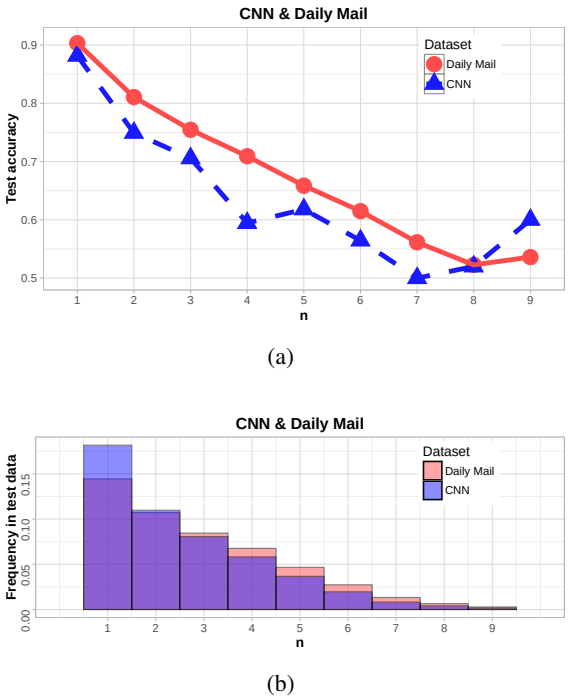


Figure 7: Subfigure (a) shows the model accuracy when the correct answer is the n^{th} most frequent named entity for $n \in [1, 10]$. Subfigure (b) shows the number of test examples for which the correct answer was the n–th most frequent entity. The plots for CBT look almost identical (see Appendix B).

mance of our single model is a little bit worse than performance of simultaneously published models (Chen et al., 2016; Kobayashi et al., 2016). Compared to our work these models were trained with Dropout regularization (Srivastava et al., 2014) which might improve single model performance. However, ensemble of our models outperforms these models even though they use pre-trained word embeddings.

On the CNN dataset our single model with best validation accuracy achieves a test accuracy of 69.5%. The average performance of the top 20% models according to validation accuracy is 69.9% which is even 0.5% better than the single best-validation model. This shows that there were many models that performed better on test set than the best-validation model. Fusing multiple models then gives a significant further increase in accuracy on both CNN and Daily Mail datasets..

CBT. In named entity prediction our best single model with accuracy of 68.6% performs 2% absolute better than the MemNN with self supervision, the averaging ensemble performs 4% absolute better than the best previous result. In common noun prediction our single models is 0.4% absolute better than MemNN however the ensemble improves the performance to 69% which is 6% absolute better than MemNN.

Dataset	$k = 1$	$k = 2$	$k = 5$
CNN	74.8	85.5	94.8
Daily Mail	77.7	87.6	94.8
CBT NE	71.0	86.9	96.8
CBT CN	67.5	82.5	95.4

Table 4: Proportion of test examples for which the top k answers proposed by the greedy ensemble included the correct answer.

6 Analysis

To further analyze the properties of our model, we examined the dependence of accuracy on the length of the context document (Figure 5), the number of candidate answers (Figure 6) and the frequency of the correct answer in the context (Figure 7).

On the CNN and Daily Mail datasets, the accuracy decreases with increasing document length (Figure 5a). We hypothesize this may be due to multiple factors. Firstly long documents may make the task more complex. Secondly such cases are quite rare in the training data (Figure 5b) which motivates the model to specialize on shorter contexts. Finally the context length is correlated with the number of named entities, i.e. the number of possible answers which is itself negatively correlated with accuracy (see Figure 6).

On the CBT dataset this negative trend seems to disappear (Fig. 5c). This supports the later two explanations since the distribution of document lengths is somewhat more uniform (Figure 5d) and the number of candidate answers is constant (10) for all examples in this dataset.

The effect of increasing number of candidate answers on the model's accuracy can be seen in Figure 6a. We can clearly see that as the number of candidate answers increases, the accuracy drops. On the other hand, the amount of examples with large number of candidate answers is quite small (Figure 6b).

Finally, since the summation of attention in our model inherently favours frequently occurring tokens, we also visualize how the accuracy depends on the frequency of the correct answer in the document. Figure 7a shows that the accuracy significantly drops as the correct answer gets less and less frequent in the document compared to other candidate answers. On the other hand, the correct answer is likely to occur frequently (Fig. 7a).

7 Conclusion

In this article we presented a new neural network architecture for natural language text comprehension. While our model is simpler than previously published models, it gives a new state-of-the-art accuracy on all the evaluated datasets.

An analysis by (Chen et al., 2016) suggests that on CNN and Daily Mail datasets a significant proportion of questions is ambiguous or too difficult to answer even for humans (partly due to entity anonymization) so the ensemble of our models may be very near to the maximal accuracy achievable on these datasets.

Acknowledgments

We would also like to thank Tim Klinger for providing us with masked softmax code that we used in our implementation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations*.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Danqi Chen, Jason Bolton, and Christopher D. Manning. 2016. A Thorough Examination of the CNN / Daily Mail Reading Comprehension Task. In *Association for Computational Linguistics (ACL)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Empirical Methods in Natural Language Processing (EMNLP)*.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv*, pages 1–9.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya a. Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59–79.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. *International Conference on Learning Representations*, pages 1–13.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic Entity Representation with Max-pooling Improves Machine Reading. *Proceedings of the North American Chapter of the Association for Computational Linguistics and Human Language Technologies (NAACL-HLT)*.

Karthik Narasimhan and Regina Barzilay. 2015. Machine Comprehension with Discourse Relations. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1253–1262.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *Proceedings of The 30th International Conference on Machine Learning*, pages 1310–1318.

Matthew Richardson, Christopher J C Burges, and Erin Renshaw. 2013. MCTest: A Challenge Dataset for the Open-Domain Machine Comprehension of Text. *Empirical Methods in Natural Language Processing (EMNLP)*, pages 193–203.

Mrinmaya Sachan, Avinava Dubey, Eric P Xing, and Matthew Richardson. 2015. Learning Answer-Entailing Structures for Machine Comprehension. *Association for Computational Linguistics (ACL)*, pages 239–249.

Andrew M Saxe, James L Mcclelland, and Surya Ganguli. 2014. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: prevent NN from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Wilson L Taylor. 1953. Cloze procedure: a new tool for measuring readability. *Journalism and Mass Communication Quarterly*, 30(4):415.

Bart van Merrienboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and Fuel : Frameworks for deep learning. pages 1–5.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Appendix A Training Details

During training we evaluated the model performance after each epoch and stopped the training when the error on the validation set started increasing.

The models usually converged after two epochs of training. Time needed to complete a single epoch of training on each dataset on an Nvidia K40 GPU is shown in Table 5.

Dataset	Time per epoch
CNN	10h 22min
Daily Mail	25h 42min
CBT Named Entity	1h 5min
CBT Common Noun	0h 56min

Table 5: Average duration of one epoch of training on the four datasets.

The hyperparameters, namely the recurrent hidden layer dimension and the source embedding dimension, were chosen by grid search. We started with a range of 128 to 384 for both parameters and subsequently kept increasing the upper bound by 128 until we started observing a consistent decrease in validation accuracy. The region of the parameter space that we explored together with the parameters of the model with best validation accuracy are summarized in Table 6.

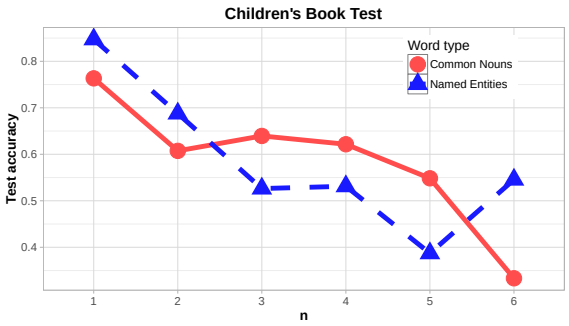
	Rec. Hid. Layer			Embedding		
Dataset	min	max	best	min	max	best
CNN	128	512	384	128	512	128
Daily Mail	128	1024	512	128	512	384
CBT NE	128	512	384	128	512	384
CBT CN	128	1536	256	128	512	384

Table 6: Dimension of the recurrent hidden layer and of the source embedding for the best model and the range of values that we tested. We report number of hidden units of the unidirectional GRU; the bidirectional GRU has twice as many hidden units.

Our model was implemented using Theano (Bastien et al., 2012) and Blocks (van Merrienboer et al., 2015).

Appendix B Dependence of accuracy on the frequency of the correct answer

In Section 6 we analysed how the test accuracy depends on how frequent the correct answer is compared to other answer candidates for the news datasets. The plots for the Children's Book Test looks very similar, however we are adding it here for completeness.



(a)

(b)

Figure 8: Subfigure (a) shows the model accuracy when the correct answer is among n most frequent named entities for $n \in [1, 10]$. Subfigure (b) shows the number of test examples for which the correct answer was the n–th most frequent entity.

Investigating LSTMs for Joint Extraction of Opinion Entities and Relations

Arzoo Katiyar and **Claire Cardie**
Department of Computer Science
Cornell University
Ithaca, NY, 14853, USA
arzoo, cardie@cs.cornell.edu

Abstract

We investigate the use of deep bi-directional LSTMs for joint extraction of opinion entities and the IS-FROM and IS-ABOUT relations that connect them — the first such attempt using a deep learning approach. Perhaps surprisingly, we find that standard LSTMs are not competitive with a state-of-the-art CRF+ILP joint inference approach (Yang and Cardie, 2013) to opinion entities extraction, performing below even the standalone sequence-tagging CRF. Incorporating sentence-level and a novel relation-level optimization, however, allows the LSTM to identify opinion relations and to perform within 1–3% of the state-of-the-art joint model for opinion entities and the IS-FROM relation; and to perform as well as the state-of-the-art for the IS-ABOUT relation — all without access to opinion lexicons, parsers and other preprocessing components required for the feature-rich CRF+ILP approach.

1 Introduction

There has been much research in recent years in the area of fine-grained opinion analysis where the goal is to identify subjective expressions in text along with their associated sources and targets. More specifically, fine-grained opinion analysis aims to identify three types of *opinion entities*:

- **opinion expressions**, O, which are direct subjective expressions (i.e., explicit mentions of otherwise private states or speech events expressing private states (Wiebe and Cardie, 2005));
- **opinion targets**, T, which are the entities or topics that the opinion is about; and
- **opinion holders**, H, which are the entities expressing the opinion.

In addition, the task involves identifying the IS-FROM and IS-ABOUT relations between an opinion expression and its holder and target, respectively. In the sample sentences, numerical subscripts indicate an IS-FROM or IS-ABOUT relation.

S1 [The sale]$_{T_1}$ [infuriated]$_{O_1}$ [Beijing]$_{H_{1,2}}$ which [regards]$_{O_2}$ [Taiwan]$_{T_2}$ an integral part of its territory awaiting reunification, by force if necessary.

S2 "[Our agency]$_{T_1,H_2}$ [seriously needs]$_{O_2}$ [equipment for detecting drugs]$_{T_2}$," [he]$_{H_1}$ [said]$_{O_1}$.

In S1, for example, "infuriated" indicates that there is an (negative) opinion from "Beijing" regarding "the sale."[1]

Traditionally, the task of extracting opinion entities and opinion relations was handled in a pipelined manner, i.e., extracting the opinion expressions first and then extracting opinion targets and opinion holders based on their syntactic and semantic associations with the opinion expressions (Kim and Hovy, 2006; Kobayashi et al., 2007). More recently, methods that *jointly* infer the opinion entity and relation extraction tasks (e.g., using Integer Linear Programming (ILP)) have been introduced (Choi et al., 2006; Yang and Cardie, 2013) and show that the existence of opinion relations provides clues for the identification of opinion entities and vice-versa, and thus results in better performance than a pipelined approach. However, the success of these methods depends critically on the availability of opinion lexicons, dependency parsers, named-entity taggers, etc.

[1] This paper does not attempt to determine the sentiment, i.e., the positive or negative polarity, of an opinion.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 919–929,
Berlin, Germany, August 7-12, 2016.

Alternatively, neural network-based methods have been employed. In these approaches, the required latent features are automatically learned as dense vectors of the hidden layers. Liu et al. (2015), for example, compare several variations of recurrent neural network methods and find that long short-term memory networks (LSTMs) perform the best in identifying opinion expressions and opinion targets for the specific case of product/service reviews.

Motivated by the recent success of LSTMs on this and other problems in NLP, we investigate here the use of deep bi-directional LSTMs for joint extraction of opinion expressions, holders, targets and the relations that connect them. This is the first attempt to handle the full opinion entity and relation extraction task using a deep learning approach.

In experiments on the MPQA dataset for opinion entities (Wiebe and Cardie, 2005; Wilson, 2008), we find that standard LSTMs are not competitive with the state-of-the-art CRF+ILP joint inference approach of Yang and Cardie (2013), performing below even the standalone sequence-tagging CRF. Inspired by Huang et al. (2015), we show that incorporating sentence-level, and our newly proposed relation-level optimization, allows the LSTM to perform within 1–3% of the ILP joint model for all three opinion entity types and to do so without access to opinion lexicons, parsers or other preprocessing components.

For the primary task of identifying opinion entities together with their IS-FROM and IS-ABOUT relations, we show that the LSTM with sentence- and relation-level optimizations outperforms an LSTM baseline that does not employ joint inference. When compared to the CRF+ILP-based joint inference approach, the optimized LSTM performs slightly better for the IS-ABOUT[2] relation and within 3% for the IS-FROM relation.

In the sections that follow, we describe: related work (Section 2) and the multi-layer bi-directional LSTM (Section 3); the LSTM extensions (Section 4); the experiments on the MPQA corpus (Sections 5 and 6) and error analysis (Section 7).

[2]Target and IS-ABOUT relation identification is one important aspect of opinion analysis that hasn't been much addressed in previous work and has proven to be difficult for existing methods.

2 Related Work

LSTM-RNNs (Hochreiter and Schmidhuber, 1997) have recently been applied to many sequential modeling and prediction tasks, such as machine translation (Bahdanau et al., 2014; Sutskever et al., 2014), speech recognition (Graves et al., 2013), NER (Hammerton, 2003). The bi-directional variant of RNNs has been found to perform better as it incorporates information from both the past and the future (Schuster and Paliwal, 1997; Graves et al., 2013). Deep RNNs (stacked RNNs) (Schmidhuber, 1992; Hihi and Bengio, 1996) capture more abstract and higher-level representation in different layers and benefit sequence modeling tasks (İrsoy and Cardie, 2014). Collobert et al. (2011) found that adding dependencies between the tags in the output layer improves the performance of Semantic Role Labeling task. Later, Huang et al. (2015) also found that adding a CRF layer on top of bi-directional LSTMs to capture these dependencies can produce state-of-the-art performance on part-of-speech (POS), chunking and NER.

For fine-grained opinion extraction, earlier work (Wilson et al., 2005; Breck et al., 2007; Yang and Cardie, 2012) focused on extracting subjective phrases using a CRF-based approach from open-domain text such as news articles. Choi et al. (2005) extended the task to jointly extract opinion holders and these subjective expressions. Yang and Cardie (2013) proposed a ILP-based joint-inference model to jointly extract the opinion entities and opinion relations, which performed better than the pipelined based approaches (Kim and Hovy, 2006).

In the neural network domain, İrsoy and Cardie (2014) proposed a deep bi-directional recurrent neural network for identifying subjective expressions, outperforming the previous CRF-based models. Irsoy and Cardie (2013) additionally proposed a bi-directional recursive neural network over a binary parse tree to jointly identify opinion entities, but performed significantly worse than the feature-rich CRF+ILP approach of Yang and Cardie (2013). Liu et al. (2015) used several variants of recurrent neural networks for joint opinion expression and aspect/target identification on customer reviews for restaurants and laptops, outperforming the feature-rich CRF based baseline. In the product reviews domain, however, the opinion holder is generally the reviewer and the task

does not involve identification of relations between opinion entities. Hence, standard LSTMs are applicable in this domain. None of the above neural network based models can jointly model opinion entities and opinion relations.

In the relation extraction domain, several neural networks have been proposed for relation classification, such as RNN-based models (Socher et al., 2012) and LSTM-based models (Xu et al., 2015). These models depend on constituent or dependency tree structures for relation classification, and also do not model entities jointly. Recently, Miwa and Bansal (2016) proposed a model to jointly represent both entities and relations with shared parameters, but it is not a joint-inference framework.

3 Methodology

For our task, we propose the use of multi-layer bi-directional LSTMs, a type of recurrent neural network. Recurrent neural networks have recently been used for modeling sequential tasks. They are capable of modeling sequences of arbitrary length by repetitive application of a recurrent unit along the tokens in the sequence. However, recurrent neural networks are known to have several disadvantages like the problem of vanishing and exploding gradients. Because of these problems, it has been found that recurrent neural networks are not sufficient for modeling long term dependencies. Hochreiter and Schmidhuber (1997), thus proposed long short term memory (LSTMs), a variant of recurrent neural networks.

3.1 Long Short Term Memory (LSTM)

Long short term memory networks are capable of learning long-term dependencies. The recurrent unit is replaced by a memory block. The memory block contains two cell states – memory cell C_t and hidden state h_t; and three multiplicative gates – input gate i_t, forget gate f_t and output gate o_t. These gates regulate the addition or removal of information to the cell state thus overcoming vanishing and exploding gradients.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

The forget gate f_t and input gate i_t above decides what part of the information we are going to throw away from the cell state and what new information we are going to store in the cell state. The sigmoid outputs a number between 0 and 1 where 0 implies that the information is completely lost and 1 means that the information is completely retained.

$$\widetilde{C}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$C_t = i_t * \widetilde{C}_t + f_t * C_{t-1}$$

Thus, the intermediate cell state $\widetilde{C}_t$ and previous cell state C_{t-1} are used to update the new cell state C_t.

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Next, we update the hidden state h_t based on the output gate o_t and the cell state C_t. We pass both the cell state C_t and the hidden state h_t to the next time step.

3.2 Multi-layer Bi-directional LSTM

In sequence tagging problems, it has been found that only using past information for computing the hidden state h_t may not be sufficient. Hence, previous works (Graves et al., 2013; İrsoy and Cardie, 2014) proposed the use of bi-directional recurrent neural networks for speech and NLP tasks, respectively. The idea is to also process the sequence in the backward direction. Hence, we can compute the hidden state $\overrightarrow{h_t}$ in the forward direction and $\overleftarrow{h_t}$ in the backward direction for every token.

Also, in more traditional feed-forward networks, deep networks have been found to learn abstract and hierarchical representations of the input in different layers (Bengio, 2009). The multi-layer LSTMs have been proposed (Hermans and Schrauwen, 2013) to capture long-term dependencies of the input sequences in different layers.

For the first hidden layer, the computation proceeds similar to that described in Section 3.1. However, for higher hidden layers i the input to the memory block is the hidden state and memory cell from the previous layer $i-1$ instead of the input vector representation.

For this paper, we only use the hidden state from the last layer L to compute the output state y_t.

$$z_t = \overrightarrow{V}\overrightarrow{h_t}^{(L)} + \overleftarrow{V}\overleftarrow{h_t}^{(L)} + c$$
$$y_t = g(z_t)$$

4 Network Training

For our problem, we wish to predict a label y from a discrete set of classes Y for every word in a sentence. As is the norm, we train the network by

maximizing the log-likelihood

$$\sum_{(x,y)\in \mathrm{T}} \log p(y|x,\theta)$$

over the training data T, with respect to the parameters θ, where x is the input sentence and y is the corresponding tag sequence. We propose three alternatives for the log-likelihood computation.

4.1 Word-Level Log-Likelihood (WLL)

We first formulate a word-level log-likelihood (WLL) (adapted from Collobert et al. (2011)) that considers all words in a sentence independently. We interpret the score z_t corresponding to the i^{th} tag $[z_t]_i$ as a conditional tag probability $\log p(i|x,\theta)$ by applying a softmax operation.

$$\begin{aligned} p(i|x,\theta) &= softmax(z_t^i) \\ &= \frac{e^{z_t^i}}{\sum_j e^{z_t^j}} \end{aligned}$$

For the tag sequence y given the input sentence x the log-likelihood is :

$$\log p(y|x,\theta) = z^y - \operatorname*{logadd}_j z^j$$

4.2 Sentence-Level Log-Likelihood (SLL)

In the word-level approach above, we discard the dependencies between the tags in a tag sequence. In our sentence-level log-likelihood (SLL) formulation (also adapted from Collobert et al. (2011)) we incorporate these dependencies: we introduce a transition score $[A]_{i,j}$ for jumping from tag i to tag j of adjacent words in the tag sequence to the set of parameters $\widetilde{\theta}$. These transition scores are going to be trained.

We use both the transition scores $[A]$ and the output scores z to compute the sentence score $s(x|_{t=1}^T, y|_{t=1}^T, \widetilde{\theta})$.

$$s(x,y,\widetilde{\theta}) = \sum_{t=1}^{T} \left([A]_{y_{t-1},y_t} + z_t^{y_t} \right)$$

We normalize this sentence score over all possible paths of tag sequences $\widetilde{y}$ to get the log conditional probability as below :

$$\log p_{sent}(y|x,\widetilde{\theta}) = s(x,y,\widetilde{\theta}) - \operatorname*{logadd}_{\widetilde{y}} s(x,\widetilde{y},\widetilde{\theta})$$

Even though the number of tag sequences grows exponentially with the length of the sentence, we can compute the normalization factor in linear time (Collobert et al., 2011).

At inference time, we find the best tag sequence

$$\operatorname*{argmax}_{\widetilde{y}} s(x,\widetilde{y},\widetilde{\theta})$$

for an input sentence x using Viterbi decoding. In this case, we basically maximize the same likelihood as in a CRF except that a CRF is a linear model.

The above sentence-level log-likelihood is useful for sequential tagging, but it cannot be directly used for modeling relations between non-adjacent words in the sentence. In the next subsection, we extend the above idea to also model relations between non-adjacent words.

4.3 Relation-Level Log-Likelihood (RLL)

For every word x_t in the sentence x, we output the tag y_t and a distance d_t. If a word at position t is related to a word at position k and $k < t$, then $d_t = (t-k)$. If word t is not related to any other word to its left, then $d_t = 0$. Let D_{Left} be the maximum distance we model for such *left*-relations [3].

$$z_t = \overrightarrow{V_r}\overrightarrow{h_t}^{(L)} + \overleftarrow{V_r}\overleftarrow{h_t}^{(L)} + c_r$$

We let $\overrightarrow{V_r} \in \mathbb{R}^{(D_{Left}+1)\times Y \times d_h}$ (where d_h is the dimensionality of hidden units) such that the output state $z_t \in \mathbb{R}^{(D_{Left}+1)\times Y}$ as compared to $z_t \in \mathbb{R}^{(1)\times Y}$ in case of sentence-level log-likelihood.

In order to add dependencies between tags and relations, we introduce a transition score $[A]_{i,j,d',d"}$ for jumping from tag i and relation distance $d^{'}$ to tag j and relation distance $d^{"}$ of adjacent words in the tag sequence, to the set of parameters $\theta^{'}$. These transition scores are also going to be trained similar to the transition scores in sentence-level log-likelihood.

The sentence score $s(x|_{t=1}^T, y|_{t=1}^T, d|_{t=1}^T, \theta^{'})$ is:

$$s(x,y,d,\theta^{'}) = \sum_{t=1}^{T} \left([A]_{y_{t-1},y_t,d_{t-1},d_t} + z_t^{y_t,d_t} \right)$$

We normalize this sentence score over all possible paths of tag $\widetilde{y}$ and relation sequences $\widetilde{d}$ to get the log conditional probability as below :

$$\begin{aligned} \log p_{rel,Left}(y,d|x,\widetilde{\theta}) =& s(x,y,d,\theta^{'}) \\ & - \operatorname*{logadd}_{\widetilde{y},\widetilde{d}} s(x,\widetilde{y},\widetilde{d},\theta^{'}) \end{aligned}$$

[3] Later in this section, we will also add a similar likelihood in the objective function for *right*-relations, i.e., for each word the related words are in its right context.

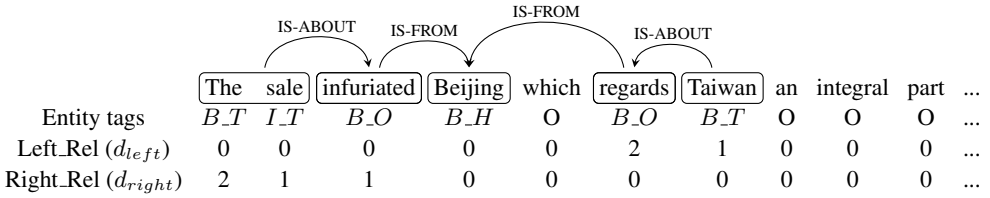


Figure 1: Gold standard annotation for an example sentence from MPQA dataset. O represents the 'Other' tag in the BIO scheme.

We can still compute the normalization factor in linear time similar to sentence-level log-likelihood.

At inference time, we jointly find the best tag and relation sequence

$$\underset{\widetilde{y},\widetilde{d}}{\operatorname{argmax}}\, s(x, \widetilde{y}, \widetilde{d}, \theta^{'})$$

for an input sentence x using Viterbi decoding.

For our task of joint extraction of opinion entities and relations, we train our model to predict tag y and relation distance d for every word in the sentence by maximizing the log-likelihood (**SLL+RLL**) below using Adadelta (Zeiler, 2012).

$$\sum_{(x,y)\in \mathrm{T}} \log p_{sent}(y|x,\theta^{'}) + \log p_{rel,Left}(y,d|x,\theta^{'}) + \log p_{rel,Right}(y,d|x,\theta^{'})$$

5 Experiments

5.1 Data

We use the MPQA 2.0 corpus (Wiebe and Cardie, 2005; Wilson, 2008). It contains news articles and editorials from a wide variety of news sources. There are a total of 482 documents in our dataset containing 9471 sentences with phrase-level annotations. We set aside 132 documents as a development set and use the remaining 350 documents as the evaluation set. We report the results using 10-fold cross validation at the document level to mimic the methodology of Yang and Cardie (2013).

The dataset contains gold-standard annotations for opinion entities — expressions, targets, holders. We use only the direct subjective/opinion expressions. There are also annotations for opinion relations – IS-FROM between opinion holders and opinion expressions; and IS-ABOUT between opinion targets and opinion expressions. These relations can overlap but we discard all relations that contain sub-relations similar to Yang and Cardie (2013). We also leave identification of overlapping relations for future work.

Figure 1 gives an example of an annotated sentence from the dataset: boxes denote opinion entities and opinion relations are shown by arcs. We interpret these relations arcs as directed — from an opinion expression towards an opinion holder, and from an opinion target towards an opinion expression.

In order to use the RLL formulation as defined in Section 4.3, we pre-process these relation arcs to obtain the left-relation distances (d_{left}) and right-relation distances (d_{right}) as shown in Figure 1. For each word in an entity, we find its distance to the nearest word in the related entity. These distances become our relation tags. The entity tags are interpreted using the BIO scheme, also shown in the figure. Our RLL model jointly models the entity tags and relation tags. At inference time, these entity tags and relation tags are used together to determine IS-FROM and IS-ABOUT relations. We use a simple majority vote to determine the final entity tag from SLL+RLL model.

5.2 Evaluation Metrics

We use precision, recall and F-measure (as in Yang and Cardie (2013)) as evaluation metrics. Since the identification of exact boundaries for opinion entities is hard even for humans (Wiebe and Cardie, 2005), soft evaluation methods such as Binary Overlap and Proportional Overlap are reported. Binary Overlap counts every overlapping predicted and gold entity as correct, while Proportional Overlap assigns a partial score proportional to the ratio of overlap span and the correct span (Recall) or the ratio of overlap span and the predicted span (Precision).

For the case of opinion relations, we report precision, recall and F-measure according to the Binary Overlap. It considers a relation correct if there is an overlap between the predicted opin-

	Opinion Expression			Opinion Target			Opinion Holder		
Method	P	R	F1	P	R	F1	P	R	F1
CRF	**84.42**$^{\mathbf{3.24}}$	61.61$^{3.20}$	*71.17*$^{2.66}$	**80.38**$^{\mathbf{2.72}}$	46.80$^{4.41}$	*59.10*$^{4.06}$	**73.37**$^{\mathbf{4.09}}$	49.71$^{3.46}$	59.21$^{3.49}$
CRF+ILP	73.53$^{3.90}$	**74.89**$^{\mathbf{2.51}}$	**74.11**$^{\mathbf{2.49}}$	*77.27*$^{3.49}$	56.94$^{3.94}$	**65.40**$^{\mathbf{3.07}}$	*67.00*$^{3.17}$	**67.22**$^{\mathbf{3.50}}$	**67.22**$^{\mathbf{2.54}}$
LSTM+WLL	67.88$^{4.49}$	66.13$^{3.20}$	66.87$^{2.66}$	58.71$^{4.87}$	54.92$^{3.23}$	56.50$^{1.51}$	60.33$^{4.54}$	*63.34*$^{2.33}$	61.65$^{2.37}$
LSTM+SLL	70.45$^{5.12}$	66.65$^{3.46}$	68.37$^{3.14}$	63.02$^{4.61}$	56.77$^{3.98}$	*59.65*$^{3.61}$	61.85$^{3.82}$	*63.12*$^{3.59}$	*62.35*$^{2.46}$
LSTM+SLL+RLL	71.73$^{5.35}$	*70.92*$^{3.96}$	*71.11*$^{2.71}$	64.52$^{5.52}$	**65.94**$^{\mathbf{4.74}}$	*64.84*$^{1.44}$	62.75$^{3.75}$	*67.17*$^{4.37}$	*64.71*$^{2.23}$
CRF	**80.78**$^{\mathbf{3.27}}$	57.62$^{3.24}$	*67.19*$^{2.63}$	**71.81**$^{\mathbf{3.22}}$	42.36$^{3.78}$	*53.23*$^{3.69}$	**71.56**$^{\mathbf{3.54}}$	48.61$^{3.51}$	57.86$^{3.43}$
CRF+ILP	71.03$^{4.03}$	**69.72**$^{\mathbf{2.37}}$	**70.22**$^{\mathbf{2.44}}$	**71.94**$^{\mathbf{3.25}}$	49.83$^{3.24}$	**58.72**$^{\mathbf{2.80}}$	*65.70*$^{3.07}$	**65.91**$^{\mathbf{3.63}}$	**65.68**$^{\mathbf{2.61}}$
LSTM+WLL	64.47$^{4.79}$	59.45$^{3.52}$	61.67$^{2.26}$	52.72$^{5.01}$	44.21$^{2.54}$	47.85$^{1.41}$	58.41$^{4.72}$	59.72$^{2.52}$	52.45$^{2.23}$
LSTM+SLL	65.97$^{5.46}$	61.76$^{3.69}$	63.60$^{3.05}$	54.46$^{4.49}$	50.16$^{4.38}$	52.01$^{3.05}$	59.80$^{3.29}$	*61.27*$^{3.75}$	60.40$^{2.26}$
LSTM+SLL+RLL	65.48$^{4.92}$	*65.54*$^{3.65}$	*65.56*$^{2.71}$	52.75$^{6.81}$	**60.54**$^{\mathbf{4.78}}$	*55.81*$^{1.96}$	59.44$^{3.56}$	*65.51*$^{4.22}$	*62.18*$^{2.50}$

Table 1: Performance on opinion entity extraction. Top table shows Binary Overlap performance; bottom table shows Proportional Overlap performance. Superscripts designate one standard deviation.

ion expression and the gold opinion expression as well as an overlap between the predicted entity (holder/target) and the gold entity (holder/target).

5.3 Baselines

CRF+ILP. We use the ILP-based joint inference model (Yang and Cardie, 2013) as baseline for both the entity and relation extraction tasks. It represents the state-of-the-art for fine-grained opinion extraction. Their method first identifies opinion entities using **CRF**s (an additional baseline) with a variety of features such as words, POS tags, and lexicon features (the subjectivity strength of the word in the Subjectivity Lexicon). They also train a relation classifier (logistic regression) by over-generating candidates from the CRFs (50-best paths) using local features such as word, POS tags, subjectivity lexicons as well as semantic and syntactic features such as semantic frames, dependency paths, WordNet hypernyms, etc. Finally, they use ILP for joint-inference to find the optimal prediction for both opinion entity and opinion relation extraction.

LSTM+SLL+Softmax. As an additional baseline for relation extraction, we train a softmax classifier on top of our SLL framework. We jointly learn the relation classifier and SLL model. For every entity pair $[x]_i^j$, $[x]_k^l$, we first sum the start and end word output representation $[z_t]$ and then concatenate them to learn softmax weight W' where $W' \in \mathbb{R}^{3 \times 2d_h}$.

$$y_{rel} = softmax(W' \begin{bmatrix} [z_t]_i + [z_t]_j \\ [z_t]_k + [z_t]_l \end{bmatrix})$$

The inference is pipelined in this case. At the time of inference, we first predict the entity spans and then use these spans for relation classification.

5.4 Hyperparameter and Training Details

We use multi-layer bi-directional LSTMs for all the experiments such that the number of hidden layers is 3 and the dimensionality of hidden units (d_h) is 50. We use Adadelta for training. We initialize our word representation using publicly available word2vec (Mikolov et al., 2013) trained on Google News dataset and keep them fixed during training. For RLL, we keep D_{Left} and D_{Right} as 15. All the weights in the network are initialized from small random uniform noise. We train all our models for 200 epochs. We do not pretrain our network. We regularize our network using dropout (Srivastava et al., 2014) with the dropout rate tuned using the development set. We select the final model based on development-set performance (average of Proportional Overlap for entities and Binary Overlap for relations).

6 Results

6.1 Opinion Entities

Table 1 shows the performance of opinion entity identification using the Binary Overlap and Proportional Overlap evaluation metrics. We discuss specific results in the paragraphs below.

WLL vs. SLL. SLL performs better than WLL on all entity types, particularly with respect to Proportional Overlap on opinion holder and target entities. A similar trend can be seen for the example sentences in Table 3. In S1, SLL extracts "has been in doubt" as the opinion expression whereas WLL only identifies "has". Similarly in S2, WLL annotates "Saudi Arabia's request on a case-by-case" as the target while SLL correctly includes "basis" in its annotation. Thus, we find that modeling the transitions between adjacent tags enables

	IS-ABOUT			IS-FROM		
Method	P	R	F1	P	R	F1
CRF+ILP	*61.57*$^{4.56}$	*47.65*$^{3.12}$	*54.39*$^{2.49}$	*64.04*$^{3.08}$	**58.79**$^{\mathbf{4.42}}$	**61.17**$^{\mathbf{3.02}}$
LSTM+SLL+Softmax	36.23$^{5.10}$	36.12$^{7.75}$	35.40$^{3.35}$	36.44$^{5.26}$	40.19$^{6.13}$	37.60$^{3.42}$
LSTM+SLL+RLL	**62.48**$^{\mathbf{3.87}}$	**49.80**$^{\mathbf{2.84}}$	**54.98**$^{\mathbf{2.54}}$	**64.19**$^{\mathbf{3.81}}$	*53.75*$^{6.00}$	*58.22*$^{3.01}$

Table 2: Performance on opinion relation extraction using Binary Overlap on the opinion entities. Superscripts designate one standard deviation.

SLL to find entire opinion entity phrases better than WLL, leading to better Proportional Overlap scores.

SLL vs. SLL+RLL. From Table 1, we see that the joint-extraction model (SLL+RLL) performs better than SLL as expected. More specifically, SLL+RLL model has better recall for all opinion entity types. The example sentences from Table 3 corroborate these results. In S1, SLL+RLL identifies "announced" as an opinion expression, which was missing in both WLL and SLL. In S3, neither the WLL nor the SLL model can annotate opinion holder (H_1) or the target (T_1), but SLL+RLL correctly identifies the opinion entities because of modeling the relations between the opinion expression "will decide" and the holder/target entities.

CRF vs. LSTM-based Models. From the analysis of the performance in Table 1, we find that our WLL and SLL models perform worse while our best SLL+RLL model can only match the performance of the CRF baseline on opinion expressions. Even though the recall of all our LSTM-based models is higher than the recall of the CRF-baseline for opinion expressions, we cannot match the precision of CRF baseline. We suspect that the reason for such high precision on the part of the CRF is its access to carefully prepared subjectivity-lexicons[4]. Our LSTM-based models do not rely on such features except via the word-vectors. With respect to holders and targets, we find that our SLL model performs similar to the CRF baseline. However, the SLL+RLL model outperforms CRF baseline.

CRF+ILP vs. SLL+RLL. Even though we find that our LSTM-based joint-model (SLL+RLL) outperforms our LSTM-based only-entity extraction model (SLL), the performance is still below the ILP-based joint-model (CRF+ILP). However, we perform comparably with respect to target entities (Binary Overlap). Also, our recall on targets is much better than all other models whereas the recall on holders is very similar to CRF+ILP. Our SLL+RLL model can identify targets such as "Australia's involvement in Kyoto" which the ILP-based model cannot, as observed for S1 in Table 3. In S3, the ILP-based model also erroneously divides the target "consider Saudi Arabia's request on a case-by-case basis" into a holder "Saudi Arabia's" and opinion expression "request", while SLL+RLL model can correctly identify it. We will compare the two models in detail in Section 7.

[4]http://mpqa.cs.pitt.edu/lexicons/ subj_lexicon/

6.2 Opinion Relations

The extraction of opinion relations is our primary task. Table 2[5] shows the performance on opinion relation extraction task using Binary Overlap.

SLL+Softmax vs. SLL+RLL. The opinion entities and relations are jointly modeled in both the models, but we see a significant improvement in performance by adding relation level dependencies to the model vs. learning a classifier on top of sentence-level dependencies to learn the relation between entities. LSTM+SLL+RLL performs much better in terms of both precision and recall on both IS-FROM and IS-ABOUT relations.

CRF+ILP vs. SLL+RLL. We find that our SLL+RLL model performs comparably and even slightly better on IS-ABOUT relations. Such performance is encouraging because our LSTM-based model does not rely on features such as dependency paths, semantic frames or subjectivity lexicons for our model. Our sequential LSTM model is able to learn these relations thus validating that LSTMs can model long-term dependencies. However, for IS-FROM relations, we find that our recall is lower than the ILP-based joint model.

[5]Yang and Cardie (2013) omitted a subset of targets and IS-ABOUT relations. We fixed this and re-ran their models on the updated dataset, obtaining the lower F-score 54.39 for IS-ABOUT relations.

S1 :	[Australia's involvement in Kyoto]$_{\mathbf{T_1}}$ [has been in doubt]$_{\mathbf{O_1}}$ ever since [the US President, George Bush]$_{\mathbf{H_2}}$, [announced]$_{\mathbf{O_2}}$ last year that [ratifying the protocol]$_{\mathbf{T_2}}$ would hurt the US economy.
CRF+ILP	Australia's involvement in Kyoto [has been in doubt]$_{\mathbf{O_1}}$ ever since the US President, George Bush, announced last year that [ratifying the protocol]$_{\mathbf{T_1}}$ would hurt the US economy.
WLL	[Australia's involvement in Kyoto]$_{\mathbf{T}}$ [has]$_{\mathbf{O}}$ been in doubt ever since the US [President]$_{\mathbf{H}}$, [George Bush]$_{\mathbf{H}}$, announced last year that ratifying the protocol would hurt the US economy.
SLL	[Australia's involvement in Kyoto]$_{\mathbf{T}}$ [has been in doubt]$_{\mathbf{O}}$ ever since the US President, George Bush, announced last year that ratifying the protocol would hurt the US economy.
SLL+RLL	[Australia's involvement in Kyoto]$_{\mathbf{T}}$ [has been in doubt]$_{\mathbf{O}}$ ever since the US President, [George Bush]$_{\mathbf{H_2}}$, [announced]$_{\mathbf{O_2}}$ last year that [ratifying the protocol]$_{\mathbf{T_2}}$ would hurt the US economy.
S2 :	Bush said last week [he]$_{\mathbf{H_{1,2}}}$ [was willing]$_{\mathbf{O_1}}$ [to consider]$_{\mathbf{O_2}}$ [Saudi Arabia's request on a case-by-case basis]$_{\mathbf{T_2}}$ but [U.S. officials]$_{\mathbf{H_3}}$ [doubted]$_{\mathbf{O_3}}$ [it would happen any time soon]$_{\mathbf{T_3}}$.
CRF+ILP	[Bush]$_{\mathbf{H_1}}$ [said]$_{\mathbf{O_1}}$ last week [he]$_{\mathbf{H_2}}$ [was willing to consider]$_{\mathbf{O_2}}$ [Saudi Arabia's]$_{\mathbf{H_3}}$ [request]$_{\mathbf{O_3}}$ on a case-by-case basis but [U.S. officials]$_{\mathbf{H_4}}$ [doubted]$_{\mathbf{O_4}}$ [it]$_{\mathbf{T_4}}$ would happen any time soon.
WLL	Bush said last week [he]$_{\mathbf{H}}$ [was willing]$_{\mathbf{O}}$ to [consider]$_{\mathbf{O}}$ [Saudi Arabia's request on a case-by-case]$_{\mathbf{T}}$ basis but [U.S. officials]$_{\mathbf{H}}$ [doubted]$_{\mathbf{O}}$ [it]$_{\mathbf{T}}$ would [happen any time soon]$_{\mathbf{T}}$.
SLL	Bush said last week [he]$_{\mathbf{H}}$ [was willing]$_{\mathbf{O}}$ to [consider Saudi Arabia's request on a case-by-case basis]$_{\mathbf{T}}$ but [U.S. officials]$_{\mathbf{H}}$ [doubted]$_{\mathbf{O}}$ [it]$_{\mathbf{T}}$ would happen any time soon.
SLL+RLL	Bush said last week [he]$_{\mathbf{H_1}}$ [was willing to consider]$_{\mathbf{O_1}}$ [Saudi Arabia's request on a case-by-case basis]$_{\mathbf{T_1}}$ but [U.S. officials]$_{\mathbf{H_2}}$ [doubted]$_{\mathbf{O_2}}$ [it would happen any time soon]$_{\mathbf{T_2}}$.
S3 :	Hence, [the Organization of Petroleum Exporting Countries (OPEC)]$_{\mathbf{H_1}}$, [will decide]$_{\mathbf{O_1}}$ at its meeting on Wednesday [whether or not to cut its worldwide crude production in an effort to shore up energy prices]$_{\mathbf{T_1}}$.
CRF+ILP	Hence, the Organization of Petroleum Exporting Countries (OPEC), [will decide]$_{\mathbf{O_1}}$ at its meeting on Wednesday whether [or not to cut its worldwide crude production in an effort to shore up energy prices]$_{\mathbf{T_1}}$.
WLL	Hence, the Organization of Petroleum Exporting Countries (OPEC), will [decide]$_{\mathbf{O}}$ at its meeting on Wednesday whether or not to cut its worldwide crude production in an effort to shore up energy prices.
SLL	Hence, the Organization of Petroleum Exporting Countries (OPEC), [will decide]$_{\mathbf{O}}$ at its meeting on Wednesday whether or not to cut its worldwide crude production in an effort to shore up energy prices.
SLL+RLL	Hence, [the Organization of Petroleum Exporting Countries (OPEC)]$_{\mathbf{H_1}}$, [will decide]$_{\mathbf{O_1}}$ at its meeting on Wednesday whether [or not to cut its worldwide crude production in an effort to shore up energy prices]$_{\mathbf{T_1}}$.

Table 3: Output from different models. The first row for each example is the gold standard.

7 Discussion

In this section, we discuss the various advantages and disadvantages of the LSTM-based SLL+RLL model as compared to the joint-inference (CRF+ILP) model. We provide examples from the dataset in Table 4.

From Table 2, we find that SLL+RLL model performs worse with respect to the opinion expression entities and opinion holder entities. On careful analysis of the output, we found cases such as S1 in Table 4. For such sentences SLL+RLL model prefers to annotate the opinion target (T_3) "US requests for more oil exports", whereas the ILP model annotates the embedded opinion holder (H_4) "US" and opinion expression (T_4) "requests". Both models are valid with respect to the gold-standard. In order to simplify our problem, we discard these embedded relations during training similar to Yang and Cardie (2013). However, for future work we would like to model these overlapping relations which could potentially improve our performance on opinion holders and opinion expressions.

We also found several cases such as S2, where the SLL+RLL model fails to annotate "said" as an opinion expression. The gold standard opinion expressions include speech events like "said" or "a statement", but not all occurrences of these speech events are opinion expressions, some are merely objective events. In S2, "was martyred" is an indication of an opinion being expressed, so "said" is annotated as an opinion expression. From our observation, the ILP model is more relaxed in annotating most of these speech events as opinion expressions and thus likely to identify corresponding

S1 :	However, [Chavez]$_{\mathbf{T_1}}$ who [is known for]$_{\mathbf{O_1}}$ [his]$_{\mathbf{H_2}}$ [ala Fidel Castro left-leaning anti-American philosophy]$_{O_2}$ had on a number of occasions [rebuffed]$_{O_3}$ [[US]$_{\mathbf{H_4}}$ [requests]$_{\mathbf{O_4}}$ for [more oil exports]$_{\mathbf{T_4}}$]$_{\mathbf{T_3}}$.
CRF+ILP	However, [Chavez]$_{\mathbf{H_1}}$ who [is known]$_{\mathbf{O}}$ for [his ala Fidel Castro]$_{\mathbf{H_2}}$ [left-leaning anti-American philosophy]$_{\mathbf{O_2}}$ had on a number of occasions [rebuffed]$_{\mathbf{O_1}}$ [US]$_{\mathbf{H_3}}$ [requests]$_{\mathbf{O_3}}$ for more oil exports.
SLL+RLL	However, Chavez who [is known]$_{\mathbf{O}}$ for his ala Fidel Castro left-leaning anti-American [philosophy]$_{\mathbf{O}}$ had on a number of occasions [rebuffed]$_{\mathbf{O_1}}$ [US requests for more oil exports]$_{\mathbf{T_1}}$.
S2 :	A short while ago, [our correspondent in Bethlehem]$_{\mathbf{H_1}}$ [said]$_{\mathbf{O_1}}$ that [Ra'fat al-Bajjali]$_{\mathbf{T_1}}$ **was martyred** of wounds sustained in the explosion.
CRF+ILP	A short while ago, [our correspondent]$_{\mathbf{H_1}}$ in Bethlehem [said]$_{\mathbf{O_1}}$ that [Ra'fat al-Bajjali]$_{T_1}$ was martyred of wounds sustained in the explosion.
SLL+RLL	A short while ago, our correspondent in Bethlehem said that Ra'fat al-Bajjali was martyred of wounds sustained in the explosion.
S3 :	This is no criticism, and is widely known and appreciated.
CRF+ILP	This is no criticism, and is widely known and appreciated.
SLL+RLL	[This]$_{\mathbf{T_1}}$ [is no criticism]$_{\mathbf{O_1}}$, and is widely [known and appreciated]$_{\mathbf{O}}$.
S4 :	From the fact that mothers care for their young, we can not deduce that they ought to do so, Hume argued.
CRF+ILP	From the fact that [mothers]$_{\mathbf{H_1}}$ [care]$_{\mathbf{O_1}}$ for their young, we can not deduce that they ought to do so, [Hume]$_{\mathbf{H_2}}$ [argued]$_{\mathbf{O_2}}$.
SLL+RLL	From the fact that mothers care for their young, [we]$_{\mathbf{H_1}}$ [can not deduce]$_{\mathbf{O_1}}$ that [they]$_{\mathbf{T_1}}$ ought to do so, [Hume]$_{\mathbf{H_2}}$ [argued]$_{\mathbf{O_2}}$.

Table 4: Examples from the dataset with label annotations from CRF+ILP and SLL+RLL models for comparison. The first row for each example is the gold standard.

opinion holders and opinion targets as compared to SLL+RLL model.

There were also instances such as S3 and S4 in Table 4 for which the gold standard does not have an annotation but the SLL+RLL output looks reasonable with respect to our task. In S3, SLL+RLL identifies "is no criticism" as an opinion expression for the target "This". However, it fails to identify the relation-link between "known and appreciated" and the target "This". Similarly, SLL+RLL also identifies reasonable opinion entities in S4, whereas the ILP model erroneously annotates "mothers" as the opinion holder and "care" as the opinion expression.

We handle the task of joint-extraction of opinion entities and opinion relations as a sequence labeling task in this paper and report the performance of the 1-best path at the time of Viterbi inference. However, there are approaches such as discriminative reranking (Collins and Koo, 2005) to rerank the output of an existing system that offer a means for further improving the performance of our SLL+RLL model. In particular, the oracle performance using the top-10 Viterbi paths from our SLL+RLL model has an F-score of 82.11 for opinion expressions, 76.77 for targets and 78.10 for holders. Similarly, IS-ABOUT relations have an F-score of 65.99 and IS-FROM relations, an F-score of 70.80. These scores are on average 10 points better than the performance of the current SLL+RLL model, indicating that substantial gains might be attained via reranking.

8 Conclusion

In this paper, we explored LSTM-based models for the joint extraction of opinion entities and relations. Experimentally, we found that adding sentence-level and relation-level dependencies on the output layer improves the performance on opinion entity extraction, obtaining results within 1-3% of the ILP-based joint model on opinion entities, within 3% for IS-FROM relation and comparable for IS-ABOUT relation.

In future work, we plan to explore the effects of pre-training (Bengio et al., 2009) and scheduled sampling (Bengio et al., 2015) for training our LSTM network. We would also like to explore re-ranking methods for our problem. With respect to the fine-grained opinion mining task, a potential future direction to be able to model overlapping and embedded entities and relations and also to extend this model to handle cross-sentential relations.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *CoRR*, abs/1506.03099.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2683–2688, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. 2005. Identifying sources of opinions with conditional random fields and extraction patterns. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 355–362, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yejin Choi, Eric Breck, and Claire Cardie. 2006. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 431–439, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Comput. Linguist.*, 31(1):25–70, March.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 273–278.

James Hammerton. 2003. Named entity recognition with long short-term memory. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 172–175, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 190–198.

Salah El Hihi and Yoshua Bengio. 1996. Hierarchical recurrent neural networks for long-term dependencies.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Ozan Irsoy and Claire Cardie. 2013. Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*.

Ozan İrsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 720–728.

Soo-Min Kim and Eduard Hovy. 2006. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text*, SST '06, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nozomi Kobayashi, Kentaro Inui, and Yuji Matsumoto. 2007. Extracting aspect-evaluation and aspect-of relations in opinion mining. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL.*

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443, Lisbon, Portugal, September. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *CoRR*, abs/1601.00770.

Jürgen Schmidhuber. 1992. Learning complex, extended sequences using the principle of history compression. *Neural Comput.*, 4(2):234–242, March.

M. Schuster and K.K. Paliwal. 1997. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, November.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1201–1211, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Janyce Wiebe and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. language resources and evaluation. In *Language Resources and Evaluation (formerly Computers and the Humanities*, page 2005.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.

Theresa Ann Wilson. 2008. *Fine-grained Subjectivity and Sentiment Analysis: Recognizing the intensity, polarity, and attitudes of private states*. Ph.D. thesis, The University of Pittsburgh, June.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *In Proceedings of Conference on Empirical Methods in Natural Language Processing*.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1335–1345, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August. Association for Computational Linguistics.

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

Transition-Based Left-Corner Parsing for Identifying PTB-Style Nonlocal Dependencies

Yoshihide Kato[1] and **Shigeki Matsubara**[2]
[1]Information & Communications, Nagoya University
[2]Graduate School of Information Science, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, 464-8601 Japan
yoshihide@icts.nagoya-u.ac.jp

Abstract

This paper proposes a left-corner parser which can identify nonlocal dependencies. Our parser integrates nonlocal dependency identification into a transition-based system. We use a structured perceptron which enables our parser to utilize global features captured by nonlocal dependencies. An experimental result demonstrates that our parser achieves a good balance between constituent parsing and nonlocal dependency identification.

1 Introduction

Many constituent parsers based on the Penn Treebank (Marcus et al., 1993) are available, but most of them do not deal with nonlocal dependencies. Nonlocal dependencies represent syntactic phenomenon such as wh-movement, A-movement in passives, topicalization, raising, control, right node raising and so on. Nonlocal dependencies play an important role on semantic interpretation. In the Penn Treebank, a nonlocal dependency is represented as a pair of an empty element and a filler.

Several methods of identifying nonlocal dependencies have been proposed so far. These methods can be divided into three approaches: pre-processing approach (Dienes and Dubey, 2003b), in-processing approach (Dienes and Dubey, 2003a; Schmid, 2006; Cai et al., 2011; Kato and Matsubara, 2015) and post-processing approach (Johnson, 2002; Levy and Manning, 2004; Campbell, 2004; Xue and Yang, 2013; Xiang et al., 2013; Takeno et al., 2015).[1] In pre-processing approach, a tagger called "trace tagger" detects empty elements. The trace tagger uses only surface word information. In-processing approach integrates nonlocal dependency identification into a parser. The parser uses a probabilistic context-free grammar to rank candidate parse trees. Post-processing approach recovers nonlocal dependencies from a parser output which does not include nonlocal dependencies.

The parsing models of the previous methods cannot use global features captured by nonlocal dependencies. Pre- or in-processing approach uses a probabilistic context-free grammar, which makes it difficult to use global features. Post-processing approach performs constituent parsing and nonlocal dependency identification separately. This means that the constituent parser cannot use any kind of information about nonlocal dependencies.

This paper proposes a parser which integrates nonlocal dependency identification into constituent parsing. Our method adopts an in-processing approach, but does not use a probabilistic context-free grammar. Our parser is based on a transition system with structured perceptron (Collins, 2002), which can easily introduce global features to its parsing model. We adopt a left-corner strategy in order to use the syntactic relation *c-command*, which plays an important role on nonlocal dependency identification. Previous work on transition-based constituent parsing adopts a shift-reduce strategy with a tree binarization (Sagae and Lavie, 2005; Sagae and Lavie, 2006; Zhang and Clark, 2009; Zhu et al., 2013; Wang and Xue, 2014; Mi and Huang, 2015; Thang et al., 2015; Watanabe and Sumita, 2015), or convert constituent trees to "spinal trees", which are similar to dependency trees (Ballesteros and Carreras, 2015). These conversions make it difficult for their parsers to capture c-command relations in the parsing process. On the other hand, our parser does not require such kind of conversion.

[1]The methods of (Cai et al., 2011; Xue and Yang, 2013; Xiang et al., 2013; Takeno et al., 2015) only detect empty elements.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 930–940,
Berlin, Germany, August 7-12, 2016.

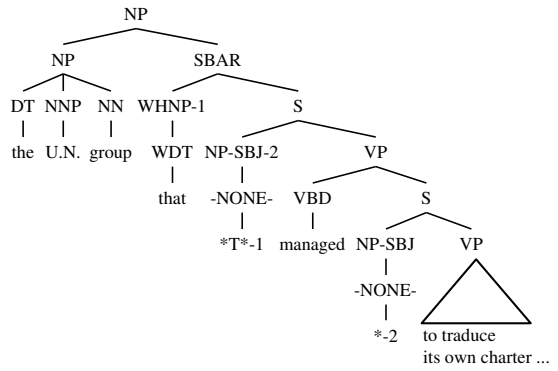


Figure 1: A parse tree in the Penn Treebank.

Our contribution can be summarized as follows:

1. We introduce empty element detection into transition-based left-corner constituent parsing.
2. We extend c-command relation to deal with nodes in *parse tree stack* in the transition system, and develop heuristic rules which coindex empty elements with their fillers on the basis of the extended version of c-command.
3. We introduce new features about nonlocal dependency to our parsing model.

This paper is organized as follows: Section 2 explains how to represent nonlocal dependencies in the Penn Treebank. Section 3 describes our transition-based left-corner parser. Section 4 introduces nonlocal dependency identification into our parser. Section 5 describes structured perceptron and features. Section 6 reports an experimental result, which demonstrated that our parser achieved a good balance between constituent parsing and nonlocal dependency identification. Section 7 concludes this paper.

2 Nonlocal Dependency

This section describes nonlocal dependencies in the Penn Treebank (Marcus et al., 1993). A nonlocal dependency is represented as a pair of an *empty element* and a *filler*. Figure 1 shows an example of (partial) parse tree in the Penn Treebank. The parse tree includes several nonlocal dependencies. The nodes labeled with -NONE- are empty elements. The terminal symbols such as $*$ and $*$T$*$ represent the type of nonlocal dependency: $*$ represents an unexpressed subject of to-infinitive. $*$T$*$ represents a trace of wh-movement. When a terminal symbol of empty element is indexed, its filler exists in the parse tree. The filler has the same number. For example, $*$T$*$-1 means that the node WHNP-1 is the corresponding filler. Table 1 gives a brief description of empty elements quoted from the annotation guideline (Bies et al., 1995). For more details, see the guideline.

3 Transition-Based Left-Corner Parsing

This section describes our transition-based left-corner parser.

As with previous work (Sagae and Lavie, 2005; Sagae and Lavie, 2006; Zhang and Clark, 2009; Zhu et al., 2013; Wang and Xue, 2014; Mi and Huang, 2015; Thang et al., 2015; Watanabe and Sumita, 2015), our transition-based parsing system consists of a set of parser states and a finite set of transition actions, each of which maps a state into a new one. A parser state consists of a stack of parse tree nodes and a buffer of input words. A state is represented as a tuple (σ, i), where σ is the stack and i is the next input word position in the buffer. The initial state is $(\langle\rangle, 0)$. The final states are in the form of $(\langle[\cdots]_{\texttt{TOP}}\rangle, n)$, where TOP is a special symbol for the root of the parse tree and n is the length of the input sentence. The transition actions for our parser are as follows:

- SHIFT(X): pop up the first word from the buffer, assign a POS tag X to the word and push it onto the stack.

The SHIFT action assigns a POS tag to the shifted word to perform POS tagging and constituent parsing simultaneously. This is in the same way as Wang and Xue (2014).

- LEFTCORNER-$\{$H/$\emptyset\}(X)$: pop up the first node from the stack, attach a new node labeled with X to the node as the parent and push it back onto the stack. H and $\emptyset$ indicate whether or not the popped node is the head child of the new node.
- ATTACH-$\{$H/$\emptyset\}$: pop up the top two nodes from the stack, attach the first one to the second one as the rightmost child and push it back onto the stack. H and $\emptyset$ indicate whether or not the first node is the head child of the second one.

We introduce new actions LEFTCORNER and ATTACH. ATTACH action is similar to REDUCE action standardly used in the previous transition-based parsers. However, there is an important

type	description	n-posi
*	arbitrary PRO, controlled PRO and trace of A-movement	L, R, –
EXP	expletive (extraposition)	R
ICH	interpret constituent here (discontinuous dependency)	L, R
RNR	right node raising	R
T	trace of A′-movement	A, L
0	null complementizer	–
U	unit	–
?	placeholder for ellipsed material	–
NOT	anti-placeholder in template gapping	–

Table 1: Empty elements in the Penn Treebank.

SHIFT(X)	$(\langle s_m, \ldots, s_0 \rangle, i) \Rightarrow (\langle s_m, \ldots, s_0, [w_i]_X \rangle, i+1)$
LEFTCORNER-{H/∅}(X)	$(\langle s_m, \ldots, s_1, s_0 \rangle, i) \Rightarrow (\langle s_m, \ldots, s_1, [s_0]_X \rangle, i)$
ATTACH-{H/∅}	$(\langle s_m, \ldots, s_2, [\sigma_1]_X, s_0 \rangle, i) \Rightarrow (\langle s_m, \ldots, s_2, [\sigma_1 s_0]_X \rangle, i)$

Figure 2: Transition actions for left-corner parsing.

difference between ATTACH and REDUCE. The REDUCE action cannot deal with any node with more than two children. For this reason, the previous work converts parse trees into binarized ones. The conversion makes it difficult to capture the hierarchical structure of the parse trees. On the other hand, ATTACH action can handle more than two children. Therefore, our parser does not require such kind of tree binarization. These transition actions are similar to the ones described in (Henderson, 2003), although his parser uses right-binarized trees and does not identify head-children. Figure 2 summarizes the transition actions for our parser.

To guarantee that every non-terminal node has exactly one head child, our parser uses the following constraints:

- LEFTCORNER and ATTACH are not allowed when s_0 has no head child.
- ATTACH-H is not allowed when s_1 has a head child.

Table 2 shows the first several transition actions which derive the parse tree shown in Figure 1. Head children are indicated by the superscript *.

Previous transition-based constituent parsing does not handle nonlocal dependencies. One exception is the work of Maier (2015), who proposes shift-reduce constituent parsing with swap action. The parser can handle nonlocal dependencies represented as discontinuous constituents. In this framework, discontinuities are directly annotated by allowing crossing branches. Since the annotation style is quite different from the PTB annotation, the parser is not suitable for identifying the PTB style nonlocal dependencies.[2]

4 Nonlocal Dependency Identification

Nonlocal dependency identification consists of two subtasks:

- empty element detection.
- empty element resolution, which coindexes empty elements with their fillers.

Our parser can insert empty elements at an arbitrary position to realize empty element detection. This is in a similar manner as the in-processing approach. Our method coindexes empty elements with their fillers using simple heuristic rules, which are developed for our transition system.

4.1 Empty Element Detection

We introduce the following action to deal with empty elements:

$$\text{E-SHIFT}(E, t):$$
$$(\langle s_m, \ldots, s_0 \rangle, i) \Rightarrow (\langle s_m, \ldots, s_0, [t]_E \rangle, i)$$

This action simply inserts an empty element at an arbitrary position and pops up no element from the buffer (see the transition from #11 to #12 shown in Table 2 as an example).

4.2 Annotations

For empty element resolution, we augment the Penn Treebank. For nonlocal dependency types

[2]In (Evang and Kallmeyer, 2011), the PTB-style annotation of types *EXP, *ICH*, *RNR* and *T* is transformed into an annotation with crossing branches.

action	#	state
(initial state)	1	$(\langle\rangle, 0)$
SHIFT(DT)	2	$(\langle [\text{the}]_{\texttt{DT}}\rangle, 1)$
LEFTCORNER-$\emptyset$(NP)	3	$(\langle [[\text{the}]_{\texttt{DT}}]_{\texttt{NP}}\rangle, 1)$
SHIFT(NNP)	4	$(\langle [[\text{the}]_{\texttt{DT}}]_{\texttt{NP}}, [\text{U.N.}]_{\texttt{NNP}}\rangle, 2)$
ATTACH-$\emptyset$	5	$(\langle [[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}]_{\texttt{NP}}\rangle, 2)$
SHIFT(NN)	6	$(\langle [[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}]_{\texttt{NP}}, [\text{group}]_{\texttt{NN}}\rangle, 3)$
ATTACH-H	7	$(\langle [[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}[\text{group}]_{\texttt{NN*}}]_{\texttt{NP}}\rangle, 3)$
LEFTCORNER-H(NP)	8	$(\langle [[[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}[\text{group}]_{\texttt{NN*}}]_{\texttt{NP*}}]_{\texttt{NP}}\rangle, 3)$
SHIFT(WDT)	9	$(\langle [[[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}[\text{group}]_{\texttt{NN*}}]_{\texttt{NP*}}]_{\texttt{NP}}, [\text{that}]_{\texttt{WDT}}\rangle, 4)$
LEFTCORNER-H(WHNP-*T*-NP-L)	10	$(\langle [[[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}[\text{group}]_{\texttt{NN*}}]_{\texttt{NP*}}]_{\texttt{NP}}, [[\text{that}]_{\texttt{WDT*}}]_{\texttt{WHNP-*T*-NP-L}}\rangle, 4)$
LEFTCORNER-H(SBAR)	11	$(\langle [[[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}[\text{group}]_{\texttt{NN*}}]_{\texttt{NP*}}]_{\texttt{NP}}, [[[\text{that}]_{\texttt{WDT*}}]_{\texttt{WHNP-*T*-NP-L*}}]_{\texttt{SBAR}}\rangle, 4)$
E-SHIFT(-NONE-NP-L, *T*)	12	$(\langle [[[\text{the}]_{\texttt{DT}}[\text{U.N.}]_{\texttt{NNP}}[\text{group}]_{\texttt{NN*}}]_{\texttt{NP*}}]_{\texttt{NP}}, [[[\text{that}]_{\texttt{WDT*}}]_{\texttt{WHNP-*T*-NP-L*}}]_{\texttt{SBAR}}, [*\texttt{T}*]_{\texttt{-NONE-NP-L}}\rangle, 4)$

Table 2: An example of transition action sequence.

EXP, *ICH*, *RNR* and *T*, we assign the following information to each filler and each empty element:

- The nonlocal dependency type (only for filler).
- The nonlocal dependency category, which is defined as the category of the parent of the empty element.
- The relative position of the filler, which take a value from $\{\texttt{A}, \texttt{L}, \texttt{R}\}$. "A" means that the filler is an ancestor of the empty element. "L" ("R") means that the filler occurs to the left (right) of the empty element. Table 1 summarizes which value each empty element can take.

The information is utilized for coindexing empty elements with fillers. Below, we write $\texttt{n-type}(x)$, $\texttt{n-cat}(x)$ and $\texttt{n-posi}(x)$ for the information of a node x, respectively.

If an empty element of type $*$ is indexed, we annotate the empty element in the same way.[3] Furthermore, we assign a tag OBJCTRL to every empty element if its coindexed constituent does not have the function tag SBJ.[4] This enables our parser to distinguish between subject control and object control.

Figure 3 shows the augmented version of the parse tree of Figure 1.

4.3 Empty Element Resolution

Nonlocal dependency annotation in the Penn Treebank is based on Chomsky's GB-theory (Chomsky, 1981). This means that there exist c-command relations between empty elements and

[3]We omit its nonlocal dependency category, since it is always NP.

[4]In the Penn Treebank, every subject has the tag SBJ.

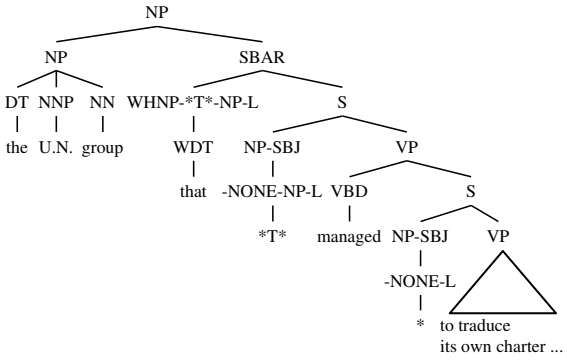


Figure 3: An augmented parse tree.

fillers in many cases. For example, all the empty elements in Figure 1 are c-commanded by their fillers. Our method coindexes empty elements with their fillers by simple heuristic rules based on the c-command relation.

4.3.1 C-command Relation

Here, we define c-command relation in a parse tree as follows:

- A node x *c-commands* a node y if and only if there exists some node z such that z is a sibling of $x (x \neq z)$ and y is a descendant of z.

It is difficult for previous transition-based shift-reduce constituent parsers to recognize c-command relations between nodes, since parse trees are binarized. On the other hand, our left-corner parser needs not to binarize parse trees and can easily recognize c-command relations. Furthermore, we extend c-command relation to handle nodes in a stack of our transition system. For two nodes x and y in a stack, the following statement necessarily holds:

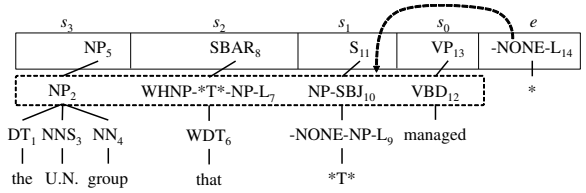


Figure 4: An example of resolution of $[*]$-NONE-L.

- Let $S = (\langle s_m, \ldots, s_0 \rangle, i)$ be a parser state. Let y be a descendant of s_j and x be a child of some node $s_k (j < k \leq m)$, respectively. Then, x c-commands y in any final state derived from the state S.

Below, we say that x c-commands y, even when the nodes x and y satisfy the above statement.

As an example, let us consider the state shown in Figure 4. The subscripts of nodes indicate the order in which the nodes are instantiated. The nodes in dotted box c-command the shifted node -NONE-L$_{14}$ in terms of the above statement. In the parse tree shown in Figure 3, which is derived from this state, these nodes c-commands -NONE-L$_{14}$ by the original definition.

4.3.2 Resolution Rules

Our parser coindexes an empty element with its filler, when E-SHIFT or ATTACH is executed. E-SHIFT action coindexes the shifted empty element e such that $\texttt{n-posi}(e) = \texttt{L}$ with its filler. ATTACH action coindexes the attached filler s_0 such that $\texttt{n-posi}(s_0) = \texttt{R}$ with its corresponding empty element. Resolution rules consist of three parts: PRECONDITION, CONSTRAINT and SELECT. Empty element resolution rule is applied to a state when the state satisfies PRECONDITION of the rule. CONSTRAINT represents the conditions which coindexed element must satisfy. SELECT can take two values ALL and RIGHTMOST. When there exist several elements satisfying the CONSTRAINT, SELECT determines how to select coindexed elements. ALL means that all the elements satisfying the CONSTRAINT are coindexed. RIGHTMOST selects the rightmost element satisfying the CONSTRAINT.

The most frequent type of nonlocal dependency in the Penn Treebank is $*$. Figure 5 shows the resolution rules for type $*$. Here, $\texttt{ch}(s)$ designates the set of the children of s. $\texttt{sbj}(x)$ means that x has a function tag SBJ. $\texttt{par}(x)$ designates the parent of x. $\texttt{cat}(x)$ represents the constituent category of x. $\texttt{des}(s)$ designates the set of the proper descendants of s. $\texttt{free}(x, \sigma)$ means that x is not coindexed with a node included in σ.

Rule: $*$-L
- PRECONDITION
  - ACTION=E-SHIFT(-NONE-L, $*$)
- CONSTRAINT for coindexed element x
  - $x \in \bigcup_{j=0}^{m} \texttt{ch}(s_j)$ # x c-commands e
  - $\texttt{sbj}(x)$
- SELECT: RIGHTMOST

Rule: $*$-L-OBJCTRL
- PRECONDITION
  - ACTION=E-SHIFT(-NONE-L-OBJCTRL, $*$)
- CONSTRAINT for coindexed element x
  - $x \in \bigcup_{j=0}^{m} \texttt{ch}(s_j)$ # x c-commands e
  - $\texttt{cat}(x) = \texttt{NP} \vee \texttt{cat}(x) = \texttt{PP}$
  - $\texttt{cat}(\texttt{par}(x)) = \texttt{VP}$
- SELECT: RIGHTMOST

Rule: $*$-R
- PRECONDITION
  - ACTION=ATTACH
  - $\texttt{sbj}(s_0)$
- CONSTRAINT for coindexed element x
  - $x \in \texttt{des}(s_1)$ # s_0 c-commands x
  - $x = [*]$-NONE-R
  - $\texttt{free}(x, \langle s_m, \ldots, s_0 \rangle)$
- SELECT: ALL

Figure 5: Resolution rules for type $*$.

The first rule $*$-L is applied to a state when E-SHIFT action inserts an empty element $e = [*]$-NONE-L. This rule seeks a subject which c-commands the shifted empty element. The first constraint means that the node x c-commands the empty element e, since the resulting state of E-SHIFT action is $(\langle s_m, \ldots, s_0, e \rangle, i)$, and x and e satisfy the statement in section 4.3.1. For example, the node NP-SBJ$_{10}$ shown in Figure 4 satisfies these constraints (the dotted box represents the first constraint). Therefore, our parser coindexes NP-SBJ$_{10}$ with -NONE-L$_{14}$.

The second rule $*$-L-OBJCTRL seeks an object instead of a subject. The second and third constraints identify whether or not x is an argument. If x is a prepositional phrase, our parser coindexes e with x's child noun phrase instead of x itself, in order to keep the PTB-style annotation.

The third rule $*$-R is for null subject of participial clause. Figure 6 shows an example of applying the rule $*$-R to a state. This rule is applied to a state when the transition action is ATTACH and s_0 is a subject. By definition, the first constraint means that s_0 c-commands x.

The second most frequent type is $*$T$*$. Figure 7 shows the rule for $*$T$*$. This rule is ap-

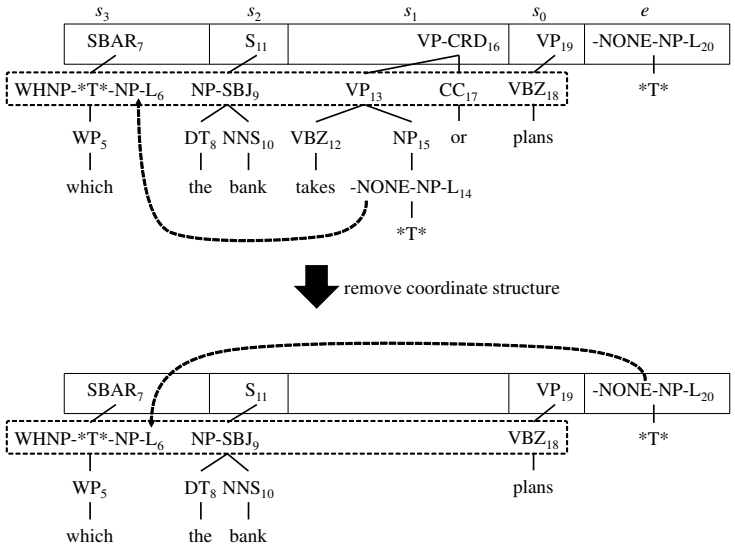


Figure 9: An example of resolution of $[*\mathtt{T}*]_{\mathtt{-NONE-NP-L}}$ in the case where the stack has coordinate structure.

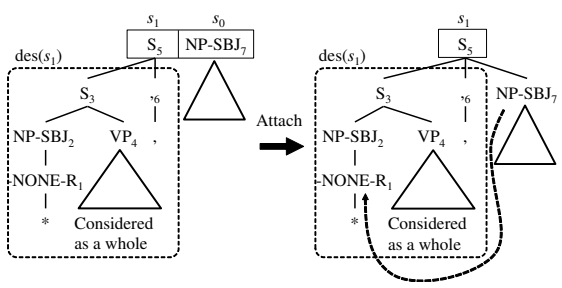


Figure 6: An example of resolution of $[*]_{\mathtt{-NONE-R}}$.

Rule: $*\mathtt{T}*$-L
PRECONDITION
ACTION=E-SHIFT(-NONE-L, $*\mathtt{T}*$)
CONSTRAINT for coindexed element x
x c-commands e
$x \in \bigcup_{s \in \mathtt{removeCRD}(\langle s_m, \ldots, s_0 \rangle)} \mathtt{ch}(s)$
$\mathtt{match}(x, e)$
$\mathtt{free}(x, \mathtt{removeCRD}(\langle s_m, \ldots, s_0 \rangle))$
SELECT: RIGHTMOST

Figure 7: Resolution rule for type $*\mathtt{T}*$.

plied to a state when E-SHIFT action inserts an empty element of type $*\mathtt{T}*$. Here, $\mathtt{match}(x, y)$ checks the consistency between x and y, that is, $\mathtt{match}(x, y)$ holds if and only if $\mathtt{n\text{-}type}(x) = \mathtt{n\text{-}type}(y)$, $\mathtt{n\text{-}cat}(x) = \mathtt{n\text{-}cat}(y)$, $\mathtt{n\text{-}posi}(x) = \mathtt{n\text{-}posi}(y)$, $\mathtt{cat}(x) \neq$ -NONE- and $\mathtt{cat}(y) =$ -NONE-. $\mathtt{removeCRD}(\langle s_m, \ldots, s_0 \rangle)$ is a stack which is obtained by removing $s_j (0 \leq j \leq m)$ which is annotated with a tag CRD.[5] The tag CRD

[5]We assign a tag CRD to a node, when it matches the pattern $[\cdots[\cdots]_X \cdots [\cdots]_{(\mathtt{CC|CONJP}|,|:)} \cdots [\cdots]_X \cdots]_X$.

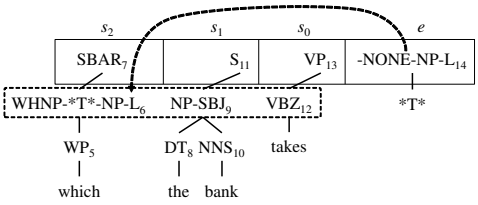


Figure 8: An example of resolution of $[*\mathtt{T}*]_{\mathtt{-NONE-NP-L}}$.

means that the node is coordinate structure. In general, each filler of type $*\mathtt{T}*$ is coindexed with only one empty element. However, a filler of type $*\mathtt{T}*$ can be coindexed with several empty elements if the empty elements are included in coordinate structure. This is the reason why our parser uses removeCRD. Figure 8 and 9 give examples of resolution for type $*\mathtt{T}*$.

The empty elements $[*\mathtt{T}*]_{\mathtt{-NONE-A}}$ are handled by an exceptional process. When ATTACH action is applied to a state $(\langle s_m, \ldots, s_0 \rangle, i)$ such that $\mathtt{cat}(s_0) = \mathtt{PRN}$, the parser coindexes the empty element $x = [*\mathtt{T}*]_{\mathtt{-NONE-A}}$ included in s_0 with s_1. More precisely, the coindexation is executed if the following conditions hold:

- $x \in \mathtt{des}(s_0)$
- $\mathtt{match}(s_1, x)$
- $\mathtt{free}(x, \langle s_m, \ldots, s_0 \rangle)$

For the other types of nonlocal dependencies, that is, $*\mathtt{EXP}*$, $*\mathtt{ICH}*$ and $*\mathtt{RNR}*$, we use a simi-

Rule: ∗EXP∗-R
  PRECONDITION
    ACTION=ATTACH
    s_0 is a filler of type ∗EXP∗
  CONSTRAINT for coindexed element x
    $x \in \bigcup_{j=1}^{m} \texttt{ch}(s_j)$ # x c-commands s_0
    $x = [[\texttt{it}]_{\texttt{PRP}}]_{\texttt{NP}}$
  SELECT: RIGHTMOST

Rule: ∗ICH∗-L
  PRECONDITION
    ACTION=E-SHIFT(-NONE-L, ∗ICH∗)
  CONSTRAINT for coindexed element x
    $\texttt{match}(x, e)$
    $\texttt{free}(x, \langle s_m, \ldots, s_0 \rangle)$
  SELECT: RIGHTMOST

Rule: ∗ICH∗-R
  PRECONDITION
    ACTION=ATTACH
    s_0 is a filler of type ∗ICH∗
  CONSTRAINT for coindexed element x
    $x \in \bigcup_{j=1}^{m} \texttt{des}(s_j)$
    $\texttt{match}(s_0, x)$
    $\texttt{free}(x, \langle s_m, \ldots, s_0 \rangle)$
  SELECT: RIGHTMOST

Rule: ∗RNR∗-R
  PRECONDITION
    ACTION=ATTACH
    s_0 is a filler of type ∗RNR∗
  CONSTRAINT for coindexed element x
    $x \in \texttt{des}(s_1)$ # s_0 c-commands x
    $\texttt{match}(s_0, x)$
    $\texttt{free}(x, \langle s_m, \ldots, s_0 \rangle)$
  SELECT: ALL

Figure 10: Resolution rule for ∗EXP∗, ∗ICH∗ and ∗RNR∗.

lar idea to design the resolution rules. Figure 10 shows the resolution rules.

These heuristic resolution rules are similar to the previous work (Campbell, 2004; Kato and Matsubara, 2015), which also utilizes c-command relation. An important difference is that we design heuristic rules not for fully-connected parse tree but for stack of parse trees derived by left-corner parsing. That is, the extend version of c-command relation plays an important role on our heuristic rules.

5 Parsing Strategy

We use a beam-search decoding with the structured perceptron (Collins, 2002). A transition action a for a state S has a score defined as follows:

$$score(S, a) = \mathbf{w} \cdot \mathbf{f}(S, a)$$

where $\mathbf{f}(S, a)$ is the feature vector for the state-action pair (S, a), and $\mathbf{w}$ is a weight vector. The

input: sentence $w_1 \cdots w_n$, beam size k
$H \leftarrow \{S_0\}$ # S_0 is the initial state for $w_1 \cdots w_0$
repeat N times **do**
  $C \leftarrow \{\}$
  for each $S \in H$ **do**
    for each possible action a **do**
      $S' \leftarrow$ apply a to S
      push S' to C
  $H \leftarrow k$ best states of C
return best final state in C

Figure 11: Beam-search parsing.

score of a state S' which is obtained by applying an action a to a state S is defined as follows:

$$score(S') = score(S) + score(S, a)$$

For the initial state S_0, $score(S_0) = 0$.

We learn the weight vector $\mathbf{w}$ by max-violation method (Huang et al., 2012) and average the weight vector to avoid overfitting the training data (Collins, 2002).

In our method, action sequences for the same sentence have different number of actions because of E-SHIFT action. To absorb the difference, we use an IDLE action, which are proposed in (Zhu et al., 2013):

$$\text{IDLE} : (\langle [\cdots]_{\texttt{TOP}} \rangle, n) \Rightarrow (\langle [\cdots]_{\texttt{TOP}} \rangle, n)$$

Figure 11 shows details of our beam-search parsing. The algorithm is the same as the previous transition-based parsing with structured perceptron. One point to be noted here is how to determine the maximum length of action sequence ($= N$) which the parser allows. Since it is impossible to know in advance how many empty elements a parse tree has, we need to set this parameter as a sufficiently larger value.

5.1 Features

A feature is defined as the concatenation of a transition action and a state feature which is extracted using a feature template. Table 3 shows our baseline feature templates. The feature templates are similar to the ones of (Zhang and Clark, 2009), which are standardly used as baseline templates for transition-based constituent parsing. Here, b_i and s_i stand for the i-th element of buffer and stack, respectively. $x.\texttt{c}$ represents x's augmented label. $x.\texttt{l}$, $x.\texttt{r}$ and $x.\texttt{h}$ represent x's leftmost, rightmost and head children. $x.\texttt{t}$ and $x.\texttt{w}$ represents x's head POS tag and head word, respectively. $x.\texttt{i}$ indicates whether or not the initial letter of x is capitalized. When a non-terminal node

type	feature templates
unigram	s_0.c $\circ$ s_0.t, s_0.c $\circ$ s_0.w,
	s_0.l.c $\circ$ s_0.l.w, s_0.r.c $\circ$ s_0.r.w, s_0.h.c $\circ$ s_0.h.w,
	s_1.c $\circ$ s_1.t, s_1.c $\circ$ s_1.w,
	s_1.l.c $\circ$ s_1.l.w, s_1.r.c $\circ$ s_1.r.w, s_1.h.c $\circ$ s_1.h.w,
	s_2.c $\circ$ s_2.t, s_2.c $\circ$ s_2.w, s_3.c $\circ$ s_3.t, s_3.c $\circ$ s_3.w,
	b_0.i, b_0.w, b_1.i, b_1.w, b_2.i, b_2.w, b_3.i, b_3.w
bigram	s_1.w $\circ$ s_0.w, s_1.c $\circ$ s_0.w, s_1.w $\circ$ s_0.c, s_1.w $\circ$ s_0.w,
	s_0.c $\circ$ b_0.i, s_0.c $\circ$ b_0.w, s_0.w $\circ$ b_0.i, s_0.w $\circ$ b_0.w,
	s_1.c $\circ$ b_0.i, s_1.c $\circ$ b_0.w, s_1.w $\circ$ b_0.i, s_1.w $\circ$ b_0.w,
	b_0.i $\circ$ b_1.i, b_0.w $\circ$ b_1.i, b_0.i $\circ$ b_1.w, b_0.w $\circ$ b_1.w
trigram	s_2.c $\circ$ s_1.c $\circ$ s_0.c, s_2.c $\circ$ s_1.c $\circ$ s_0.w,
	s_2.c $\circ$ s_1.w $\circ$ s_0.c, s_2.w $\circ$ s_1.c $\circ$ s_0.c,
	s_1.c $\circ$ s_0.c $\circ$ b_0.i, s_1.w $\circ$ s_0.c $\circ$ b_0.i,
	s_1.c $\circ$ s_0.w $\circ$ b_0.i, s_1.w $\circ$ s_0.w $\circ$ b_0.i

Table 3: Baseline feature templates.

feature templates
s_0.n$_0$.c, s_0.n$_1$.c, s_1.n$_0$.c, s_1.n$_1$.c,
$rest_2$.n$_0$.c, $rest_2$.n$_1$.c

Table 4: Nonlocal dependency feature templates.

does not yet have a head child, the head-based atomic features are set to a special symbol nil. To extract the features, we need to identify head children in parse trees. We use the head rules described in (Surdeanu et al., 2008).

In addition to these features, we introduce a new feature which is related to empty element resolution. When a transition action invokes empty element resolution described in section 4.3.2, we use as a feature, whether or not the procedure coindexes empty elements with a filler. Such a feature is difficult for a PCFG to capture. This feature enables our parsing model to learn the resolution rule preferences implicitly, while the training process is performed only with oracle action sequences.

In addition, we use features about free empty elements and fillers. Table 4 summarizes such feature templates. Here, x.n$_i$ stands for the i-th rightmost free element included in x, and $rest_i$ stands for the stack $\langle s_m, \ldots, s_i \rangle$.

6 Experiment

We conducted an experiment to evaluate the performance of our parser using the Penn Treebank. We used a standard setting, that is, section 02-21 is for the training data, section 22 is for the development data and section 23 is for the test data.

In training, we set the beam size k to 16 to achieve a good efficiency. We determined the optimal iteration number of perceptron training, and the beam size (k was set to 16, 32 and 64) for decoding on the development data. The maximum length of action sequences ($= N$) was set to $7n$, where n is the length of input sentence. This maximum length was determined to deal with the sentences in the training data.

type	system	F_1
TS	Zhu et al. (2013) (beam 16)	90.4
	Zhu et al. (2013)* (beam 16)	91.3
	Mi and Huang (2015) (beam 32)	90.3
	Mi and Huang (2015) (beam 32,DP)	90.8
	Thang et al. (2015) (A*)	91.1
	Ballesteros and Carreras (2015) (beam 64)	89.0
NDI	Charniak (2000)† (post-processing)	89.6
	Dienes and Dubey (2003a) (in-processing)	86.4
	Schmid (2006) (in-processing)	86.6
	Kato and Matsubara (2015) (in-processing)	87.7
ours	CF (beam 64)	88.9
	baseline features (beam 64)	89.0
	baseline + ND features (beam 64)	88.9

TS: transition-based parsers with structured perceptron.
NDI: parsers with nonlocal dependency identification.
DP: Dynamic Programming.
Zhu et al. (2013)* uses additional language resources.
†Johnson (2002) and Campbell (2004) used the output of Charniak's parser.

Table 5: Comparison for constituent parsing.

Table 5 presents the constituent parsing performances of our system and previous systems. We used the labeled bracketing metric PARSEVAL (Black et al., 1991). Here, "CF" is the parser which was learned from the training data where nonlocal dependencies are removed. This result demonstrates that our nonlocal dependency identification does not have a bad influence on constituent parsing. From the viewpoint of transition-based constituent parsing, our left-corner parser is somewhat inferior to other perceptron-based shift-reduce parsers. On the other hand, our parser outperforms the parsers which identify nonlocal dependency based on in-processing approach.

We use the metric proposed by Johnson (2002) to evaluate the accuracy of nonlocal dependency identification. Johnson's metric represents a nonlocal dependency as a tuple which consists of the type of the empty element, the category of the empty element, the position of the empty element, the category of the filler and the position of the filler. For example, the nonlocal dependency of the type $*$T$*$ in Figure 1 is represented as $(*\texttt{T}*, \texttt{NP}, [4, 4], \texttt{WHNP}, [3, 4])$. The precision and the recall are measured using these tuples. For more details, see (Johnson, 2002).

Table 6 shows the nonlocal dependency identification performances of our method and previous methods. Previous in-processing approach

				Unindexed empty elements are excluded		
	rec.	pre.	F_1	rec.	pre.	F_1
Johnson (2002) (post-processing)	63	73	68	–	–	–
D & D (2003a) (pre-processing)	66.0	80.5	72.6	–	–	–
D & D (2003b) (in-processing)	68.7	81.5	74.6	–	–	–
Campbell (2004) (post-processing)	75.1	78.3	76.7	–	–	–
Schmid (2006) (in-processing)	–	–	–	73.5	81.7	77.4
K&M (2015) (in-processing)	75.6	80.6	78.0	73.6	80.3	76.8
baseline features	70.4	79.7	74.8	65.4	81.1	72.4
+ ND features	75.5	81.4	78.4	73.8	79.8	76.7

Table 6: Comparison for nonlocal dependency identification.

achieved the state-of-the-art performance of nonlocal dependency identification, while it was inferior in terms of constituent parsing accuracy. Our nonlocal dependency identification is competitive with previous in-processing approach, and its accuracy of constituent parsing is higher than previous in-processing approach. As a whole, our parser achieves a good balance between constituent parsing and nonlocal dependency identification. Table 7 summarizes the accuracy of nonlocal dependency identification for each type of nonlocal dependency.

7 Conclusion

This paper proposed a transition-based parser which identifies nonlocal dependencies. Our parser achieves a good balance between constituent parsing and nonlocal dependency identification. In the experiment reported in this paper, we used simple features which are captured by nonlocal dependencies. In future work, we will develop lexical features which are captured by nonlocal dependencies.

Acknowledgements

This research was partially supported by the Grant-in-Aid for Scientific Research (B) (No.26280082) of JSPS.

(F, E, T)	freq.	pre.	rec.	F_1
(NP, NP, ∗)	1146	76.7	75.4	76.1
(−, -NONE-, 0)	545	92.3	83.7	87.8
(WHNP, NP, ∗T∗)	507	88.0	84.0	86.0
(−, NP, ∗)	477	69.0	71.7	70.3
(−, -NONE-, ∗U∗)	388	98.4	93.6	95.9
(S, S, ∗T∗)	277	83.6	80.9	82.2
(WHADVP, ADVP, ∗T∗)	164	82.1	70.1	75.7
(−, WHNP, 0)	107	73.3	51.4	60.4
(−, WHADVP, 0)	36	80.8	58.3	67.7
(PP, PP, ∗ICH∗)	29	20.0	3.5	5.9
(WHPP, PP, ∗T∗)	22	84.2	72.7	78.1
(SBAR, SBAR, ∗EXP∗)	16	71.4	31.3	43.5
(S, S, ∗ICH∗)	15	36.4	26.7	30.8
(S, S, ∗EXP∗)	14	50.0	42.9	46.2
(SBAR, SBAR, ∗ICH∗)	12	0.0	0.0	0.0
(−, NP, ∗?∗)	11	0.0	0.0	0.0
(−, S, ∗?∗)	9	100.0	11.1	20.0
(−, VP, ∗?∗)	8	45.5	62.5	52.6
(VP, VP, ∗T∗)	8	40.0	25.0	30.8
(ADVP, ADVP, ∗T∗)	7	80.0	57.1	66.7
(PP, PP, ∗T∗)	7	80.0	57.1	66.7
(−, -NONE-, ∗?∗)	7	0.0	0.0	0.0
(ADJP, ADJP, ∗T∗)	6	66.7	33.3	44.4
(ADVP, ADVP, ∗ICH∗)	6	0.0	0.0	0.0
(NP, NP, ∗ICH∗)	6	0.0	0.0	0.0
(VP, VP, ∗ICH∗)	6	0.0	0.0	0.0

Table 7: Accuracy of nonlocal dependency identification for all types of nonlocal dependency that occurred more than 5 times in section 23 of the Penn Treebank. F, E and T give the category of its filler, the category of its empty element and the type of nonlocal dependency, respectively.

References

Miguel Ballesteros and Xavier Carreras. 2015. Transition-based spinal parsing. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 289–299, Beijing, China, July.

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. *Bracketing Guidelines for Treebank II Style Penn Treebank Project*. University of Pennsylvania.

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the 4th DARPA Speech and Natural Language Workshop*, pages 306–311.

Shu Cai, David Chiang, and Yoav Goldberg. 2011. Language-independent parsing with empty elements. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 212–

216, Portland, Oregon, USA, June.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 645–652, Barcelona, Spain, July.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics*, pages 132–139, April.

Noam Chomsky. 1981. *Lectures on government and binding: The Pisa lectures*. Walter de Gruyter.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8, July.

Péter Dienes and Amit Dubey. 2003a. Antecedent recovery: Experiments with a trace tagger. In Michael Collins and Mark Steedman, editors, *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 33–40, July.

Péter Dienes and Amit Dubey. 2003b. Deep syntactic processing by combining shallow methods. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 431–438, Sapporo, Japan, July.

Kilian Evang and Laura Kallmeyer. 2011. Plcfrs parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland, October.

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

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured perceptron with inexact search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June.

Mark Johnson. 2002. A simple pattern-matching algorithm for recovering empty nodes and their antecedents. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 136–143, Philadelphia, Pennsylvania, USA, July.

Yoshihide Kato and Shigeki Matsubara. 2015. Identifying nonlocal dependencies in incremental parsing. *IEICE Transactions on Information and Systems*, E98-D(4):994–998.

Roger Levy and Christopher Manning. 2004. Deep dependencies from context-free statistical parsers: Correcting the surface dependency approximation. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 327–334, Barcelona, Spain, July.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China, July.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):310–330.

Haitao Mi and Liang Huang. 2015. Shift-reduce constituency parsing with dynamic programming and pos tag lattice. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1030–1035, Denver, Colorado, May–June.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia, October.

Kenji Sagae and Alon Lavie. 2006. A best-first probabilistic shift-reduce parser. In *Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 691–698, Sydney, Australia, July.

Helmut Schmid. 2006. Trace prediction and recovery with unlexicalized PCFGs and slash features. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 177–184, Sydney, Australia, July.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In *CoNLL 2008: Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 159–177, Manchester, England, August.

Shunsuke Takeno, Masaaki Nagata, and Kazuhide Yamamoto. 2015. Empty category detection using path features and distributed case frames. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1335–1340, Lisbon, Portugal, September.

Le Quang Thang, Hiroshi Noji, and Yusuke Miyao. 2015. Optimal shift-reduce constituent parsing with structured perceptron. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint*

Conference on Natural Language Processing (Volume 1: Long Papers), pages 1534–1544, Beijing, China, July.

Zhiguo Wang and Nianwen Xue. 2014. Joint POS tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 733–742, Baltimore, Maryland, June.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1169–1179, Beijing, China, July.

Bing Xiang, Xiaoqiang Luo, and Bowen Zhou. 2013. Enlisting the ghost: Modeling empty categories for machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 822–831, Sofia, Bulgaria, August.

Nianwen Xue and Yaqin Yang. 2013. Dependency-based empty category detection via phrase structure trees. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1051–1060, Atlanta, Georgia, June.

Yue Zhang and Stephen Clark. 2009. Transition-based parsing of the chinese treebank using a global discriminative model. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 162–171, Paris, France, October.

Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. 2013. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 434–443, Sofia, Bulgaria, August.

Siamese CBOW: Optimizing Word Embeddings for Sentence Representations

Tom Kenter[1] tom.kenter@uva.nl
Alexey Borisov[1, 2] alborisov@yandex-team.ru
Maarten de Rijke[1] derijke@uva.nl
[1] University of Amsterdam, Amsterdam
[2] Yandex, Moscow

Abstract

We present the *Siamese Continuous Bag of Words* (Siamese CBOW) model, a neural network for efficient estimation of high-quality sentence embeddings. Averaging the embeddings of words in a sentence has proven to be a surprisingly successful and efficient way of obtaining sentence embeddings. However, word embeddings trained with the methods currently available are not optimized for the task of sentence representation, and, thus, likely to be suboptimal. Siamese CBOW handles this problem by training word embeddings directly for the purpose of being averaged. The underlying neural network learns word embeddings by predicting, from a sentence representation, its surrounding sentences. We show the robustness of the Siamese CBOW model by evaluating it on 20 datasets stemming from a wide variety of sources.

1 Introduction

Word embeddings have proven to be beneficial in a variety of tasks in NLP such as machine translation (Zou et al., 2013), parsing (Chen and Manning, 2014), semantic search (Reinanda et al., 2015; Voskarides et al., 2015), and tracking the meaning of words and concepts over time (Kim et al., 2014; Kenter et al., 2015). It is not evident, however, how word embeddings should be combined to represent larger pieces of text, like sentences, paragraphs or documents. Surprisingly, simply averaging word embeddings of all words in a text has proven to be a strong baseline or feature across a multitude of tasks (Faruqui et al., 2014; Yu et al., 2014; Gershman and Tenenbaum, 2015; Kenter and de Rijke, 2015).

Word embeddings, however, are not optimized specifically for representing sentences. In this paper we present a model for obtaining word embeddings that are tailored specifically for the task of averaging them. We do this by directly including a comparison of sentence embeddings—the averaged embeddings of the words they contain—in the cost function of our network.

Word embeddings are typically trained in a fast and scalable way from unlabeled training data. As the training data is unlabeled, word embeddings are usually not task-specific. Rather, word embeddings trained on a large training corpus, like the ones from (Collobert and Weston, 2008; Mikolov et al., 2013b) are employed across different tasks (Socher et al., 2012; Kenter and de Rijke, 2015; Hu et al., 2014). These two qualities— (i) being trainable from large quantities of unlabeled data in a reasonable amount of time, and (ii) robust performance across different tasks—are highly desirable and allow word embeddings to be used in many large-scale applications. In this work we aim to optimize word embeddings for sentence representations in the same manner. We want to produce general purpose sentence embeddings that should score robustly across multiple test sets, and we want to leverage large amounts of unlabeled training material.

In the word2vec algorithm, Mikolov et al. (2013a) construe a supervised training criterion for obtaining word embeddings from unsupervised data, by predicting, for every word, its surrounding words. We apply this strategy at the sentence level, where we aim to predict a sentence from its adjacent sentences (Kiros et al., 2015; Hill et al., 2016). This allows us to use unlabeled training data, which is easy to obtain; the only restriction is that documents need to be split into sentences and that the order between sentences is preserved.

The main research question we address is

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 941–951,
Berlin, Germany, August 7-12, 2016.

whether directly optimizing word embeddings for the task of being averaged to produce sentence embeddings leads to word embeddings that are better suited for this task than word2vec does. Therefore, we test the embeddings in an unsupervised learning scenario. We use 20 evaluation sets that stem from a wide variety of sources (newswire, video descriptions, dictionary descriptions, microblog posts). Furthermore, we analyze the time complexity of our method and compare it to our baselines methods.

Summarizing, our main contributions are:

- We present Siamese CBOW, an efficient neural network architecture for obtaining high-quality word embeddings, directly optimized for sentence representations;
- We evaluate the embeddings produced by Siamese CBOW on 20 datasets, originating from a range of sources (newswire, tweets, video descriptions), and demonstrate the robustness of embeddings across different settings.

2 Siamese CBOW

We present the *Siamese Continuous Bag of Words* (CBOW) model, a neural network for efficient estimation of high-quality sentence embeddings. Quality should manifest itself in embeddings of semantically close sentences being similar to one another, and embeddings of semantically different sentences being dissimilar. An efficient and surprisingly successful way of computing a sentence embedding is to average the embeddings of its constituent words. Recent work uses pre-trained word embeddings (such as word2vec and GloVe) for this task, which are not optimized for sentence representations. Following these approaches, we compute sentence embeddings by averaging word embeddings, but we optimize word embeddings directly for the purpose of being averaged.

2.1 Training objective

We construct a supervised training criterion by having our network predict sentences occurring next to each other in the training data. Specifically, for a pair of sentences (s_i, s_j), we define a probability $p(s_i, s_j)$ that reflects how likely it is for the sentences to be adjacent to one another in the training data. We compute the probability $p(s_i, s_j)$ using a softmax function:

$$p_\theta(s_i, s_j) = \frac{e^{\cos(\mathbf{s_i^\theta}, \mathbf{s_j^\theta})}}{\sum_{s_{\prime} \in S} e^{\cos(\mathbf{s_i^\theta}, \mathbf{s_{\prime}^\theta})}}, \qquad (1)$$

where $\mathbf{s}_x^\theta$ denotes the embedding of sentence s_x, based on the model parameters θ. In theory, the summation in the denominator of Equation 1 should range over all possible sentences S, which is not feasible in practice. Therefore, we replace the set S with the union of the set S^+ of sentences that occur next to the sentence s_i in the training data, and S^-, a set of n randomly chosen sentences that are not observed next to the sentence s_i in the training data. The loss function of the network is categorical cross-entropy:

$$L = - \sum_{s_j \in \{S^+ \cup S^-\}} p(s_i, s_j) \cdot \log(p_\theta(s_i, s_j)),$$

where $p(\cdot)$ is the target probability the network should produce, and $p_\theta(\cdot)$ is the prediction it estimates based on parameters θ, using Equation 1. The target distribution simply is:

$$p(s_i, s_j) = \begin{cases} \frac{1}{|S^+|}, & \text{if } s_j \in S^+ \\ 0, & \text{if } s_j \in S^-. \end{cases}$$

I.e., if there are 2 positive examples (the sentences preceding and following the input sentence) and 2 negative examples, the target distribution is (0.5, 0.5, 0, 0).

2.2 Network architecture

Figure 1 shows the architecture of the proposed Siamese CBOW network. The input is a projection layer that selects embeddings from a word embedding matrix W (that is shared across inputs) for a given input sentence. The word embeddings are averaged in the next layer, which yields a sentence representation with the same dimensionality as the input word embeddings (the boxes labeled $average_i$ in Figure 1). The cosine similarities between the sentence representation for $sentence_i$ and the other sentences are calculated in the penultimate layer and a softmax is applied in the last layer to produce the final probability distribution.

2.3 Training

The weights in the word embedding matrix are the only trainable parameters in the Siamese CBOW network. They are updated using stochastic gradient descent. The initial learning rate is monotonically decreased proportionally to the number of training batches.

3 Experimental Setup

To test the efficacy of our siamese network for producing sentence embeddings we use multiple

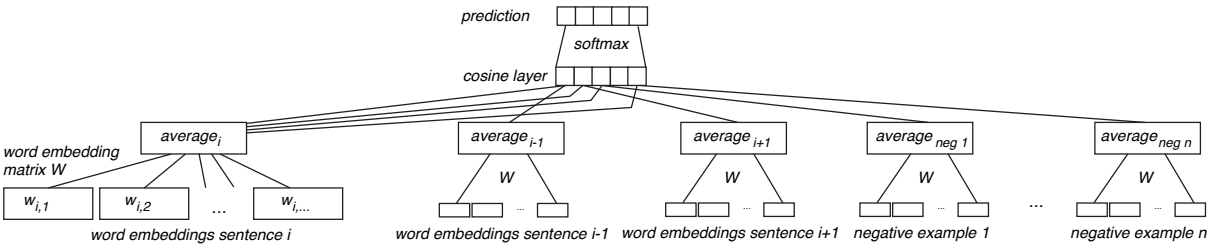


Figure 1: Siamese CBOW network architecture. (Input projection layer omitted.)

test sets. We use Siamese CBOW to learn word embeddings from an unlabeled corpus. For every sentence pair in the test sets, we compute two sentence representations by averaging the word embeddings of each sentence. Words that are missing from the vocabulary and, hence, have no word embedding, are omitted. The cosine similarity between the two sentence vectors is produced as a final semantic similarity score.

As we want a clean way to directly evaluate the embeddings on multiple sets we train our model and the models we compare with on exactly the same training data. We do not compute extra features, perform extra preprocessing steps or incorporate the embeddings in supervised training schemes. Additional steps like these are very likely to improve evaluation scores, but they would obscure our main evaluation purpose in this paper, which is to directly test the embeddings.

3.1 Data

We use the Toronto Book Corpus[1] to train word embeddings. This corpus contains 74,004,228 already pre-processed sentences in total, which are made up of 1,057,070,918 tokens, originating from 7,087 unique books. In our experiments, we consider tokens appearing 5 times or more, which leads to a vocabulary of 315,643 words.

3.2 Baselines

We employ two baselines for producing sentence embeddings in our experiments. We obtain similarity scores between sentence pairs from the baselines in the same way as the ones produced by Siamese CBOW, i.e., we calculate the cosine similarity between the sentence embeddings they produce.

Word2vec We average word embeddings trained with word2vec.[2] We use both architectures, Skipgram and CBOW, and apply default settings: minimum word frequency 5, word embedding size 300, context window 5, sample threshold 10^{-5}, no hierarchical softmax, 5 negative examples.

Skip-thought As a second baseline we use the sentence representations produced by the skip-thought architecture (Kiros et al., 2015).[3] Skip-thought is a recently proposed method that learns sentence representations in a different way from ours, by using recurrent neural networks. This allows it to take word order into account. As it trains sentence embeddings from unlabeled data, like we do, it is a natural baseline to consider.

Both methods are trained on the Toronto Book Corpus, the same corpus used to train Siamese CBOW. We should note that as we use skip-thought vectors as trained by Kiros et al. (2015), skip-thought has an advantage over both word2vec and Siamese CBOW as the vocabulary used for encoding sentences contains 930,913 words, three times the size of the vocabulary that we use.

3.3 Evaluation

We use 20 SemEval datasets from the SemEval semantic textual similarity task in 2012, 2013, 2014 and 2015 (Agirre et al., 2012; Agirre et al., 2013; Agirre et al., 2014; Agirre et al., 2015), which consist of sentence pairs from a wide array of sources (e.g., newswire, tweets, video descriptions) that have been manually annotated by multiple human assessors on a 5 point scale (1: semantically unrelated, 5: semantically similar). In the ground truth, the final similarity score for every sentence pair is

[1]The corpus can be downloaded from `http://www.cs.toronto.edu/~mbweb/`; cf. (Zhu et al., 2015).

[2]The code is available from `https://code.google.com/archive/p/word2vec/`.

[3]The code and the trained models can be downloaded from `https://github.com/ryankiros/skip-thoughts/`.

Table 1: Results on SemEval datasets in terms of Pearson's r (Spearman's r). Highest scores, in terms of Pearson's r, are displayed in bold. Siamese CBOW runs statistically significantly different from the word2vec CBOW baseline runs are marked with a †. See §3.3 for a discussion of the statistical test used.

Dataset	w2v skipgram	w2v CBOW	skip-thought	Siamese CBOW
2012				
MSRpar	.3740 (.3991)	.3419 (.3521)	.0560 (.0843)	**.4379**† (.4311)
MSRvid	.5213 (.5519)	.5099 (.5450)	**.5807** (.5829)	.4522† (.4759)
OnWN	.6040 (.6476)	.6320 (.6440)	.6045 (.6431)	**.6444**† (.6475)
SMTeuroparl	.3071 (.5238)	.3976 (.5310)	.4203 (.4999)	**.4503**† (.5449)
SMTnews	**.4487** (.3617)	.4462 (.3901)	.3911 (.3628)	.3902† (.4153)
2013				
FNWN	**.3480** (.3401)	.2736 (.2867)	.3124 (.3511)	.2322† (.2235)
OnWN	.4745 (.5509)	**.5165** (.6008)	.2418 (.2766)	.4985† (.5227)
SMT	.1838 (.2843)	.2494 (.2919)	**.3378** (.3498)	.3312† (.3356)
headlines	.5935 (.6044)	.5730 (.5766)	.3861 (.3909)	**.6534**† (.6516)
2014				
OnWN	.5848 (.6676)	.6068 (.6887)	.4682 (.5161)	**.6073**† (.6554)
deft-forum	.3193 (.3810)	.3339 (.3507)	.3736 (.3737)	**.4082**† (.4188)
deft-news	.5906 (.5678)	.5737 (.5577)	.4617 (.4762)	**.5913**† (.5754)
headlines	.5790 (.5544)	.5455 (.5095)	.4031 (.3910)	**.6364**† (.6260)
images	.5131 (.5288)	.5056 (.5213)	.4257 (.4233)	**.6497**† (.6484)
tweet-news	.6336 (.6544)	.6897 (.6615)	.5138 (.5297)	**.7315**† (.7128)
2015				
answ-forums	.1892 (.1463)	.1767 (.1294)	**.2784** (.1909)	.2181 (.1469)
answ-students	.3233 (.2654)	.3344 (.2742)	.2661 (.2068)	**.3671**† (.2824)
belief	.2435 (.2635)	.3277 (.3280)	.4584 (.3368)	**.4769** (.3184)
headlines	.1875 (.0754)	.1806 (.0765)	.1248 (.0464)	**.2151**† (.0846)
images	.2454 (.1611)	.2292 (.1438)	.2100 (.1220)	**.2560**† (.1467)

the mean of the annotator judgements, and as such can be a floating point number like 2.685.

The evaluation metric used by SemEval, and hence by us, is Pearson's r. As Spearman's r is often reported as well, we do so too.

Statistical significance To see whether Siamese CBOW yields significantly different scores for the same input sentence pairs from word2vec CBOW—the method it is theoretically most similar to—we compute Wilcoxon signed-rank test statistics between all runs on all evaluation sets. Runs are considered statistically significantly different for p-values < 0.0001.

3.4 Network

To comply with results reported in other research (Mikolov et al., 2013b; Kusner et al., 2015) we fix the embedding size to 300 and only consider words appearing 5 times or more in the training corpus. We use 2 negative examples (see §4.2.2 for an analysis of different settings). The embeddings are initialized randomly, by drawing from a normal distribution with $\mu = 0.0$ and $\sigma = 0.01$. The batch size is 100. The initial learning rate α is 0.0001, which we obtain by observing the loss on the training data. Training consists of one epoch.

We use Theano (Theano Development Team, 2016) to implement our network.[4] We ran our experiments on GPUs in the DAS5 cluster (Bal et al., 2016).

[4]The code for Siamese CBOW is available under an open-source license at `https://bitbucket.org/TomKenter/siamese-cbow`.

4 Results

In this section we present the results of our experiments, and analyze the stability of Siamese CBOW with respect to its (hyper)parameters.

4.1 Main experiments

In Table 1, the results of Siamese CBOW on 20 SemEval datasets are displayed, together with the results of the baseline systems. As we can see from the table, Siamese CBOW outperforms the baselines in the majority of cases (14 out of 20). The very low scores of skip-thought on MSRpar appear to be a glitch, which we will ignore.

It is interesting to see that for the set with the highest average sentence length (2013 SMT, with 24.7 words per sentence on average) Siamese CBOW is very close to skip-thought, the best performing baseline. In terms of lexical term overlap, unsurprisingly, all methods have trouble with the sets with little overlap (2013 FNWN, 2015 answers-forums, which both have 7% lexical overlap). It is interesting to see, however, that for the next two sets (2015 belief and 2012 MSRpar, 11% and 14% overlap respectively) Siamese CBOW manages to get the best performance. The highest performance on all sets is 0.7315 Pearson's r of Siamese CBOW on the 2014 tweet-news set. This figure is not very far from the best performing SemEval run that year which has 0.792 Pearson's r. This is remarkable as Siamese CBOW is completely unsupervised, while the NTNU system which scored best on this set (Lynum et al., 2014) was optimized using multiple training sets.

In recent work, Hill et al. (2016) present FastSent, a model similar to ours (see §5 for a more elaborate discussion); results are not reported for all evaluation sets we use, and hence, we compare the results of FastSent and Siamese CBOW separately, in Table 2.

FastSent and Siamese CBOW each outperform the other on half of the evaluation sets, which clearly suggests that the differences between the two methods are complementary.[5]

4.2 Analysis

Next, we investigate the stability of Siamese CBOW with respect to its hyper-parameters. In particular, we look into stability across iterations, different numbers of negative examples, and the dimensionality of the embeddings. Other parameter settings are set as reported in §3.4.

Table 2: Results on SemEval 2014 datasets in terms of Pearson's r (Spearman's r). Highest scores (in Pearson's r) are displayed in bold. FastSent results are reprinted from (Hill et al., 2016) where they are reported in two-digit precision.

Dataset	FastSent	Siamese CBOW
OnWN	**.74** (.70)	.6073 (.6554)
deft-forum	**.41** (.36)	.4082 (.4188)
deft-news	.58 (.59)	**.5913** (.5754)
headlines	.57 (.59)	**.6364** (.6260)
images	**.74** (.78)	.6497 (.6484)
tweet-news	.63 (.66)	**.7315** (.7128)

4.2.1 Performance across iterations

Ideally, the optimization criterion of a learning algorithm ranges over the full domain of its loss function. As discussed in §2, our loss function only observes a sample. As such, convergence is not guaranteed. Regardless, an ideal learning system should not fluctuate in terms of performance relative to the amount of training data it observes, provided this amount is substantial: as training proceeds the performance should stabilize.

To see whether the performance of Siamese CBOW fluctuates during training we monitor it during 5 epochs; at every 10,000,000 examples, and at the end of every epoch. Figure 2 displays the results for all 20 datasets. We observe that on the majority of datasets the performance shows very little variation. There are three exceptions. The performance on the 2014 deft-news dataset steadily decreases while the performance on 2013 OnWN steadily increases, though both seem to stabilize at the end of epoch 5. The most notable exception, however, is 2012 MSRvid, where the score, after an initial increase, drops consistently. This effect might be explained by the fact that this evaluation set primarily consists of very short sentences—it has the lowest average sentence length of all set: 6.63 with a standard deviation of 1.812. Therefore, a 300-dimensional representation appears too large for this dataset; this hypothesis is supported by the fact that 200-dimensional embeddings work slightly better for this dataset (see Figure 4).

[5]The comparison is to be interpreted with caution as it is not evident what vocabulary was used for the experiments in (Hill et al., 2016); hence, the differences observed here might simply be due to differences in vocabulary coverage.

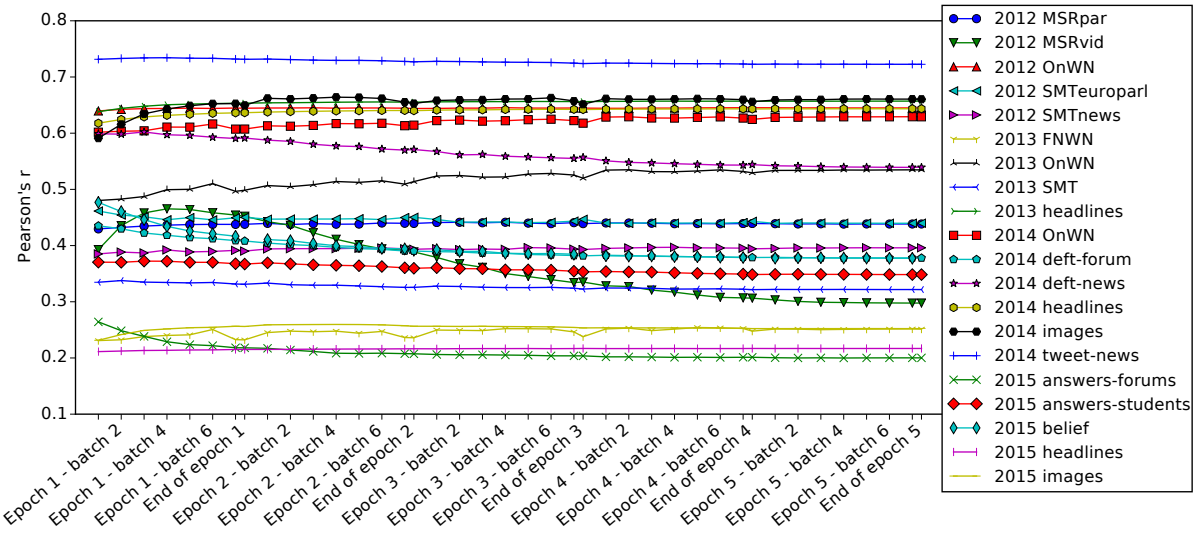


Figure 2: Performance of Siamese CBOW across 5 iterations.

4.2.2 Number of negative examples

In Figure 3, the results of Siamese CBOW in terms of Pearson's r are plotted for different numbers of negative examples. We observe that on most sets, the number of negative examples has limited effect on the performance of Siamese CBOW. Choosing a higher number, like 10, occasionally leads to slightly better performance, e.g., on the 2013 FNWN set. However, a small number like 1 or 2 typically suffices, and is sometimes markedly better, e.g., in the case of the 2015 belief set. As a high number of negative examples comes at a substantial computational cost, we conclude from the findings presented here that, although Siamese CBOW is robust against different settings of this parameter, setting the number of negative examples to 1 or 2 should be the default choice.

4.2.3 Number of dimensions

Figure 4 plots the results of Siamese CBOW for different numbers of vector dimensions. We observe from the figure that for some sets (most notably 2014 deft-forum, 2015 answ-forums and 2015 belief) increasing the number of embedding dimensions consistently yields higher performance. A dimensionality that is too low (50 or 100) invariably leads to inferior results. As, similar to a higher number of negative examples, a higher embedding dimension leads to higher computational costs, we conclude from these findings

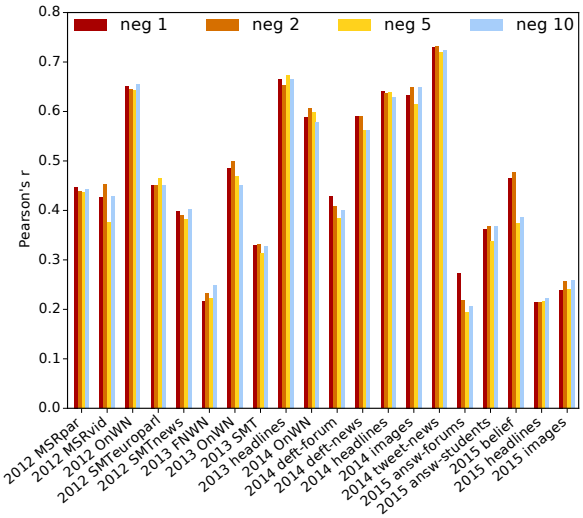


Figure 3: Performance of Siamese CBOW with different numbers of negative examples.

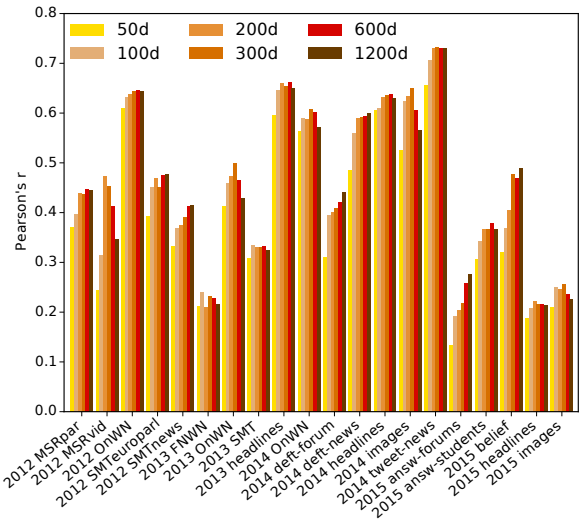


Figure 4: Performance of Siamese CBOW across number of embedding dimensions.

that a moderate number of dimensions (200 or 300) is to be preferred.

4.3 Time complexity

For learning systems, time complexity comes into play in the training phase and in the prediction phase. For an end system employing sentence embeddings, the complexity at prediction time is the most crucial factor, which is why we omit an analysis of training complexity. We focus on comparing the time complexity for generating sentence embeddings for Siamese CBOW, and compare it to the baselines we use.

The complexity of all algorithms we consider is $\mathcal{O}(n)$, i.e., linear in the number of input terms. As in practice the number of arithmetic operations is the critical factor in determining computing time, we will now focus on these.

Both word2vec and the Siamese CBOW compute embeddings of a text $T = t_1, \ldots, t_{|T|}$ by averaging the term embeddings. This requires $|T|-1$ vector additions, and 1 multiplication by a scalar value (namely, $1/|T|$). The skip-thought model is a recurrent neural network with GRU cells, which computes a set of equations for every term t in T, which we reprint for reference (Kiros et al., 2015):

$$\begin{aligned}
\mathbf{r}^t &= \sigma(\mathbf{W}_r\mathbf{x}^t + \mathbf{U}_r\mathbf{h}^{t-1}) \\
\mathbf{z}^t &= \sigma(\mathbf{W}_z\mathbf{x}^t + \mathbf{U}_z\mathbf{h}^{t-1}) \\
\overline{\mathbf{h}}^t &= \tanh(\mathbf{W}\mathbf{x}^t + \mathbf{U}(\mathbf{r}^t \odot \mathbf{h}^{t-1})) \\
\mathbf{h}^t &= (1 - \mathbf{z}^t) \odot \mathbf{h}^{t-1} + \mathbf{z}^t \odot \overline{\mathbf{h}}^t
\end{aligned}$$

As we can see from the formulas, there are $5|T|$ vector additions (+/-), $4|T|$ element-wise multiplications by a vector, $3|T|$ element-wise operations and $6|T|$ matrix multiplications, of which the latter, the matrix multiplications, are most expensive.

This considerable difference in numbers of arithmetic operations is also observed in practice. We run tests on a single CPU, using identical code for extracting sentences from the evaluation sets, for every method. The sentence pairs are presented one by one to the models. We disregard the time it takes to load models. Speedups might of course be gained for all methods by presenting the sentences in batches to the models, by computing sentence representations in parallel and by running code on a GPU. However, as we are interested in the differences between the systems, we run the most simple and straightforward scenario. Table 3 lists the number of seconds each method takes to generate and compare sentence embeddings for an input sentence pair. The difference between word2vec and Siamese CBOW is because of a different implementation of word lookup.

Table 3: Time spent per method on all 20 SemEval datasets, 17,608 sentence pairs, and the average time spent on a single sentence pair (time in seconds unless indicated otherwise).

	20 sets	1 pair
Siamese CBOW (300d)	7.7	0.0004
word2vec (300d)	7.0	0.0004
skip-thought (1200d)	98,804.0	5.6

We conclude from the observations presented here, together with the results in §4.1, that in a setting where speed at prediction time is pivotal, simple averaging methods like word2vec or Siamese CBOW are to be preferred over more involved methods like skip-thought.

4.4 Qualitative analysis

As Siamese CBOW directly averages word embeddings for sentences, we expect it to learn that words with little semantic impact have a low vector norm. Indeed, we find that the 10 words with lowest vector norm are *to*, *of*, *and*, *the*, *a*, *in*, *that*, *with*, *on*, and *as*. At the other side of the spectrum we find many personal pronouns: *had*, *they*, *we*, *me*, *my*, *he*, *her*, *you*, *she*, *I*, which is natural given that the corpus on which we train consists of fiction, which typically contains dialogues.

It is interesting to see what the differences in related words are between Siamese CBOW and word2vec when trained on the same corpus. For example, for a cosine similarity > 0.6, the words related to *her* in word2vec space are *she*, *his*, *my* and *hers*. For Siamese CBOW, the only closely related word is *she*. Similarly, for the word *me*, word2vec finds *him* as most closely related word, while Siamese CBOW comes up with *I* and *my*. It seems from these few examples that Siamese CBOW learns to be very strict in choosing which words to relate to each other.

From the results presented in this section we conclude that optimizing word embeddings for the task of being averaged across sentences with Siamese CBOW leads to embeddings that are effective in a large variety of settings. Furthermore, Siamese CBOW is robust to different parameter settings and its performance is stable across itera-

tions. Lastly, we show that Siamese CBOW is fast and efficient in computing sentence embeddings at prediction time.

5 Related Work

A distinction can be made between supervised approaches for obtaining representations of short texts, where a model is optimised for a specific scenario, given a labeled training set, and unsupervised methods, trained on unlabeled data, that aim to capture short text semantics that are robust across tasks. In the first setting, word vectors are typically used as features or network initialisations (Kenter and de Rijke, 2015; Hu et al., 2014; Severyn and Moschitti, 2015; Yin and Schütze, 2015). Our work can be classified in the latter category of unsupervised approaches.

Many models related to the one we present here are used in a multilingual setting (Hermann and Blunsom, 2014b; Hermann and Blunsom, 2014a; Lauly et al., 2014). The key difference between this work and ours is that in a multilingual setting the goal is to predict, from a distributed representation of an input sentence, the same sentence in a different language, whereas our goals is to predict surrounding sentences.

Wieting et al. (2016) apply a model similar to ours in a related but different setting where explicit semantic knowledge is leveraged. As in our setting, word embeddings are trained by averaging them. However, unlike in our proposal, a margin-based loss function is used, which involves a parameter that has to be tuned. Furthermore, to select negative examples, at every training step, a computationally expensive comparison is made between all sentences in the training batch. The most crucial difference is that a large set of phrase pairs explicitly marked for semantic similarity has to be available as training material. Obtaining such high-quality training material is non-trivial, expensive and limits an approach to settings for which such material is available. In our work, we leverage unlabeled training data, of which there is a virtually unlimited amount.

As detailed in §2, our network predicts a sentence from its neighbouring sentences. The notion of learning from context sentences is also applied in (Kiros et al., 2015), where a recurrent neural network is employed. Our way of averaging the vectors of words contained in a sentence is more similar to the CBOW architecture of word2vec (Mikolov et al., 2013a), in which all context word vectors are aggregated to predict the one omitted word. A crucial difference between our approach and the word2vec CBOW approach is that we compare sentence representations directly, rather than comparing a (partial) sentence representation to a word representation. Given the correspondence between word2vec's CBOW model and ours, we included it as a baseline in our experiments in §3. As the skip-gram architecture has proven to be a strong baseline too in many settings, we include it too.

Yih et al. (2011) also propose a siamese architecture. Short texts are represented by tf-idf vectors and a linear combination of input weights is learnt by a two-layer fully connected network, which is used to represent the input text. The cosine similarity between pairs of representations is computed, but unlike our proposal, the differences between similarities of a positive and negative sentence pair are combined in a logistic loss function.

Finally, independently from our work, Hill et al. (2016) also present a log-linear model. Rather than comparing sentence representations to each other, as we propose, words in one sentence are compared to the representation of another sentence. As both input and output vectors are learnt, while we tie the parameters across the entire model, Hill et al. (2016)'s model has twice as many parameters as ours. Most importantly, however, the cost function used in (Hill et al., 2016) is crucially different from ours. As words in surrounding sentences are being compared to a sentence representation, the final layer of their network produces a softmax over the entire vocabulary. This is fundamentally different from the final softmax over cosines between sentence representations that we propose. Furthermore, the softmax over the vocabulary is, obviously, of vocabulary size, and hence grows when bigger vocabularies are used, causing additional computational cost. In our case, the size of the softmax is the number of positive plus negative examples (see §2.1). When the vocabulary grows, this size is unaffected.

6 Conclusion

We have presented Siamese CBOW, a neural network architecture that efficiently learns word embeddings optimized for producing sentence representations. The model is trained using only unla-

beled text data. It predicts, from an input sentence representation, the preceding and following sentence.

We evaluated the model on 20 test sets and show that in a majority of cases, 14 out of 20, Siamese CBOW outperforms a word2vec baseline and a baseline based on the recently proposed skip-thought architecture. As further analysis on various choices of parameters show that the method is stable across settings, we conclude that Siamese CBOW provides a robust way of generating high-quality sentence representations.

Word and sentence embeddings are ubiquitous and many different ways of using them in supervised tasks have been proposed. It is beyond the scope of this paper to provide a comprehensive analysis of all supervised methods using word or sentence embeddings and the effect Siamese CBOW would have on them. However, it would be interesting to see how Siamese CBOW embeddings would affect results in supervised tasks.

Lastly, although we evaluated Siamese CBOW on sentence pairs, there is no theoretical limitation restricting it to sentences. It would be interesting to see how embeddings for larger pieces of texts, such as documents, would perform in document clustering or filtering tasks.

Acknowledgments

The authors wish to express their gratitude for the valuable advice and relevant pointers of the anonymous reviewers. Many thanks to Christophe Van Gysel for implementation-related help. This research was supported by Ahold, Amsterdam Data Science, the Bloomberg Research Grant program, the Dutch national program COMMIT, Elsevier, the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nr 312827 (VOX-Pol), the ESF Research Network Program ELIAS, the Royal Dutch Academy of Sciences (KNAW) under the Elite Network Shifts project, the Microsoft Research Ph.D. program, the Netherlands eScience Center under project number 027.012.105, the Netherlands Institute for Sound and Vision, the Netherlands Organisation for Scientific Research (NWO) under project nrs 727.011.005, 612.001.116, HOR-11-10, 640.006.013, 612.066.930, CI-14-25, SH-322-15, 652.002.001, 612.001.551, the Yahoo Faculty Research and Engagement Program, and Yandex. All content represents the opinion of the authors, which is not necessarily shared or endorsed by their respective employers and/or sponsors.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task (*SEM 2013)*, pages 32–43.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 81–91.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, I Nigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 252–263.

Henri Bal, Dick Epema, Cees de Laat, Rob van Nieuwpoort, John Romein, Frank Seinstra, Cees Snoek, and Harry Wijshoff. 2016. A medium-scale distributed system for computer science research: Infrastructure for the long term. *Computer*, 49(5):54–63.

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

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning (ICML 2008)*, pages 160–167.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. 2014. Retrofitting word vectors to semantic lexicons. In *Proceedings of the North American Chapter of the*

Association for Computational Linguistics (NAACL 2014).

Samuel J. Gershman and Joshua B. Tenenbaum. 2015. Phrase similarity in humans and machines. In *Proceedings of the 37th Annual Conference of the Cognitive Science Society*, pages 776–781.

Karl Moritz Hermann and Phil Blunsom. 2014a. Multilingual distributed representations without word alignment. In *Proceedings of the International Conference on Learning Representations (ICLR 2014)*.

Karl Moritz Hermann and Phil Blunsom. 2014b. Multilingual models for compositional distributed semantics. In *Proceeedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 58–68.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2016)*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems (NIPS 2014)*, pages 2042–2050.

Tom Kenter and Maarten de Rijke. 2015. Short text similarity with word embeddings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM 2015)*, pages 1411–1420.

Tom Kenter, Melvin Wevers, Pim Huijnen, and Maarten de Rijke. 2015. Ad hoc monitoring of vocabulary shifts over time. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM 2015)*, pages 1191–1200.

Yoon Kim, I Yi-Chiu., Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *Proceeedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 61–65.

Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 3294–3302. Curran Associates, Inc.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Q Weinberger. 2015. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*, pages 957–966.

Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems (NIPS 2014)*, pages 1853–1861.

André Lynum, Partha Pakray, Björn Gambäck, and Sergio Jimenez. 2014. Ntnu: Measuring semantic similarity with sublexical feature representations and soft cardinality. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 448–453.

Tomas Mikolov, Kai Chen, Greg S. Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv e-prints*, 1301.3781.

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

Ridho Reinanda, Edgar Meij, and Maarten de Rijke. 2015. Mining, ranking and recommending entity aspects. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 263–272.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2015)*, pages 373–382.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 1201–1211.

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten de Rijke, and Wouter Weerkamp. 2015. Learning to explain entity relationships in knowledge graphs. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2015)*, pages 564–574.

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

Wentau Yih, Kristina Toutanova, John C. Platt, and Christopher Meek. 2011. Learning discriminative projections for text similarity measures. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning*, pages 247–256.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL 2015)*, pages 901–911.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS 2014 Deep Learning and Representation Learning Workshop*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 19–27.

Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1393–1398.

Unanimous Prediction for 100% Precision with Application to Learning Semantic Mappings

Fereshte Khani
Stanford University
fereshte@cs.stanford.edu

Martin Rinard
MIT
rinard@lcs.mit.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

Can we train a system that, on any new input, either says "don't know" or makes a prediction that is guaranteed to be correct? We answer the question in the affirmative provided our model family is well-specified. Specifically, we introduce the unanimity principle: only predict when all models consistent with the training data predict the same output. We operationalize this principle for semantic parsing, the task of mapping utterances to logical forms. We develop a simple, efficient method that reasons over the infinite set of all consistent models by only checking two of the models. We prove that our method obtains 100% precision even with a modest amount of training data from a possibly adversarial distribution. Empirically, we demonstrate the effectiveness of our approach on the standard GeoQuery dataset.

1 Introduction

If a user asks a system "*How many painkillers should I take?*", it is better for the system to say "don't know" rather than making a costly incorrect prediction. When the system is learned from data, uncertainty pervades, and we must manage this uncertainty properly to achieve our precision requirement. It is particularly challenging since training inputs might not be representative of test inputs due to limited data, covariate shift (Shimodaira, 2000), or adversarial filtering (Nelson et al., 2009; Mei and Zhu, 2015). In this unforgiving setting, can we still train a system that is *guaranteed* to either abstain or to make the correct prediction?

Our present work is motivated by the goal of building reliable question answering systems and natural language interfaces. Our goal is to learn a semantic mapping from examples of utterance-logical form pairs (Figure 1). More generally, we assume the input x is a bag (multiset) of source atoms (e.g., words $\{area, of, Ohio\}$), and the output y is a bag of target atoms (e.g., predicates $\{\texttt{area}, \texttt{OH}\}$). We consider learning mappings M that decompose according to the multiset sum: $M(x) = \uplus_{s \in x} M(s)$ (e.g., $M(\{Ohio\}) = \{\texttt{OH}\}$, $M(\{area,of,Ohio\}) = \{\texttt{area},\texttt{OH}\}$). The main challenge is that an individual training example (x, y) does not tell us which source atoms map to which target atoms.[1]

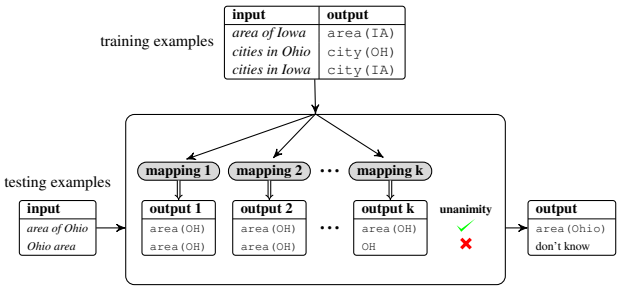


Figure 1: Given a set of training examples, we compute $\mathcal{C}$, the set of all mappings consistent with the training examples. On an input x, if all mappings in $\mathcal{C}$ unanimously predict the same output, we return that output; else we return "don't know".

How can a system be 100% sure about something if it has seen only a small number of possibly non-representative examples? Our approach is based on what we call the *unanimity* principle (Section 2.1). Let $\mathcal{M}$ be a model family that contains the true mapping from inputs to outputs. Let $\mathcal{C}$ be the subset of mappings that are consistent

[1] A semantic parser further requires modeling the context dependence of words and the logical form structure joining the predicates. Our framework handles these cases with a different choice of source and target atoms (see Section 4.2).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 952–962,
Berlin, Germany, August 7-12, 2016.

with the training data. If all mappings $M \in \mathcal{C}$ unanimously predict the same output on a test input, then we return that output; else we return "don't know" (see Figure 1). The unanimity principle provides robustness to the particular input distribution, so that we can tolerate even adversaries (Mei and Zhu, 2015), provided the training outputs are still mostly correct.

To operationalize the unanimity principle, we need to be able to efficiently reason about the predictions of all consistent mappings $\mathcal{C}$. To this end, we represent a mapping as a matrix M, where M_{st} is number of times target atom t (e.g., `OH`) shows up for each occurrence of the source atom s (e.g., *Ohio*) in the input. We show that unanimous prediction can be performed by solving two integer linear programs. With a linear programming relaxation (Section 3), we further show that checking unanimity over $\mathcal{C}$ can be done very efficiently without any optimization but rather by checking the predictions of just two random mappings, while still guaranteeing 100% precision with probability 1 (Section 3.2).

We further relax the linear program to a linear system, which gives us a geometric view of the unanimity: We predict on a new input if it can be expressed as a "linear combination" of the training inputs. As an example, suppose we are given training data consisting of (CI) *cities in Iowa*, (CO) *cities in Ohio*, and (AI) *area of Iowa* (Figure 1). We can compute (AO) *area of Ohio* by analogy: (AO) = (CO) - (CI) + (AI). Other reasoning patterns fall out from more complex linear combinations.

We can handle noisy data (Section 3.4) by asking for unanimity over additional slack variables. We also show how the linear algebraic formulation enables other extensions such as learning from denotations (Section 5.1), active learning (Section 5.2), and paraphrasing (Section 5.3). We validate our methods in Section 4. On artificial data generated from an adversarial distribution with noise, we show that unanimous prediction obtains 100% precision, whereas point estimates fail. On GeoQuery (Zelle and Mooney, 1996), a standard semantic parsing dataset, where our model assumptions are violated, we still obtain 100% precision. We were able to reach 70% recall on recovering predicates and 59% on full logical forms.

source atoms	target atoms
{*area*, *of*, *Iowa*}	{`area`, `IA`}
{*cities*, *in*, *Ohio*}	{`city`, `OH`}
{*cities*, *in*, *Iowa*}	{`city`, `IA`}

mapping 1	mapping 2	mapping 3	mapping 4
cities → {`city`}	*cities* → {}	*cities* → {`city`}	*cities* → {}
in → {}	*in* → {`city`}	*in* → {}	*in* → {`city`}
of → {}	*of* → {}	*of* → {`area`}	*of* → {`area`}
area → {`area`}	*area* → {`area`}	*area* → {}	*area* → {}
Iowa → {`IA`}	*Iowa* → {`IA`}	*Iowa* → {`IA`}	*Iowa* → {`IA`}
Ohio → {`OH`}	*Ohio* → {`OH`}	*Ohio* → {`OH`}	*Ohio* → {`OH`}

Figure 2: Given the training examples in the top table, there are exactly four mappings consistent with these training examples.

2 Setup

We represent an input x (e.g., *area of Ohio*) as a bag (multiset) of *source atoms* and an output y (e.g., `area(OH)`) as a bag of *target atoms*. In the simplest case, *source atoms* are words and *target atoms* are predicates—see Figure 2(top) for an example.[2] We assume there is a true mapping M^* from a source atom s (e.g., *Ohio*) to a bag of target atoms $t = M^*(s)$ (e.g., $\{\texttt{OH}\}$). Note that M^* can also map a source atom s to no target atoms ($M^*(\textit{of}) = \{\}$) or multiple target atoms ($M^*(\textit{grandparent}) = \{\texttt{parent}, \texttt{parent}\}$). We extend M^* to bag of source atoms via multiset sum: $M^*(x) = \uplus_{s \in x} M^*(s)$.

Of course, we do not know M^* and must estimate it from training data. Our training examples are input-output pairs $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. For now, we assume that there is no noise so that $y_i = M^*(x_i)$; Section 3.4 shows how to deal with noise. Our goal is to output a *mapping* $\hat{M}$ that maps each input x to either a bag of target atoms or "don't know." We say that $\hat{M}$ has 100% precision if $\hat{M}(x) = M^*(x)$ whenever $\hat{M}(x)$ is not "don't know." The chief difficulty is that the source atoms x_i and the target atoms y_i are unaligned. While we could try to infer the alignment, we will show that it is unnecessary for obtaining 100% precision.

2.1 Unanimity principle

Let $\mathcal{M}$ be the set of mappings (which contains the true mapping M^*). Let $\mathcal{C}$ be the subset of map-

[2] Our semantic parsing experiments (Section 4.2) use more complex source and target atoms to capture some context and structure.

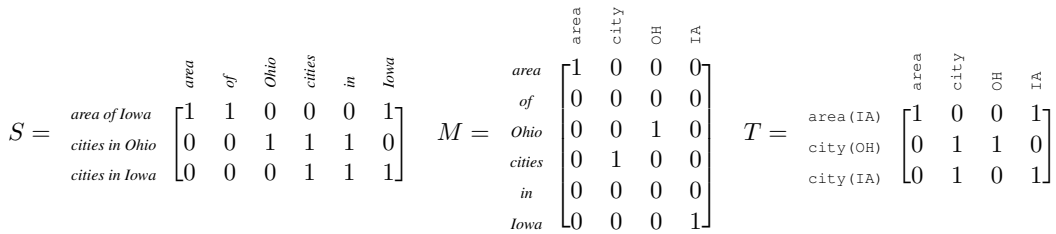


Figure 3: Our training data encodes a system of linear equations $SM = T$, where the rows of S are inputs, the rows of T are the corresponding outputs, and M specifies the mapping between source and target atoms.

pings consistent with the training examples.

$$\mathcal{C} \stackrel{\text{def}}{=} \{M \in \mathcal{M} \mid M(x_i) = y_i, \forall i = 1, \ldots, n\} \tag{1}$$

Figure 2 shows the four mappings consistent with the training set in our running example. Let $\mathcal{F}$ be the set of *safe inputs*, those on which all mappings in $\mathcal{C}$ agree:

$$\mathcal{F} \stackrel{\text{def}}{=} \{x : |\{M(x) : M \in \mathcal{C}\}| = 1\}. \tag{2}$$

The *unanimity principle* defines a mapping $\hat{M}$ that returns the unanimous output on $\mathcal{F}$ and "don't know" on its complement. This choice obtains the following strong guarantee:

Proposition 1. *For each safe input $x \in \mathcal{F}$, we have $\hat{M}(x) = M^*(x)$. In other words, M obtains 100% precision.*

Furthermore, $\hat{M}$ obtains the best possible recall given this model family subject to 100% precision, since for any $x \notin \mathcal{F}$ there are at least two possible outputs generated by consistent mappings, so we cannot safely guess one of them.

3 Linear algebraic formulation

To solve the learning problem laid out in the previous section, let us recast the problem in linear algebraic terms. Let n_s (n_t) be the number of source (target) atom types. First, we can represent the bag x (y) as a n_s-dimensional (n_t-dimensional) row vector of counts; for example, the vector form of "*area of Ohio*" is $\begin{matrix}\textit{area} & \textit{of} & \textit{Ohio} & \textit{cities} & \textit{in} & \textit{Iowa}\end{matrix}$ $[1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0]$.

We represent the mapping M as a non-negative integer-valued matrix, where M_{st} is the number of times target atom t appears in the bag that source atom s maps to (Figure 3). We also encode the n training examples as matrices: S is an $n \times n_\text{s}$ matrix where the i-th row is x_i; T as an $n \times n_\text{t}$ matrix where the i-th row is y_i. Given these matrices, we can rewrite the set of consistent mappings (2) as:

$$\mathcal{C} = \{M \in \mathbb{Z}_{\geq 0}^{n_\text{s} \times n_\text{t}} : SM = T\}. \tag{3}$$

See Figure 3 for the matrix formulation of S and T, along with one possible consistent mapping M for our running example.

3.1 Integer linear programming

Finding an element of $\mathcal{C}$ as defined in (3) corresponds to solving an integer linear program (ILP), which is NP-hard in the worst case, though there exist relatively effective off-the-shelf solvers such as Gurobi. However, *one* solution is not enough. To check whether an input x is in the safe set $\mathcal{F}$ (2), we need to check whether *all* mappings $M \in \mathcal{C}$ predict the same output on x; that is, xM is the same for all $M \in \mathcal{C}$.

Our insight is that we can check whether $x \in \mathcal{F}$ by solving just two ILPs. Recall that we want to know if the output vector xM can be different for different $M \in \mathcal{C}$. To do this, we pick a random vector $v \in \mathbb{R}^{n_\text{t}}$, and consider the scalar projection xMv. The first ILP maximizes this scalar and the second one minimizes it. If both ILPs return the same value, then with probability 1, we can conclude that xM is the same for all mappings $M \in \mathcal{C}$ and thus $x \in \mathcal{F}$. The following proposition formalizes this:

Proposition 2. *Let x be any input. Let $v \sim \mathcal{N}(0, I_{n_t \times n_t})$ be a random vector. Let $a = \min_{M \in \mathcal{C}} xMv$ and $b = \max_{M \in \mathcal{C}} xMv$. With probability 1, $a = b$ iff $x \in \mathcal{F}$.*

Proof. If $x \in \mathcal{F}$, there is only one output xM, so $a = b$. If $x \notin \mathcal{F}$, there exists two $M_1, M_2 \in \mathcal{C}$ for which $xM_1 \neq xM_2$. Then $w \stackrel{\text{def}}{=} x(M_1 -$

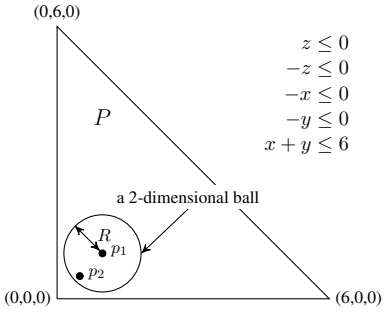


Figure 4: Our goal is to find two points p_1, p_2 in the relative interior of a polytope P defined by inequalities shown on the right. The inequalities $z \le 0$ and $-z \le 0$ are always active. Therefore, P is a 2-dimensional polytope. One solution to the LP (6) is $\alpha^* = 1, p^* = (1,1,0), \xi^{*\top} = [0,0,1,1,1]$, which results in $p_1 = (1,1,0)$ with $R = 1/\sqrt{2}$. The other point p_2 is chosen randomly from the ball of radius R.

$M_2) \in \mathbb{R}^{1 \times n_t}$ is nonzero. The probability of $wv = 0$ is zero because the space orthogonal to w is a $(n_t - 1)$-dimensional space while v is drawn from a n_t-dimensional space. Therefore, with probability 1, $xM_1v \neq xM_2v$. Without loss of generality, $a \le xM_1v < xM_2v \le b$, so $a \neq b$. □

3.2 Linear programming

Proposition 2 requires solving two non-trivial ILPs per input *at test time*. A natural step is to relax the integer constraint so that we solve two LPs instead.

$$\mathcal{C}_{\text{LP}} \stackrel{\text{def}}{=} \{M \in \mathbb{R}^{n_s \times n_t}_{\ge 0} \mid SM = T\} \quad (4)$$

$$\mathcal{F}_{\text{LP}} \stackrel{\text{def}}{=} \{x : |\{M(x) : M \in \mathcal{C}_{\text{LP}}\}| = 1\}. \quad (5)$$

The set of consistent mappings is larger ($\mathcal{C}_{\text{LP}} \supseteq \mathcal{C}$), so the set of safe inputs is smaller ($\mathcal{F}_{\text{LP}} \subseteq \mathcal{F}$). Therefore, if we predict only on $\mathcal{F}_{\text{LP}}$, we still maintain 100% precision, although the recall could be lower.

Now we will show how to exploit the convexity of $\mathcal{C}_{\text{LP}}$ (unlike $\mathcal{C}$) to avoid solving any LPs at test time at all. The basic idea is that if we choose two mappings $M_1, M_2 \in \mathcal{C}_{\text{LP}}$ "randomly enough", whether $xM_1 = xM_2$ is equivalent to unanimity over $\mathcal{C}_{\text{LP}}$. We could try to sample M_1, M_2 uniformly from $\mathcal{C}_{\text{LP}}$, but this is costly. We instead show that "less random" choice suffices. This is formalized as follows:

Proposition 3. *Let X be a finite set of test inputs. Let d be the dimension of $\mathcal{C}_{\text{LP}}$. Let M_1 be any mapping in $\mathcal{C}_{\text{LP}}$, and let vec(M_2) be sampled from a proper density over a d-dimensional ball lying in $\mathcal{C}_{\text{LP}}$ centered at vec(M_1). Then, with probability 1, for all $x \in X$, $xM_1 = xM_2$ implies $x \in \mathcal{F}_{\text{LP}}$.*

Proof. We will prove the contrapositive. If $x \notin \mathcal{F}_{\text{LP}}$, then xM is not the same for all $M \in \mathcal{C}_{\text{LP}}$. Without loss of generality, assume not all $M \in \mathcal{C}_{\text{LP}}$ agree on the i-th component of xM. Note that $(xM)_i = \text{tr}(Me_ix)$, which is the inner product of vec(M) and vec(e_ix). Since $(xM)_i$ is not the same for all $M \in \mathcal{C}_{\text{LP}}$ and $\mathcal{C}_{\text{LP}}$ is convex, the projection of $\mathcal{C}_{\text{LP}}$ onto vec(e_ix) must be a one-dimensional polytope. For both vec(M_1) and vec(M_2) to have the same projection on vec(e_ix), they would have to both lie in a $(d-1)$-dimensional polytope orthogonal to vec(e_ix). Since vec(M_2) is sampled from a proper density over a d-dimensional ball, this has probability 0. □

Algorithm. We now provide an algorithm to find two points p_1, p_2 inside a general d-dimensional polytope $P = \{p : Ap \le b\}$ satisfying the conditions of Proposition 3, where for clarity we have simplified the notation from vec(M_i) to p_i and $\mathcal{C}_{\text{LP}}$ to P.

We first find a point p_1 in the relative interior of P, which consists of points for which the fewest number of inequalities j are active (i.e., $a_jp = b_j$). We can achieve this by solving the following LP from Freund et al. (1985):

$$\max \mathbf{1}^\top \xi \text{ s.t. } Ap + \xi \le \alpha b, 0 \le \xi \le \mathbf{1}, \alpha \ge 1. \quad (6)$$

Here, ξ_j is a lower bound on the slack of inequality j, and α scales up the polytope so that all the ξ_j that can be positive are exactly 1 in the optimum solution. Importantly, if $\xi_j = 0$, constraint j is *always active* for all solutions $p \in P$. Let (p^*, ξ^*, α^*) be an optimal solution to the LP. Then define A_1 as the submatrix of A containing rows j for which $\xi^*_j = 1$, and A_0 consist of the remaining rows for which $\xi^*_j = 0$.

The above LP gives us $p_1 = p^*/\alpha^*$, which lies in the relative interior of P (see Figure 4). To obtain p_2, define a radius $R \stackrel{\text{def}}{=} (\alpha \max_{j:\xi^*_j=1} \|a_j\|_2)^{-1}$. Let the columns of matrix N form an orthonormal basis of the null space of A_0. Sample v from a unit d-dimensional ball centered at 0, and set $p_2 = p_1 + RNv$.

To show that $p_2 \in P$: First, p_2 satisfies the always-active constraints j, $a_j^\top(p_1 + RNv) = b_j$,

Algorithm 1 Our linear programming approach.

procedure TRAIN
Input: Training examples
Output: Generic mappings (M_1, M_2)
  Define $\mathcal{C}_{\text{LP}}$ as explained in (4).
  Compute M_1 and a radius R by solving an LP (6).
  Sample M_2 from a ball with radius R around M_1.
  return (M_1, M_2)
end procedure

procedure TEST
Input: input x, mappings (M_1, M_2)
Output: A guaranteed correct y or "don't know"
  Compute $y_1 = xM_1$ and $y_2 = xM_2$.
  if $y_1 = y_2$ **then return** y_1
  else return "don't know"
  end if
end procedure

by definition of null space. For non-active j, the LP ensures that $a_j^\top p_1 + \alpha^{-1} \leq b_j$, which implies $a_j^\top (p_1 + RNv) \leq b_j$.

Algorithm 1 summarizes our overall procedure: At training time, we solve a single LP (6) and draw a random vector to obtain M_1, M_2 satisfying Proposition 3. At test time, we simply apply M_1 and M_2, which scales only linearly with the number of source atoms in the input.

3.3 Linear system

To obtain additional intuition about the unanimity principle, let us relax $\mathcal{C}_{\text{LP}}$ (4) further by removing the non-negativity constraint, which results in a linear system. Define the relaxed set of consistent mappings to be all the solutions to the linear system and the relaxed safe set accordingly:

$$\mathcal{C}_{\text{LS}} \stackrel{\text{def}}{=} \{M \in \mathbb{R}^{n_{\text{s}} \times n_{\text{t}}} \mid SM = T\} \quad (7)$$

$$\mathcal{F}_{\text{LS}} \stackrel{\text{def}}{=} \{x : |\{M(x) : M \in \mathcal{C}_{\text{LS}}\}| = 1\}. \quad (8)$$

Note that $\mathcal{C}_{\text{LS}}$ is an affine subspace, so each $M \in \mathcal{C}_{\text{LS}}$ can be expressed as $M_0 + BA$, where M_0 is an arbitrary solution, B is a basis for the null space of S and A is an arbitrary matrix. Figure 5 presents the linear system for four training examples. In the rare case that S has full column rank (if we have many training examples), then the left inverse of S exists, and there is exactly one consistent mapping, the true one ($M^* = S^\dagger T$), but we do not require this.

Let's try to explore the linear algebraic structure in the problem. Intuitively, if we know *area of Ohio* maps to `area(OH)` and *Ohio* maps to `OH`, then we should conclude *area of* maps to `area` by subtracting the second example from the first. The following proposition formalizes and generalizes this intuition by characterizing the relaxed safe set:

Proposition 4. *The vector x is in row space of S iff $x \in \mathcal{F}_{\text{LS}}$.*

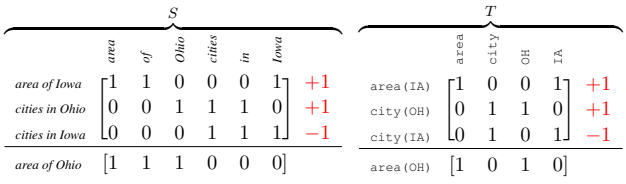


Figure 6: Under the linear system relaxation, we can predict the target atoms for the new input *area of Ohio* by adding and subtracting training examples (rows of S and T).

Proof. If x is in the row space of S, we can write x as a linear combination of S for some coefficients $\alpha \in \mathbb{R}^n$: $x = \alpha^\top S$. Then for all $M \in \mathcal{C}_{\text{LS}}$, we have $SM = T$, so $xM = \alpha^\top SM = \alpha^\top T$, which is the unique output[3] (See Figure 6). If $x \in \mathcal{F}_{\text{LS}}$ is safe, then there exists a y such that for all $M \in \mathcal{C}_{\text{LS}}$, $xM = y$. Recall that each element of $\mathcal{C}_{\text{LS}}$ can be decomposed into $M_0 + BA$. For $x(M_0 + BA)$ to be the same for each A, x should be orthogonal to each column of B, a basis for the null space of S. This means that x is in the row space of S. □

Intuitively, this proposition says that stitching new inputs together by adding and subtracting existing training examples (rows of S) gives you exactly the relaxed safe set $\mathcal{F}_{\text{LS}}$.

Note that relaxations increases the set of consistent mappings ($\mathcal{C}_{\text{LS}} \supseteq \mathcal{C}_{\text{LP}} \supseteq \mathcal{C}$), which has the contravariant effect of shrinking the safe set ($\mathcal{F}_{\text{LS}} \subseteq \mathcal{F}_{\text{LP}} \subseteq \mathcal{F}$). Therefore, using the relaxation (predicting when $x \in \mathcal{F}_{\text{LS}}$) still preserves 100% precision.

3.4 Handling noise

So far, we have assumed that our training examples are noiseless, so that we can directly add the

[3] There might be more than one set of coefficients (α_1, α_2) for writing x. However, they result to a same output: $\alpha_1^\top S = \alpha_2^\top S \implies \alpha_1^\top SM = \alpha_2^\top SM \implies \alpha_1^\top T = \alpha_2^\top T$.

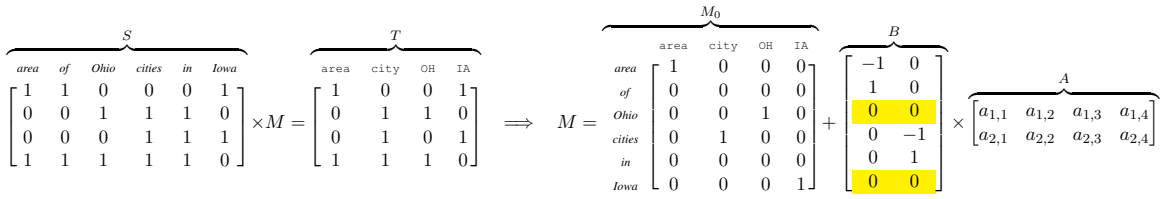


Figure 5: Under the linear system relaxation, all solutions M to $SM = T$ can be expressed as $M = M_0 + BA$, where B is the basis for the null space of S and A is arbitrary. Rows s of B which are zero (*Ohio* and *Iowa*) correspond to the safe source atoms (though not the only safe inputs).

constraint $SM = T$. Now assume that an adversary has made at most n_{mistakes} additions to and deletions of target atoms across the examples in T, but of course we do not know which examples have been tainted. Can we still guarantee 100% precision?

The answer is yes for the ILP formulation: we simply replace the exact match condition ($SM = T$) with a weaker one: $\|SM - T\|_1 \leq n_{\text{mistakes}}$ (*). The result is still an ILP, so the techniques from Section 3.1 readily apply. Note that as n_{mistakes} increases, the set of candidate mappings grows, which means that the safe set shrinks.

Unfortunately, this procedure is degenerate for linear programs. If the constraint (*) is not tight, then $M+E$ also satisfies the constraint for any matrix E of small enough norm. This means that the consistent mappings $\mathcal{C}_{\text{LP}}$ will be full-dimensional and certainly not be unanimous on any input.

Another strategy is to remove examples from the dataset if they could be potentially noisy. For each training example i, we run the ILP (*) on all but the i-th example. If the i-th example is not in the resulting safe set (2), we remove it. This procedure produces a noiseless dataset, on which we can apply the noiseless linear program or linear system from the previous sections.

4 Experiments

4.1 Artificial data

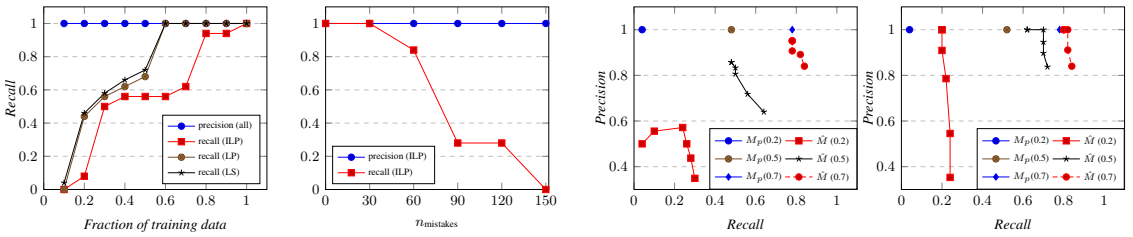
We generated a true mapping M^* from 50 source atoms to 20 target atoms so that each source atom maps to 0–2 target atoms. We then created 120 training examples and 50 test examples, where the length of every input is between 5 and 10. The source atoms are divided into 10 clusters, and each input only contains source atoms from one cluster.

Figure 7a shows the results for $\mathcal{F}$ (integer linear programming), $\mathcal{F}_{\text{LP}}$ (linear programming), and $\mathcal{F}_{\text{LS}}$ (linear system). All methods attain 100% precision, and as expected, relaxations lead to lower recall, though they all can reach 100% recall given enough data.

Comparison with point estimation. Recall that the unanimity principle $\hat{M}$ reasons over the entire set of consistent mappings, which allows us to be robust to changes in the input distribution, e.g., from training set attacks (Mei and Zhu, 2015). As an alternative, consider computing the point estimate M_p that minimizes $\|SM - T\|_2^2$ (the solution is given by $M_p = S^\dagger T$). The point estimate, by minimizing the average loss, implicitly assumes i.i.d. examples. To generate output for input x we compute $y = xM_p$ and round each coordinate y_t to the closest integer. To obtain a precision-recall tradeoff, we set a threshold ϵ and if for all target atoms t, the interval $[y_t - \epsilon, y_t + \epsilon)$ contains an integer, we set y_t to that integer; otherwise we report "don't know" for input x.

To compare unanimous prediction $\hat{M}$ and point estimation M_p, for each $f \in \{0.2, 0.5, 0.7\}$, we randomly generate 100 subsampled datasets consisting of an f fraction of the training examples. For M_p, we sweep ϵ across $\{0.0, 0.1, \ldots, 0.5\}$ to obtain a ROC curve. In Figure 7c(left/right), we select the distribution that results in the maximum/minimum difference between $F_1(\hat{M})$ and $F_1(M_p)$ respectively. As shown, $\hat{M}$ has always 100% precision, while M_p can obtain less 100% precision over its full ROC curve. An adversary can only hurt the recall of unanimous prediction.

Noise. As stated in Section 3.4, our algorithm has the ability to guarantee 100% precision even when the adversary can modify the outputs. As we increase the number of predicate additions/deletions (n_{mistakes}), Figure 7b shows that precision remains at 100%, while recall naturally decreases in response to being less confident about



(a) All the relaxations reach 100% recall; relaxation results in slightly slower convergence.

(b) Size of the safe set shrinks with increasing number of mistakes in the training data.

(c) Performance of the point estimate (M_p) and unanimous prediction ($\hat{M}$) when the inputs are chosen adversarially for M_p (left) and for $\hat{M}$ (right).

Figure 7: Our algorithm always obtains 100% precision with (a) different amounts of training examples and different relaxations, (b) existence of noise, and (c) adversarial input distributions.

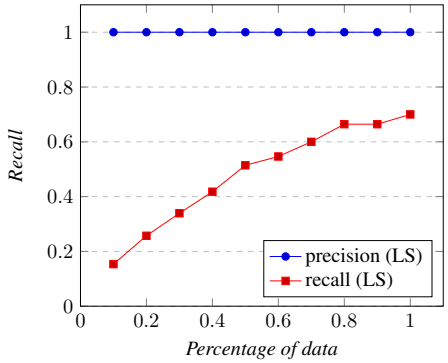


Figure 8: We maintain 100% precision while recall increases with the number of training examples.

the training outputs.

4.2 Semantic parsing on GeoQuery

We now evaluate our approach on the standard GeoQuery dataset (Zelle and Mooney, 1996), which contains 880 utterances and their corresponding logical forms. The utterances are questions related to the US geography, such as: "*what river runs through the most states*". We use the standard 600/280 train/test split (Zettlemoyer and Collins, 2005). After replacing entity names by their types[4] based on the standard entity lexicon, there are 172 different words and 57 different predicates in this dataset.

Handling context. Some words are polysemous in that they map to two predicates: in "*largest river*" and "*largest city*", the word *largest* maps to `longest` and `biggest`, respectively. Therefore, instead of using words as source atoms, we use bigrams, so that each source atom always maps to the same target atoms.

Reconstructing the logical form. We define target atoms to include more information than just the predicates, which enables us to reconstruct logical forms from the predicates. We use the variable-free functional logical forms (Kate et al., 2005), in which each target atom is a predicate conjoined with its argument order (e.g., `loc_1` or `loc_2`). Table 1 shows two different choices of target atoms. At test time, we search over all possible "compatible" ways of combining target atoms into logical forms. If there is exactly one, then we return that logical form and abstain otherwise. We call a predicate combination "compatible" if it appears in the training set.

We put a "null" word at the end of each sentence, and collapsed the `loc` and `traverse` predicates. To deal with noise, we minimized $\|SM - T\|_1$ over real-valued mappings and removed any example (row) with non-zero residual. We perform all experiments using the linear system relaxation. Training takes under 30 seconds.

Figure 8 shows precision and recall as a function of the number of the training examples. We obtain 70% recall over predicates on the test examples. 84% of these have a unique compatible way of combining target atoms into a logical form, which results in a 59% recall on logical forms.

Though our modeling assumptions are incorrect for real data, we were still able to get 100% precision for all training examples. Interestingly, the linear system (which allows negative mappings) helps model GeoQuery dataset better than the linear program (which has a non-negativity constraint). There exists a predicate `all:e` in GeoQuery that is in every sentence unless the ut-

[4]If an entity name has more than one type we replace it by concatenating all of its possible types.

utterances	logical form (A)	target atoms (A)	logical form (B)	target atoms (B)
cities traversed by the Columbia	`city(x),loc(x,Columbia)`	`city,loc,Columbia`	`city(loc_1(Columbia))`	`city,loc_1,Columbia`
cities of Texas	`city(x),loc(Texas,x)`	`city,loc,Texas`	`city(loc_2(Texas))`	`city,loc_2,Texas`

Table 1: Two different choices of target atoms: (A) shows predicates and (B) shows predicates conjoined with their argument position. (A) is sufficient for simply recovering the predicates, whereas (B) allows for logical form reconstruction.

terance contains a proper noun. With negative mappings, *null* maps to `all:e`, while each proper noun maps to its proper predicate *minus* `all:e`.

There is a lot of work in semantic parsing that tackles the GeoQuery dataset (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Kwiatkowski et al., 2010; Liang et al., 2011), and the state-of-the-art is 91.1% precision and recall (Liang et al., 2011). However, none of these methods can guarantee 100% precision, and they perform more feature engineering, so these numbers are not quite comparable. In practice, one could use our unanimous prediction approach in conjunction with others: For example, one could run a classic semantic parser and simply certify 59% of the examples to be correct with our approach. In critical applications, one could use our approach as a first-pass filter, and fall back to humans for the abstentions.

5 Extensions

5.1 Learning from denotations

Up until now, we have assumed that we have input-output pairs. For semantic parsing, this means annotating sentences with logical forms (e.g., *area of Ohio* to `area(OH)`) which is very expensive. This has motivated previous work to learn from question-answer pairs (e.g., *area of Ohio* to `44825`) (Liang et al., 2011). This provides weaker supervision: For example, `44825` is the area of Ohio (in squared miles), but it is also the zip code of Chatfield. So, the true output could be either `area(OH)` or `zipcode(Chatfield)`.

In this section, we show how to handle this form of weak supervision by asking for unanimity over additional selection variables. Formally, we have $D = \{(x_1, Y_1), \ldots, (x_n, Y_n)\}$ as a set of training examples, here each Y_i consists of k_i candidate outputs for x_i. In this case, the unknowns are the mapping M as before along with a selection vector π_i, which specifies which of the k_i outputs in Y_i is equal to $x_i M$. To implement the unanimity principle, we need to consider the set of all consistent solutions (M, π).

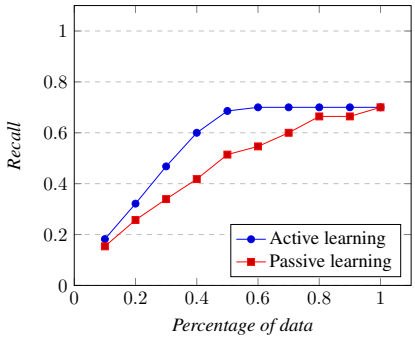


Figure 9: When we choose examples to be linearly independent, we only need half the number of examples to achieve the same performance.

We construct an integer linear program as follows: Each training example adds a constraint that the output of it should be exactly one of its candidate output. For the i-th example, we form a matrix $T_i \in \mathbb{R}^{k_i \times n_t}$ with all the k_i candidate outputs. Formally we want $x_i M = \pi_i T_i$. The entire ILP is:

$$\begin{aligned} &\forall i,\ x_i M = \pi_i T_i \\ &\forall i,\ \textstyle\sum_j \pi_{ij} = 1 \\ &\pi, M \geq 0 \end{aligned}$$

Given a new input x, we return the same output if xM is same for all consistent solutions (M, π). Note that we can effectively "marginalize out" π. We can also relax this ILP into an linear program following Section 3.2.

5.2 Active learning

A side benefit of the linear system relaxation (Section 3.3) is that it suggests an active learning procedure. The setting is that we are given a set of inputs (the matrix S), and we want to (adaptively) choose which inputs (rows of S) to obtain the output (corresponding row of T) for.

Proposition 4 states that under the linear system formulation, the set of safe inputs $\mathcal{F}_{\text{LS}}$ is exactly the same as the row space of S. Therefore, if we ask for an input that is already in the row space of S, this will not affect $\mathcal{F}_{\text{LS}}$ at all. The algo-

rithm is then simple: go through our training inputs $x_1, \ldots, x_n$ one by one and ask for the output only if it is not in the row space of the previously-added inputs $x_1, \ldots, x_{i-1}$.

Figure 9 shows the recall when we choose examples to be linearly independent in this way in comparison to when we choose examples randomly. The active learning scheme requires half as many labeled examples as the passive scheme to reach the same recall. In general, it takes $\text{rank}(S) \leq n$ examples to obtain the same recall as having labeled all n examples. Of course, the precision of both systems is 100%.

5.3 Paraphrasing

Another side benefit of the linear system relaxation (Section 3.3) is that we can easily partition the safe set $\mathcal{F}_{\text{LS}}$ (8) into subsets of utterances which are paraphrases of each other. Two utterances are paraphrase of each other if both map to the same logical form, e.g., "*Texas's capital*" and "*capital of Texas*". Given a sentence $x \in \mathcal{F}_{\text{LS}}$, our goal is to find all of its paraphrases in $\mathcal{F}_{\text{LS}}$.

As explained in Section 3.3, we can represent each input x as a linear combination of S for some coefficients $\alpha \in \mathbb{R}^n$: $x = \alpha^\top S$. We want to find all $x' \in \mathcal{F}_{\text{LS}}$ such that x' is guaranteed to map to the same output as x. We can represent $x' = \beta^\top S$ for some coefficients $\beta \in \mathbb{R}^n$. The outputs for x and x' are thus $\alpha^\top T$ and $\beta^\top T$, respectively. Thus we are interested in β's such that $\alpha^\top T = \beta^\top T$, or in other words, $\alpha - \beta$ is in the null space of $T^\top$. Let B be a basis for the null space of $T^\top$. We can then write $\alpha - \beta = Bv$ for some v. Therefore, the set of paraphrases of $x \in \mathcal{F}_{\text{LS}}$ are:

$$\text{Paraphrases}(x) \stackrel{\text{def}}{=} \{(\alpha - Bv)^\top S : v \in \mathbb{R}^n\}. \quad (9)$$

6 Discussion and related work

Our work is motivated by the semantic parsing task (though it can be applied to any set-to-set prediction task). Over the last decade, there has been much work on semantic parsing, mostly focusing on learning from weaker supervision (Liang et al., 2011; Goldwasser et al., 2011; Artzi and Zettlemoyer, 2011; Artzi and Zettlemoyer, 2013), scaling up beyond small databases (Cai and Yates, 2013; Berant et al., 2013; Pasupat and Liang, 2015), and applying semantic parsing to other tasks (Matuszek et al., 2012; Kushman and Barzilay, 2013; Artzi and Zettlemoyer, 2013). However, only Popescu et al. (2003) focuses on precision. They also obtain 100% precision, but with a hand-crafted system, whereas we *learn* a semantic mapping.

The idea of computing consistent hypotheses appears in the classic theory of version spaces for binary classification (Mitchell, 1977) and has been extended to more structured settings (Vanlehn and Ball, 1987; Lau et al., 2000). Our version space is used in the context of the unanimity principle, and we explore a novel linear algebraic structure. Our "safe set" of inputs appears in the literature as the complement of the disagreement region (Hanneke, 2007). They use this notion for active learning, whereas we use it to support unanimous prediction.

There is classic work on learning classifiers that can abstain (Chow, 1970; Tortorella, 2000; Balsubramani, 2016). This work, however, focuses on the classification setting, whereas we considered more structured output settings (e.g., for semantic parsing). Another difference is that we operate in a more adversarial setting by leaning on the unanimity principle.

Another avenue for providing user confidence is probabilistic calibration (Platt, 1999), which has been explored more recently for structured prediction (Kuleshov and Liang, 2015). However, these methods do not guarantee precision for *any* training set and test input.

In summary, we have presented the unanimity principle for guaranteeing 100% precision. For the task of learning semantic mappings, we leveraged the linear algebraic structure in our problem to make unanimous prediction efficient. We view our work as a first step in learning reliable semantic parsers. A natural next step is to explore our framework with additional modeling improvements—especially in dealing with context, structure, and noise.

Reproducibility. All code, data, and experiments for this paper are available on the CodaLab platform at `https://worksheets.codalab.org/worksheets/0x593676a278fc4e5abe2d8bac1e3df486/`.

Acknowledgments. We would like to thank the anonymous reviewers for their helpful comments. We are also grateful for a Future Of Life Research Award and NSF grant CCF-1138967, which supported this work.

References

Y. Artzi and L. Zettlemoyer. 2011. Bootstrapping semantic parsers from conversations. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 421–432.

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

A. Balsubramani. 2016. Learning to abstain from binary prediction. *arXiv preprint arXiv:1602.08151*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL)*.

C. K. Chow. 1970. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46.

R. M. Freund, R. Roundy, and M. J. Todd. 1985. Identifying the set of always-active constraints in a system of linear inequalities by a single linear program. Technical report, Massachusetts Institute of Technology, Alfred P. Sloan School of Management.

D. Goldwasser, R. Reichart, J. Clarke, and D. Roth. 2011. Confidence driven unsupervised semantic parsing. In *Association for Computational Linguistics (ACL)*, pages 1486–1495.

S. Hanneke. 2007. A bound on the label complexity of agnostic active learning. In *International Conference on Machine Learning (ICML)*, pages 353–360.

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1062–1068.

V. Kuleshov and P. Liang. 2015. Calibrated structured prediction. In *Advances in Neural Information Processing Systems (NIPS)*.

N. Kushman and R. Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Human Language Technology and North American Association for Computational Linguistics (HLT/NAACL)*, pages 826–836.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. A. Lau, P. Domingos, and D. S. Weld. 2000. Version space algebra and its application to programming by demonstration. In *International Conference on Machine Learning (ICML)*, pages 527–534.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*, pages 590–599.

C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*, pages 1671–1678.

S. Mei and X. Zhu. 2015. Using machine teaching to identify optimal training-set attacks on machine learners. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

T. M. Mitchell. 1977. Version spaces: A candidate elimination approach to rule learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 305–310.

B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. Tygar, and K. Xia. 2009. Misleading learners: Co-opting your spam filter. In *Machine learning in cyber trust*, pages 17–51.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

J. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 10(3):61–74.

A. Popescu, O. Etzioni, and H. Kautz. 2003. Towards a theory of natural language interfaces to databases. In *International Conference on Intelligent User Interfaces (IUI)*, pages 149–157.

H. Shimodaira. 2000. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244.

F. Tortorella. 2000. An optimal reject rule for binary classifiers. In *Advances in Pattern Recognition*, pages 611–620.

K. Vanlehn and W. Ball. 1987. A version space approach to learning context-free grammars. *Machine learning*, 2(1):39–74.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

Exploring Convolutional and Recurrent Neural Networks in Sequential Labelling for Dialogue Topic Tracking

Seokhwan Kim, Rafael E. Banchs, Haizhou Li
Human Language Technology Department
Institute for Infocomm Research
Singapore 138632
{kims,rembanchs,hli}@i2r.a-star.edu.sg

Abstract

Dialogue topic tracking is a sequential labelling problem of recognizing the topic state at each time step in given dialogue sequences. This paper presents various artificial neural network models for dialogue topic tracking, including convolutional neural networks to account for semantics at each individual utterance, and recurrent neural networks to account for conversational contexts along multiple turns in the dialogue history. The experimental results demonstrate that our proposed models can significantly improve the tracking performances in human-human conversations.

1 Introduction

A human conversation often involves a series of multiple topics contextually related to each other. In this scenario, every participant in the conversation is required to understand the on-going topic discussed at each moment, detect any topic shift made by others, and make a decision to self-initiate a new topic. These human capabilities for handling topics are also expected from dialogue systems to achieve natural and human-like conversations.

Many studies have been conducted on multi-domain or multi-task dialogue systems by means of sentence-level topic identification as a sub-task of natural language understanding (Lin et al., 1999; Nakata et al., 2002; Lagus and Kuusisto, 2002; Adams and Martell, 2008; Ikeda et al., 2008; Celikyilmaz et al., 2011). In these approaches, a given user input at each turn is categorized into topic classes, each of which triggers the corresponding sub-system specializing in the particular topic. Despite many previous efforts, the sentence categorization methods still have the following limitations. Firstly, the effectiveness of the approaches is limited only in user-initiative conversations, because the categorization is performed mainly based on the user's input mentioned at a given turn. Secondly, no correlation between different topics is considered neither in the topic decision process nor in each topic-specific sub-system operated independently from the others. Lastly, the conversational coherence in a given dialogue history sequence has limited effects on determining the current topic.

Another direction for multi-topic dialogue systems has been towards utilizing human knowledge represented in domain models (Roy and Subramaniam, 2006) and agendas (Bohus and Rudnicky, 2003; Lee et al., 2008). The knowledge-based approaches make the system capable of having more control of dialogue flows including topic sequences. This aspect contributes to better decisions of topics in system-initiative cases, but it can adversely affect the flexibility to deal with unexpected inputs against the system's suggestions. Moreover, the high cost of building the required resources is another problem that these methods face from a practical point of view.

Recently, some researchers (Morchid et al., 2014a; Morchid et al., 2014b; Esteve et al., 2015) have worked on topic identification for analyzing human-human dialogues. Although they don't aim at building components in dialogue systems directly, the human behaviours learned from the conversations can suggest directions for further advancement of conversational agents. However, the problem defined in the studies is under the assumption that every dialogue session is assigned with just a single theme category, which means any topic shift occurred in a session is left out of consideration in the analyses.

On the other hand, we previously addressed the problem of detecting multiple topic transitions

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 963–973,
Berlin, Germany, August 7-12, 2016.

in mixed-initiative human-human conversations, which is called dialogue topic tracking (Kim et al., 2014a; Kim et al., 2014b). In these studies, the tracking task is formulated as a classification problem for each utterance-level, similar to the sentence categorization approaches. But the target of the classification is not just an individual topic category to which each input sentence belongs, but the decision whether a topic transition occurs at a given turn as well as what the most probable topic category will follow after the transition.

This paper presents our work also on dialogue topic tracking mainly focusing on the following issues. Firstly, in addition to transitions between dialogue segments from different topics, transitions between segments belonging to the same topic are also detected. This focuses the task more on detailed aspects of topic handling that are relevant to other subtasks such as natural language understanding and dialogue state tracking, rather than the conventional tracking of changes in topic categories only. Another contribution of this work is that we introduce a way to use convolutional neural networks in topic tracking to improve the classification performances with the learned convolutional features. In addition, we also propose the architectures based on recurrent neural networks to incorporate the temporal coherence that has not played an important role in previous approaches.

The remainder of this paper is structured as follows. We present a problem definition of dialog topic tracking in Section 2. We describe our proposed approaches to this task using convolutional and recurrent neural networks in Section 3. We report the evaluation result of the methods in Section 4 and conclude this paper in Section 5.

2 Dialogue Topic Tracking

Dialogue topic tracking is defined as a multi-class classification problem to categorize the topic state at each time step into the labels encoded in BIO tagging scheme (Ramshaw and Marcus, 1995) as follows:

$$f(t) = \begin{cases} \text{B-}\{c \in C\} & \text{if } u_t \text{ is at the beginning} \\ & \text{of a segment belongs to } c, \\ \text{I-}\{c \in C\} & \text{else if } u_t \text{ is inside a} \\ & \text{segment belongs to } c, \\ \text{O} & \text{otherwise,} \end{cases}$$

where u_t is the t-th utterance in a given dialogue session and C is a closed set of topic categories.

t	Speaker	Utterance (u_t)	$f(t)$
1	Guide	How can I help you?	B-OPEN
2	Tourist	Can you recommend some good places to visit in Singapore?	B-ATTR
3	Guide	Well if you like to visit an icon of Singapore, Merlion will be a nice place to visit.	I-ATTR
4	Tourist	Okay. But I'm particularly interested in amusement parks.	B-ATTR
5	Guide	Then, what about Universal Studio?	I-ATTR
6	Tourist	Good! How can I get there from Orchard Road by public transportation?	B-TRSP
7	Guide	You can take the red line train from Orchard and transfer to the purple line at Dhoby Ghaut. Then, you could reach HarbourFront where Sentosa Express departs.	I-TRSP
8	Tourist	How long does it take in total?	I-TRSP
9	Guide	It'll take around half an hour.	I-TRSP
10	Tourist	Alright.	I-TRSP
11	Guide	Or, you can use the shuttle bus service from the hotels in Orchard, which is free of charge.	B-TRSP
12	Tourist	Great! That would be definitely better.	I-TRSP
13	Guide	After visiting the park, you can enjoy some seafoods at the riverside on the way back.	B-FOOD
14	Tourist	What food do you have any recommendations to try there?	I-FOOD
15	Guide	If you like spicy foods, you must try chilli crab which is one of our favourite dishes.	I-FOOD
16	Tourist	Great! I'll try that.	I-FOOD

Figure 1: Examples of dialogue topic tracking on a tour guide dialogue labelled with BIO tags. ATTR, TRSP and FOOD denotes the topic categories of attraction, transportation, and food, respectively.

Figure 1 shows an example of topic tracking on a dialogue fragment between a tour guide and a tourist. Since each tag starting with 'B' should occur at the beginning of a new segment after a topic transition from its previous one, the label sequence indicates that this conversation is divided into six segments at $t = \{2, 4, 6, 11, 13\}$. The initiativity of each segment can be also found from who the speaker of the first utterance of the segment is. In this example, three of the cases are initiated by the tourist at $t = \{2, 4, 6\}$, but the others are leaded by the tour guide, which means it is a mixed-initiative type of conversation.

Different from the former studies (Kim et al., 2014a; Kim et al., 2014b) that were only focused on detecting transitions between different topic categories, this work subdivides each dialogue sequence which belongs to a single topic category, but discusses more than one subject that can be more specifically differentiated from each other. The above example also has two cases of transitions with no change of topic categories at $t = \{4, 11\}$: the first one is due to the tourist's request for an alternative attraction from the recommendation in the previous segment, and the other transition is triggered by the tour guide to suggest another option of transportation which is also available for the route discussed previously.

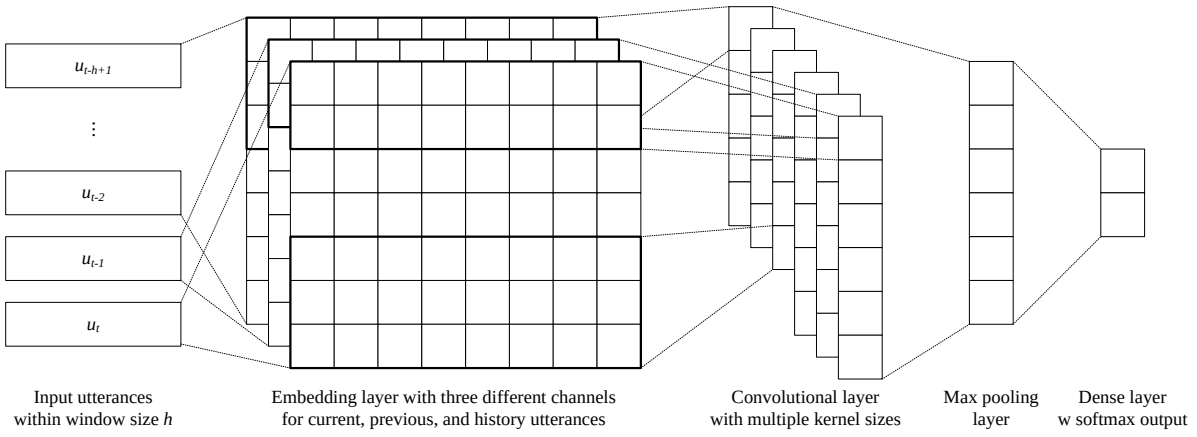


Figure 2: Convolutional neural network architecture for dialogue topic tracking.

3 Models

The classifier f can be built with supervised machine learning techniques, when a set of example dialogues manually annotated with gold standard labels are available as a training set. The earlier studies (Kim et al., 2014a; Kim et al., 2014b) also proposed supervised classification approaches particularly focusing on kernel methods to incorporate domain knowledge obtained from external resources into the linear vector space models based on bag-of-words features extracted from the training dialogues.

This work, on the other hand, aims at improving the classification capabilities only with the internal contents in given dialogues rather than making better uses of external knowledge. To overcome the limitations of the simple vector space models used in the previous work, we propose models based on convolutional and recurrent neural network architectures. These models are presented in the remainder of this section.

3.1 Convolutional Neural Networks

A convolutional neural network (CNN) automatically learns the filters in its convolutional layers which are applied to extract local features from inputs. Then, these lower-level features are combined into higher-level representations following a given network architecture. These aspects of CNNs make themselves a good fit to solve the problems which are invariant to the location where each feature is extracted on its input space and also depend on the compositional relationships between local and global features, which is the reason why CNNs have succeed in computer vision (LeCun et al., 1998). As implied by the successes of bag-of-words or bag-of-ngrams considering the existence of each linguistic unit independently and the important roles of compositional structures in linguistics, CNN models have recently achieved significant improvements also in some natural language processing tasks (Collobert et al., 2011; Shen et al., 2014; Yih et al., 2014; Kalchbrenner et al., 2014a; Kim, 2014).

The model for dialogue topic tracking (Figure 2) is basically based on the CNN architecture proposed by Collobert et al. (2011) and Kim (2014) for sentence classification tasks. In the architecture, a sentence of length n is represented as a matrix with the size of $n \times k$ concatenated with n rows each of which is the k-dimensional word vector $\vec{x}_i \in \mathbb{R}^k$ representing the i-th word in the sentence. This embedding layer can be learned from scratch with random initialization or fine-tuned from pre-trained word vectors (Mikolov et al., 2013) with back propagation during training the network.

Unlike other sentence classification tasks, dialogue topic tracking should consider not only a single sentence given at each time step, but also the other utterances previously mentioned. To incorporate the dependencies to the dialogue history into the topic tracking model, the input at the time step t is composed of three different channels each of which represents the current utterance u_t, the previous utterance u_{t-1}, and the other utterances $u_{t-h+1:t-2}$ within h time steps, respectively, where u_t is the t-th utterance in a session, $u_{i:j}$ is the concatenation of the utterances occurred from the i-th to the j-th time steps in the history, and h is the size of history window. The height of the $n \times k$ matrices of the first two channels for

the current and previous utterances is fixed to the length of the longest utterance in the whole training dataset, and then all the remaining rows after the end of each utterance are zero-padded to make all inputs same size. Since the other channel is made up by concatenating the utterances from the $(t-h+1)$-th to the $(t-2)$-th time steps, it has a matrix with the dimension of $((h-2)\cdot n)\times k$ where all the gaps between contiguous utterances in the matrix are filled with zero.

In the convolutional layer, each filter $\mathcal{F}\in\mathbb{R}^{km}$ which has the same width k as the input matrix and a given window size m as its height slides over from the first row to the $(n-m+1)$-th row of the input matrix. At the i-th position, the filter is applied to generate a feature $c_i = g\left(\mathcal{F}\cdot\vec{x}_{i:i+m-1}+b\right)$, where $\vec{x}_{i:j}$ is the subregion from the i-th row to the j-th row in the input, $b\in\mathbb{R}$ is a bias term, and g is a non-linear activation function such as rectified linear units. This series of convolution operations produces a feature map $\vec{c}=[c_1\cdots c_{n-m+1}]\in\mathbb{R}^{n-m+1}$ for the filter $\mathcal{F}$. Then, the maximum value $c'=\max(\vec{c})$ is selected from each feature map considered as the most important feature for the particular filter in the max-pooilng layer.

Every filter is shared across all the three different channels, but both the convolution and max-pooling operations are performed individually for each channel. Thus, the total number of feature values generated in the pooling layer is three times the number of filters. Finally, these values are forwarded to the fully-connected layer with softmax which generates the probabilistic distribution over the topic labels for a given input.

3.2 Recurrent Neural Networks

Dialogue topic tracking is conceptually performed on a sequence of interactions exchanged by the participants in a given session from its beginning to each turn. Thus, the contents discussed previously in the dialogue history are likely to have an important influence on tracking the current topic at a given turn, which is a fundamental difference from other text categorization problems that consider each input independently from all others.

To make use of the sequential dependencies in dialogue topic tracking, we propose the models based on recurrent neural networks (RNN) which learn the temporal dynamics by recurrent computations applied to every time step in a given input sequence. In a traditional RNN, hidden states connecting between input sequences and output labels are repeatedly updated with the operation $\vec{s_t}=g(Ux_t+Ws_{t-1})$, where x_t is the t-th element in a given input sequence, $\vec{s_t}\in\mathbb{R}^{|s|}$ is the hidden state at t with $|s|$ hidden units, and g is a non-linear activation function. The parameters U and W are shared all the time steps.

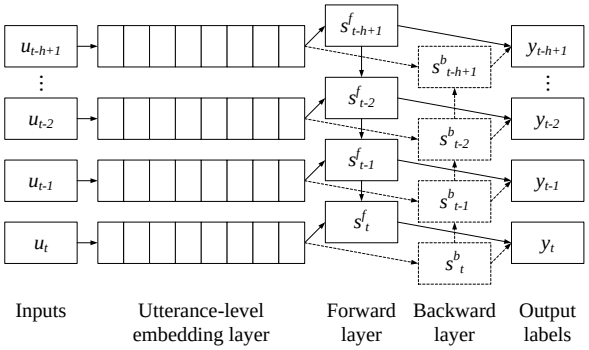


Figure 3: Recurrent neural network architecture for dialogue topic tracking. The backward layer with the dotted lines is enabled only with its bi-directional extension.

RNNs have been successfully applied to several natural language processing tasks including language modeling (Mikolov et al., 2010), text generation (Sutskever et al., 2011), and machine translation (Auli et al., 2013; Liu et al., 2014), all of which focus on dealing with variable length word sequences. On the other hand, an input sequence to be handled in dialogue topic tracking is composed of utterance-level units instead of words.

In our model (Figure 3), each utterance is represented by the k-dimensional vector $\vec{u}_t\in\mathbb{R}^k$ assigned with pre-trained sentence-level embeddings (Le and Mikolov, 2014). And then, a sequence of the utterance vectors within h time steps are connected in the recurrent layers. The default sequence of applying the recurrent operation is the ascending order from the former to the recent utterances, which is performed in the forward layer. But the opposite direction can be also considered in the backward layer which is stacked on top of the forward layer to build a bidirectional RNN (Schuster and Paliwal, 1997) which outputs the concatenation of both forward and backward states as an outcome of the recurrent operations. Then, these hidden states from the recurrent layers are passed to the fully-connected softmax layer to generate the output distributions for every time step in the sequence.

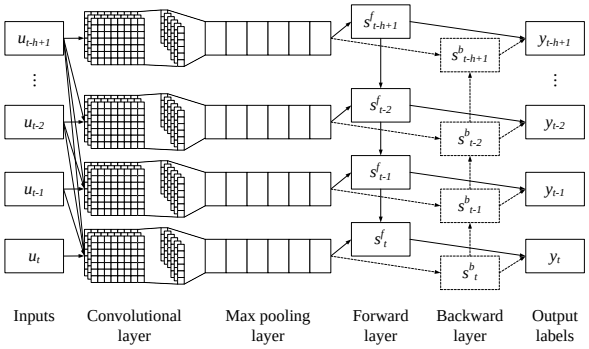


Figure 4: Recurrent convolutional network architecture for dialogue topic tracking. The backward layer is only for the bi-directional mode.

The output from the model at a given time step t is a label sequence $[y_{t-h+1}, \cdots, y_t]$ for the recent h utterances. Since the labels for the earlier utterances should have been already decided at the corresponding turns, only y_t is taken as the final outcome for the current time step. The hypothesis to be examined with this model is whether the other $h-1$ predictions that are not directly reflected to the results could help to improve the tracking performances by being considered together in the process of determining the current topic status.

3.3 Recurrent Convolutional Networks

The last approach proposed in this work aims at combining the two models described in the previous sections. In this model (Figure 4), each feature vector generated through the embedding, convolutional, and max pooling layers in the CNN network (Section 3.1) is connected to the recurrent layers in the RNN model (Section 3.2). This combination is expected to play a significant role in overcoming the limitations of the sentence-level embedding considered as a feature representation in the RNN model. While the previous approach depends only on a pre-trained and non-tunable embedding model, all the parameters in the combined network can be fine-tuned with back propagation by considering the convolutional features extracted at each time step and also the temporal dependencies occurred through multiple time steps in given dialogue sequences.

In computer vision, this kind of models connecting RNNs on top of CNNs is called recurrent convolutional neural networks (RCNN), which have been mostly used for exploring the dependencies between local convolutional features within a single image (Pinheiro and Collobert, 2014; Liang and Hu, 2015). Recently, they are also applied in video processing (Donahue et al., 2015) where visual features are extracted from the image at each frame using CNNs and the temporal aspects are learned with RNNs from the frame sequence of an input video. Our proposed model for dialogue topic tracking was originally motivated by this success of RCNNs particularly in video recognition considering that video and dialogue are analogous from the structural point of view. Each instance of a video and a dialogue consists of a temporal sequence of static units.

4 Evaluation

4.1 Data

To demonstrate the effectiveness of our proposed models, we performed experiments on TourSG corpus released for the fourth dialogue state tracking challenge (DSTC4) (Kim et al., 2016). The dataset consists of 35 dialogue sessions collected from human-human conversations about tourism in Singapore between tour guides and tourists. All the dialogues have been manually transcribed and annotated with the labels for the challenge tasks. For the multi-topic dialogue state tracking which is the main task of the challenge, each dialogue session is divided into sub-dialogues and each segment is assigned with its topic category. Since the task particularly focuses on filling out the topic-specific frame structure with the detailed information representing the dialogue states of a given segment, it has been performed under the assumption that the manual annotations for both segmentations and topic categories are provided as parts of every input. But, in this work for dialogue topic tracking, these labels are considered as the targets to be generated automatically by the models.

Every segment in the dataset belongs to one of eight topic categories. Following the nature of the tourism domain, the 'attraction' category accounts for the highest portion at 40.12% of the segments, which is followed by 'transportation', 'food', 'accommodation', 'shopping', 'closing' and 'opening' in order according to decreasing frequencies. The other 10.53% considered as beyond the scope of the task are annotated with 'other'.

Figure 5 shows the distributions of the segments by not only the topic categories, but also the transition types from two different points of views: the first one is which speaker initiates each segment, and the other is whether the segmentation causes

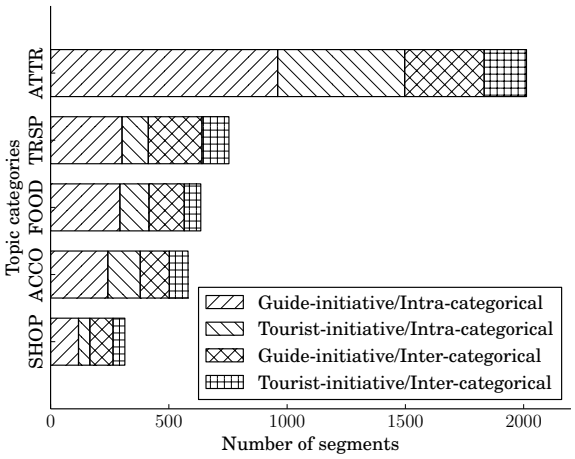


Figure 5: Distributions of the segments in TourSG corpus by topic categories and transition types. ATTR, TRSP, FOOD, ACCO and SHOP denotes the topic categories of attraction, transportation, food, accommodation, and shopping, respectively.

Table 1: Statistics of TourSG corpus. The whole dataset is divided into three subsets for training, development, and test purposes.

Set	# sessions	# segments	# utterances
Train	14	2,104	12,759
Dev	6	700	4,812
Test	15	2,210	13,463
Total	35	5,014	31,034

a topic category shift or not. The most frequent type found in the dataset is guide-initiative and intra-categorical transitions. 63.86% and 61.31% of the total segments are initiated by guides and segmented keeping topic categories, respectively.

For our experiments, all these segment-level annotations were converted into utterance-level BIO tags each of which belongs to one of 15 classes: $(\{\text{B-}, \text{I-}\} \times \{c : c \in C;$ and $c \neq$ 'other'$\}) \cup \{\text{O}\}$, where C consists of all the eight topic categories. The partition of the dataset (Table 1) have been kept the same as the one used for the state tracking task in DSTC4.

4.2 Models

Based on the dataset, we built 16 different models classified into the following five model families.

4.2.1 Baseline 1: Support Vector Machines

The first baseline uses support vector machine (SVM) (Cortes and Vapnik, 1995) models trained with the following features:

- BoN_t: bag of uni/bi/tri-grams in u_t weighted by tf-idf which is the product of term frequency in u_t and inverse document frequency across all the training utterances.
- BoN_{t-1}: bag of n-grams computed in the same way as BoN_t for the previous utterance.
- $\text{BoN}_{\text{history}} = \sum_{j=0}^{h} \left(\lambda^j \cdot \text{BoN}_j\right)$: weighted sum of n-gram vectors in the recent $h = 10$ utterances with a decay factor $\lambda = 0.9$.
- SPK_t, SPK_{t-1}: speakers of the current and the previous utterances.
- $\text{SPK}_{\{t-1,t\}}$: bi-gram of SPK_t and SPK_{t-1}.

Another variation replaces the bag of n-grams with the utterance-level neural embeddings inferred by the pre-trained 300 dimensional doc2vec (Le and Mikolov, 2014) model on 2.9M sentences with 37M words in 553k Singapore-related posts collected from travel forums. Then, the third model takes the concatenation of both bag of n-grams and doc2vec features.

All three baselines were implemented based on the one-against-all approach with the same number of binary classifiers as the total number of classes for multi-label classification. SVM^{light} (Joachims, 1999) was used for building each binary classfier with the linear kernel.

4.2.2 Baseline 2: Conditional Random Fields

To incorporate the temporal aspects also into the linear models, conditional random fields (CRFs) (Lafferty et al., 2001) which have been successfully applied for other sequential labelling problems were used for the second set of baselines. Similar to our proposed RNN architecture (Section 3.2), the recent utterances occurred within the window size of $h = 10$ composed the first-order linear-chain CRFs. Three CRF models were built using CRFsuite (Okazaki, 2007) with the same feature sets as in the SVM models.

4.2.3 CNN-based models

For the CNN architecture (Section 3.1), we compared two different models: the first one learned the word embeddings from scratch with random parameters, while the other was initialized with word2vec (Mikolov et al., 2013) trained on

the same dataset for the doc2vec model in Section 4.2.1. Both approaches generated a dense vector with a dimension of $k = 300$ for each word in utterances. Then, the embedded vectors were concatenated into three matrices representing the current, previous, and history utterances, respectively. While the first two channels for a single utterance, u_t or u_{t-1}, had a size of 65×300 according to the maximum number of words $n = 65$ in the training utterances, the number of rows in the other matrix was 520 which is eight times as large as the others to represent the history utterances from u_{t-9} to u_{t-2} where $h = 10$.

In the convolutional layer, 100 feature maps were learned for each of three different filter sizes $m = \{3, 4, 5\}$ by sliding them over the utterances, which produced 900 feature values in total after the max-pooling operations for all three channels. In addition to these learned features, SPK_t and SPK_{t-1} values introduced in Section 4.2.1 were appended to each feature vector to take the speaker information into account as in the baselines. Before the fully-connected layer, dropout was performed with the rate of 0.25 for regularization. And then, training was done with stochastic gradient descent (SGD) by minimizing categorical cross entropy loss on the training set.

All the neural network-based models in this work were implemented using Theano (Bergstra et al., 2010) with the parameters obtained from the grid search on the development set.

4.2.4 RNN and RCNN-based models

Each proposed recurrent network (Section 3.2 and 3.3) was implemented with four variations categorized by whether the backward layer is included in each model or not and also which architecture is used in the recurrent layers between traditional RNNs and long short-term memories (LSTMs) (Hochreiter and Schmidhuber, 1997). The RCNN models based on LSTMs are particularly called long-term recurrent convolutional networks (LRCN) (Donahue et al., 2015). All the RCNN-based models were initialized with the pre-trained word2vec model in the training phase.

The dimension of the hidden layers of the recurrent cells was chosen to be $|s| = 500$ based on the development set. And the other settings including the parameters, the training algorithm, and the loss fuction were the same as in Section 4.2.3.

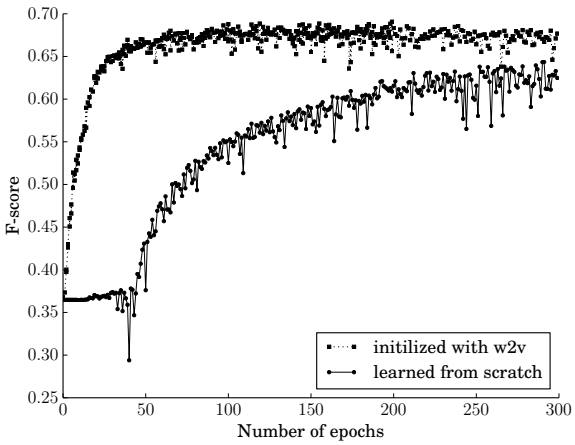


Figure 6: Comparisons of the topic tracking performances of the CNN models with different word embedding approaches according to the number of epochs for training in the development phase.

4.3 Results

Table 2 compares the performances of the models trained on the combination of the training and development sets and evaluated on the test set. The parameters for each model were decided in the development phase which built the models under various different settings only on the training set and validated them with the development set. The evaluations were performed with precision, recall, and F-measure to the manual annotations under three different schedules at tourist turns, guide turns, and all the turns. Then, the statistical significance for every pair was computed using approximate randomization (Yeh, 2000).

Comparing between two baseline families, the sequential extensions with the CRF models contributed to significant improvements ($p < 0.05$) from the SVM models in all the schedules. But in both SVM and CRF models, doc2vec features failed to achieve comparable performances to the simplest bag-of-ngrams features. Even the improvements by combining them to the word features were not statistically significant.

While these sentence-level embeddings trained in the unsupervised manner exposed the limitations in dialogue topic tracking performances, our proposed CNN-based models outperformed all these baselines. Especially, the CNN initialized with the pre-trained word2vec model achieved higher performances by 8.38%, 6.41%, and 7.21% in F-measure under each schedule, respectively, than the best baseline results.

Models	Schedule: Tourist Turns			Schedule: Guide Turns			Schedule: All		
	P	R	F	P	R	F	P	R	F
SVM (BoN+SPK)	61.60	62.18	61.89	58.65	58.42	58.53	59.85	59.94	59.90
SVM (D2V+SPK)	45.05	51.32	47.98	47.78	52.98	50.24	46.66	52.31	49.32
SVM (BoN+SPK+D2V)	61.60	62.18	61.89	58.74	58.53	58.63	59.91	60.01	59.96
CRF (BoN+SPK)	61.18	62.72	61.94	59.27	59.78	59.52	60.05	60.97	60.51
CRF (D2V+SPK)	61.53	49.42	54.81	61.94	49.68	55.13	61.77	49.57	55.00
CRF (BoN+SPK+D2V)	61.22	62.76	61.98	59.30	59.81	59.55	60.08	61.00	60.54
CNN (from scratch)	64.74	63.46	64.10	63.29	62.48	62.88	63.88	62.87	63.37
CNN (with W2V)	69.26	71.49	70.36	65.29	66.65	65.96	66.91	68.61	67.75
Uni-directional RNN	49.46	54.34	51.79	49.54	53.36	51.38	49.51	53.75	51.55
Bi-directional RNN	48.54	49.96	49.24	48.86	49.72	49.29	48.73	49.82	49.27
Uni-directional LSTM	49.52	50.81	50.15	49.41	49.85	49.63	49.45	50.23	49.84
Bi-directional LSTM	48.39	49.05	48.72	48.44	48.58	48.51	48.42	48.77	48.59
Uni-directional RCNN	69.49	71.59	70.52	65.43	66.68	66.05	67.08	68.67	67.86
Bi-directional RCNN	69.81	72.50	71.13	65.49	67.28	66.37	67.25	69.39	68.30
Uni-directional LRCN	69.37	71.45	70.40	**66.22**	67.41	66.81	67.50	69.04	68.26
Bi-directional LRCN	**69.85**	**72.56**	**71.18**	66.04	**67.62**	**66.82**	**67.60**	**69.62**	**68.59**

Table 2: Comparisons of the topic tracking performances with different models. D2V and W2V denote the vectors from doc2vec and word2vec, respectively.

Figure 6 presents the differences between two CNN models observed in the development phase. As the number of epochs increases, the performances of both models also increase up to certain points of saturation. But the model with random initialization required much longer time to be ready to gain scores in earlier iterations and its saturated performance was also lower than the other one learned on top of word2vec.

In contrast to the success of the CNN models, the proposed RNN architectures were not able to produce quality results, which was also caused by the limitations of doc2vec representations as already shown in the baseline results. Although some RNN models showed little performance gains over the SVM baselines only with doc2vec features, they were even worse than the CRF model with the same features.

On the other hand, the RCNN models connecting the results of CNNs to the RNNs contributed to performance improvements not only from the baselines, but also from the CNN models. While the uni-directional RNN was preferred in the RNN models only with doc2vec, the bi-directional LSTM showed better results in the RCNN architectures. As a result, the bi-directional LRCN model achieved the best performances against all the others, which were statistically significant ($p < 0.01$) compared to the second best results with bi-directional RCNN.

Table 3 shows the segmentation performances evaluated by considering only the beginning of each segment predicted by the best model of each architecture family. The proposed CNN and LRCN models demonstrated better capabilities of detecting topic transitions in both intra-categorical and inter-categorical conditions than the baselines. While the CNN model tended to have a higher coverage in segmentation than the others, the LRCN model produced more precise decisions to recognize the boundaries on the strength of the consideration of conversational coherences in dialogue history sequences.

However, the segmentation performances even with the best models were still very limited especially for inter-categorical transitions. And most of the models in the experiment had better performances in tourist turns than guide turns, as shown in Table 2. Considering the general characteristics of the target domain conversations that guide-driven and inter-categorical transitions are more likely to be dependent on human background knowledge than tourist-driven and intra-categorical cases, respectively, the current limitations are expected to be tackled by leveraging external resources into the models in future.

Finally, the generated errors from the models were categorized into the following error types:

Models	Intra-categorical			Inter-categorical			All		
	P	R	F	P	R	F	P	R	F
SVM (BoN+SPK+D2V)	40.22	30.19	34.49	8.68	28.14	13.26	18.65	29.51	22.85
CRF (BoN+SPK+D2V)	36.42	25.92	30.28	11.57	24.40	15.70	21.58	25.41	23.34
CNN (with W2V)	41.25	**41.50**	**41.37**	17.02	**40.87**	24.03	28.06	**41.29**	33.41
Bi-directional LRCN	**44.82**	38.28	41.29	**17.87**	40.72	**24.84**	**29.41**	39.09	**33.57**

Table 3: Comparisons of the segmentation performances with different models.

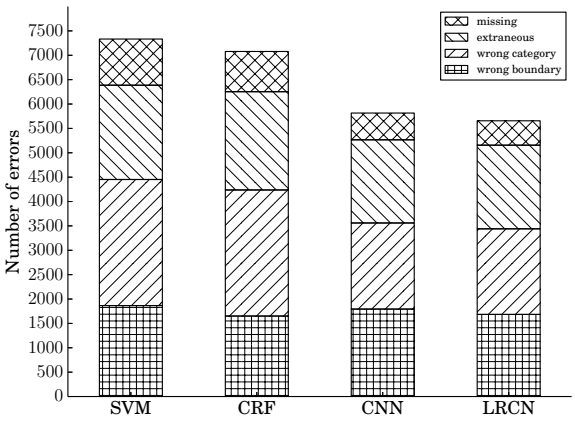


Figure 7: Distributions of errors generated from the best model of each architecture.

- Missing predictions: when the reference belongs to one of the labels other than 'O', but the model predicts it as 'O'.
- Extraneous labelling: when the reference belongs to 'O', but the model predicts it as another label.
- Wrong categorizations: when the reference belongs to a category other than 'O', but the model predicts it as another wrong category.
- Wrong boundary detections: when the model outputs the correct category, but with a wrong prediction from 'B' to 'I' or from 'I' to 'B'.

The error distributions in Figure 7 indicate that the significantly decreased numbers of wrong categories were the decisive factor in performance improvements by our proposed approaches from the baselines. Besides, the enhanced capabilities of the models in distinguishing between 'O' and other labels were demonstrated by the reduced numbers of missing and extraneous predictions. The sequential architectures in CRF and LRCN models also showed its effectiveness especially in boundary detection, as expected.

5 Conclusions

This paper presented various neural network architectures for dialogue topic tracking. Convolutional neural networks were proposed to capture the semantic aspects of utterances given at each moment, while recurrent neural networks were intended to incorporate temporal aspects in dialogue histories into tracking models. Experimental results showed that the proposed approaches helped to improve the topic tracking performance with respect to the linear baseline models.

Furthering this work, there would be still much room for improvement in future. Firstly, the architectures based on a single convolutional layer and a single bi-directional recurrent layer in the proposed models can be extended by adding more layers as well as utilizing more advanced components including hierarchical CNNs (Kalchbrenner et al., 2014b) to deal with utterance compositionalities or attention mechanisms (Denil et al., 2012) to focus on more important segments in dialogue sequences.

Secondly, the use of external knowledge could be a key to success in dialogue topic tracking, as proved in the previous studies. However, this work only takes internal dialogue information into account for making decisions. If we develop a good way of leveraging other useful resources into the neural network architectures, better performance can be expected especially for guide-driven and inter-categorical topic transitions that are considered to be more dependent on background knowledge of the speakers.

The other direction of our future work is to investigate joint models for tracking dialogue topics and states simultaneously. Although the previous multi-topic state tracking task has assumed that the topics should be given as inputs to state trackers, we expect that a joint approach can contribute to both problems by dealing with the bi-directional relationships between them.

References

P. H. Adams and C. H. Martell. 2008. Topic detection and extraction in chat. In *Proceedings of the 2008 IEEE International Conference on Semantic Computing*, pages 581–588.

M. Auli, M. Galley, C. Quirk, and G. Zweig. 2013. Joint language and translation modeling with recurrent neural networks. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1044–1054.

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. 2010. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7.

D. Bohus and A. Rudnicky. 2003. Ravenclaw: dialog management using hierarchical task decomposition and an expectation agenda. In *Proceedings of the European Conference on Speech, Communication and Technology*, pages 597–600.

A. Celikyilmaz, D. Hakkani-Tür, and G. Tür. 2011. Approximate inference for domain detection in spoken language understanding. In *Proceedings of the 12th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, pages 713–716.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

C. Cortes and V. Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

M. Denil, L. Bazzani, H. Larochelle, and N. de Freitas. 2012. Learning where to attend with deep architectures for image tracking. *Neural computation*, 24(8):2151–2184.

J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634.

Y. Esteve, M. Bouallegue, C. Lailler, M. Morchid, R. Dufour, G. Linares, D. Matrouf, and R. De Mori. 2015. Integration of word and semantic features for theme identification in telephone conversations. In *Natural Language Dialog Systems and Intelligent Assistants*, pages 223–231. Springer.

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

S. Ikeda, K. Komatani, T. Ogata, H. G. Okuno, and H. G. Okuno. 2008. Extensibility verification of robust domain selection against out-of-grammar utterances in multi-domain spoken dialogue system. In *Proceedings of the 9th INTERSPEECH*, pages 487–490.

T. Joachims. 1999. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. 2014a. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665.

N. Kalchbrenner, E. Grefenstette, and P. Blunsom. 2014b. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665.

S. Kim, R. E. Banchs, and H. Li. 2014a. A composite kernel approach for dialog topic tracking with structured domain knowledge from wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 19–23.

S. Kim, R. E. Banchs, and H. Li. 2014b. Wikipedia-based kernels for dialogue topic tracking. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 131–135.

S. Kim, L. F. D'Haro, R. E. Banchs, J. D. Williams, and M. Henderson. 2016. The fourth dialog state tracking challenge. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS)*.

Y. Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751.

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

K. Lagus and J. Kuusisto. 2002. Topic identification in natural language dialogues using neural networks. In *Proceedings of the 3rd SIGdial workshop on Discourse and dialogue*, pages 95–102.

Q. Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

C. Lee, S. Jung, and G. G. Lee. 2008. Robust dialog management with n-best hypotheses using dialog examples and agenda. In *Proceedings of the*

46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 630–637.

M. Liang and X. Hu. 2015. Recurrent convolutional neural network for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3367–3375.

B. Lin, H. Wang, and L. Lee. 1999. A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history. In *Proceedings of the IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

S. Liu, N. Yang, M. Li, and M. Zhou. 2014. A recursive recurrent neural network for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1491–1500.

T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NIPS)*, pages 3111–3119.

M. Morchid, R. Dufour, M. Bouallegue, G. Linares, and R. De Mori. 2014a. Theme identification in human-human conversations with features from specific speaker type hidden spaces. In *INTERSPEECH*, pages 248–252.

M. Morchid, R. Dufour, P.M. Bousquet, M. Bouallegue, G. Linares, and R. De Mori. 2014b. Improving dialogue classification using a topic space representation and a gaussian classifier based on the decision rule. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 126–130. IEEE.

T. Nakata, S. Ando, and A. Okumura. 2002. Topic detection based on dialogue history. In *Proceedings of the 19th international conference on Computational linguistics (COLING)*, pages 1–7.

N. Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

P. Pinheiro and R. Collobert. 2014. Recurrent convolutional neural networks for scene labeling. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 82–90.

L. A. Ramshaw and M. P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd Workshop on Very Large Corpus*, pages 88–94.

S. Roy and L. V. Subramaniam. 2006. Automatic generation of domain models for call centers from noisy transcriptions. In *Proceedings of COLING/ACL*, pages 737–744.

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

Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, pages 373–374. International World Wide Web Conferences Steering Committee.

I. Sutskever, J. Martens, and G. E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

A. Yeh. 2000. More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 947–953.

W. Yih, X. He, and C. Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 643–648.

Cross-Lingual Lexico-Semantic Transfer in Language Learning

Ekaterina Kochmar
The ALTA Institute
University of Cambridge
ek358@cam.ac.uk

Ekaterina Shutova
Computer Laboratory
University of Cambridge
es407@cam.ac.uk

Abstract

Lexico-semantic knowledge of our native language provides an initial foundation for second language learning. In this paper, we investigate whether and to what extent the lexico-semantic models of the native language (L1) are transferred to the second language (L2). Specifically, we focus on the problem of lexical choice and investigate it in the context of three typologically diverse languages: Russian, Spanish and English. We show that a statistical semantic model learned from L1 data improves automatic error detection in L2 for the speakers of the respective L1. Finally, we investigate whether the semantic model learned from a particular L1 is portable to other, typologically related languages.

1 Introduction

Lexico-semantic knowledge of our native language is one of the factors that underlie our ability to communicate and reason about the world. It is also the knowledge that guides us in the process of second language learning. Lexico-semantic variation across languages (Bach and Chao, 2008) makes lexical choice a challenging task for second language learners (Odlin, 1989). For instance, the meaning of the English expression *pull the trigger* is realised as **push the trigger* in Russian and Spanish, possibly leading to errors of lexical choice by Russian and Spanish speakers learning English. Our native language (L1) plays an essential role in the process of lexical choice. When choosing between several linguistic realisations in L2, non-native speakers may rely on the lexico-semantic information from L1 and select a translational equivalent that they deem to match their communicative intent best. For example, Russian speakers **do exceptions* and *offers* instead of *making* them, and **find decisions* instead of *finding solutions*, since in Russian *do* and *make* have a single translational equivalent (*delat'*), and so do *decision* and *solution* (*resheniye*). As a result, non-native speakers who tend to fall back to their L1 translate phrases word-for-word, violating English lexico-semantic conventions.

The effect of L1 interference on lexical choice in L2 has been pointed out in a number of studies (Chang et al., 2008; Rozovskaya, 2010; Rozovskaya, 2011; Dahlmeier and Ng, 2011). Some of these studies also demonstrated that using L1-specific properties, such as the error patterns of speakers of a given L1 or L1-induced paraphrases, improves the performance of automatic error correction in non-native writing. However, neither of the approaches has constructed a semantic model from L1 data and systematically studied the effects of its transfer onto L2. In addition, most previous work has focused on error correction, bypassing the task of error detection for lexical choice. Lexical choice is one of the most challenging tasks for both non-native speakers and automated error detection and correction (EDC) systems. The results of the most recent shared task on EDC, which spanned all error types including lexical choice, show that most teams either did not propose any algorithms for this type of errors or did not perform well on them (Ng, 2014).

In this paper, we experimentally investigate the influence of L1 on lexical choice in L2 and whether lexico-semantic models from L1 are transferred to L2 during language learning. For this purpose, we induce L1 and L2 semantic models from corpus statistics in each language independently, and then use the discrepancies between the two models to identify errors of lexical choice. We focus on two types of *verb–noun* combinations, VERB–DIRECT_OBJECT (*dobj*) and

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 974–983,
Berlin, Germany, August 7-12, 2016.

SUBJECT–VERB (*subj*), and consider two widely spoken L1s from different language families – Russian and Spanish. We conduct our experiments using the Cambridge Learner Corpus (Nicholls, 2003), containing writing samples of non-native speakers of English. Spanish speakers account for around 24.6% of the non-native speakers represented in this corpus and Russian speakers for 4%.

Our experiments test two hypotheses: (1) that L1 effects in the lexical choice in L2 reveal themselves in the difference of the word association strength in the L1 and L2; and (2) that L1 lexico-semantic models are portable to other, typologically related languages. To the best of our knowledge, our paper is the first one to experimentally investigate these questions. Our results demonstrate that L1-induced information improves automatic error detection for lexical choice, confirming the hypothesis that L1 speakers rely on semantic knowledge from their native language during L2 learning. We test the second hypothesis by verifying that Russian speakers exhibit similar trends in errors with the speakers of other Slavic languages, and Spanish speakers with the speakers of other Romance languages. We find that the L1-induced information from Russian and Spanish is effective in assessing lexical choice of the speakers of other languages for both language groups.

2 Related work

2.1 Error detection in content words

Early approaches to collocation error detection relied on manually created databases of correct and incorrect word combinations (Shei and Pain, 2000; Wible et al., 2003; Chang et al., 2008). Constructing such databases is expensive and time-consuming, and therefore, more recent research turned to the use of machine learning techniques.

Leacock et al. (2014) note that most approaches to detection and correction of collocation errors compare the writer's word choice to the set of alternatives using association strength measures and choose the combination with the highest score, reporting an error if this combination does not coincide with the original choice (Futagi et al., 2008; Östling and Knutsson, 2009; Liu et al., 2009). This strategy is expensive as it relies on comparison with a set of alternatives, limited in capacity as it depends on the quality of the alternatives generated and circular as the detection cannot be performed independently of the correction. Our approach alleviates these problems, since error detection depends on the original combination only.

Some previous approaches focused on correction only (Dahlmeier and Ng, 2011; Kochmar and Briscoe, 2015), and although they show promising results, they have not attempted to perform error detection in lexical choice. Kochmar and Briscoe (2014) focus on error detection, but their system addresses adjective–noun combinations and does not use L1-induced information.

2.2 L1 factors in L2 writing

The influence of an L1 on lexical choice in L2 and the resulting errors have been previously studied (Chang et al., 2008; Östling and Knutsson, 2009; Dahlmeier and Ng, 2011). These works focus on errors in particular L1s and use the translational equivalents directly to improve candidate selection and quality of corrections. Dahlmeier and Ng (2011) show that L1-induced paraphrases outperform approaches based on edit distance, homophones, and WordNet synonyms in selecting the appropriate corrections.

Rozovskaya and Roth (2010) show that an error correction system for prepositions benefits from restricting the set of possible corrections to those observed in the non-native data. Rozovskaya and Roth (2011) further demonstrate that the models perform better when they use knowledge about error patterns of the non-native writers. According to their results, an error correction algorithm that relies on a set of priors dependent on the writer's preposition and the writer's L1 outperforms other methods. Madnani et al. (2008) show promising results in whole-sentence grammatical error correction using round-trip translations from Google Translate via 8 different pivot languages.

The results of these studies suggest that L1 is a valuable source of information in EDC. However, all these works use isolated translational equivalents and focus on error correction only. In contrast, we construct holistic semantic models of L1 from L1 corpora and use these models to perform the more challenging task of error detection.

3 Data

We first use large monolingual corpora in Spanish, Russian and English to build word association models for each of the languages. We then apply the resulting models for error detection in the English learner data.

3.1 L1 Data

Spanish data The Spanish data was extracted from the Spanish Gigaword corpus (Mendonca et al., 2011), a one billion-word collection of news articles in Spanish. The corpus was parsed using the Spanish Malt parser (Nivre et al., 2007; Ballesteros et al., 2010). We extracted VERB–SUBJECT and VERB–DIRECT_OBJECT relations from the output of the parser, which we then used to build an L1 word association model for Spanish.

Russian data The Russian data was extracted from the RU-WaC corpus (Sharoff, 2006), a two billion-word representative collection of texts from the Russian Web. The corpus was parsed using Malt dependency parser for Russian (Sharoff and Nivre, 2011), and the VERB–SUBJECT and VERB–DIRECT_OBJECT relations were extracted from the parser output to create an L1 word association model for Russian.

Dictionaries and translation Once the L1 word associations have been computed for the verb–noun pairs, we identify possible translations for verbs and nouns (in each pair) in isolation, as a language learner might do. To create the translation dictionaries, we extracted translations from the English–Spanish and English–Russian editions of Wiktionary, both from the translation sections and the gloss sections if the latter contained single words as glosses. We focus on verb–noun pairs, therefore multi-word expressions were universally removed. We added inverse translations for every original translation. We then created separate translation dictionaries for each language and part-of-speech tag combination from the resulting collection of translations.

3.2 L2 data

To build the English word association model, we have used a combination of the British National Corpus (Burnard, 2007) and the UKWaC (Baroni et al., 2009). The corpora were parsed by the RASP parser (Briscoe et al., 2006) and VERB–SUBJECT and VERB–DIRECT_OBJECT relations were extracted from the parser output. Since the UKWaC is a Web corpus, we assume that the data contains a certain amount of noise, e.g. typographical errors, slang and non-words. We filter these out by checking that the verbs and nouns in the extracted relations are included in WordNet (Miller, 1995) with the appropriate part of speech.

3.3 Learner data

To extract the verb–noun combinations that have been used by non-native speakers in practice, we use the *Cambridge Learner Corpus* (CLC), which is a 52.5 million-word corpus of learner English collected by Cambridge University Press and Cambridge English Language Assessment since 1993 (Nicholls, 2003). It comprises English examination scripts written by learners of English with 148 different L1s, ranging across multiple examinations and covering all levels of language proficiency. A 25.5 million-word component of the CLC has been manually error-annotated.

We have preprocessed the CLC with the RASP parser (Briscoe et al., 2006), as it is robust when applied to ungrammatical sentences. We have then extracted all *dobj* and *subj* combinations: in total, we have extracted $187,109$ *dobj* and $225,716$ *subj* combinations. We have used the CLC error annotation to split the data into correct combinations and errors. We note that some verb–noun combinations are annotated both as being correct and as errors, depending on their wider context of use. To ensure that the annotation we use in our experiments is reliable and not context-dependent, we have empirically set a threshold to filter out ambiguously annotated instances. The set of correct word combinations includes only those word pairs that are used correctly in at least 70% of the cases they occur in the CLC; the set of errors includes only those that are used incorrectly at least 70% of the time.

3.4 Experimental datasets

We split the annotated CLC data by language and relation type. Table 1 presents the statistics on the datasets collected.[1] We extract the verb–noun combinations from the CLC texts written by native speakers of Russian (RU) and Spanish (ES) to test our first hypothesis, as well as by speakers of ALL L1s in the CLC to test our second hypothesis. We then filter the extracted relations using the translated verb–noun pairs from Russian and Spanish corpora.

We note that Russian and Spanish have comparable number of word combinations in L1-specific subsets – $10K$-$12K$ for *dobj* and *subj* combinations – and comparable error rates (ERR). We also note that the error rates in the *dobj* sub-

[1]The data is available at `http://www.cl.cam.ac.uk/~ek358/cross-ling-data.html`

Source	CLC	Total	ERR (%)	verbs	nouns
RU_{dobj}	RU	11,184	12.55	786	1,918
	ALL	62,923	14.02	1,387	4,168
RU_{subj}	RU	10,417	7.90	734	1,775
	ALL	63,649	9.49	1,403	4,374
ES_{dobj}	ES	11,959	14.66	705	1,926
	ALL	32,966	15.17	1,072	2,928
ES_{subj}	ES	9,899	8.09	573	1,733
	ALL	26,766	9.42	877	2,762

Table 1: Statistics on the datasets collected.

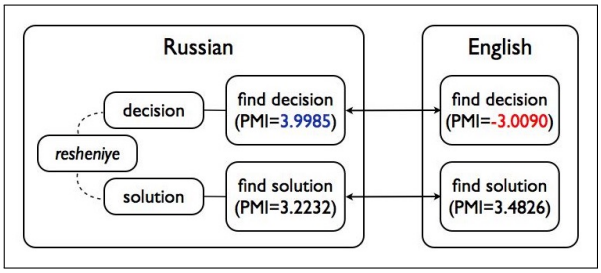


Figure 1: Russian to English interface for **find decision*.

sets are higher than in *subj* subsets, presumably, because VERB–SUBJECT combinations allow for more flexibility in lexical choice. We find a large number of translated word combinations in other L1s, and it is interesting to note that the error rates are higher across multiple languages than in the same L1s, which corroborates our second hypothesis that the lexico-semantic models from L1s transfer to L2. The last two columns of Table 1 show how diverse our datasets are in terms of verbs and nouns used in the constructions: for example, RU_{dobj} subset contains combinations with 786 different verbs and 1,918 different nouns.

4 Methods

Our approach to detecting lexico-semantic transfer errors relies on the intuition that a mismatch between the lexico-semantic models in two languages reveals itself in the difference in word association scores. We argue that a high association score of a *verb–noun* combination in L1 shows that it is a collocation in L1, but low association score of its translational equivalent in L2 signals an error in L2 stemming from the lexico-semantic transfer. Following previous research (Baldwin and Kim, 2010), we measure the strength of verb–noun association using pointwise mutual information (PMI). Figure 1 illustrates this intuition. In Russian, both **find decision* vs. *find solution* have a high PMI score. However, in English the latter has a high PMI while the former has a negative PMI. We expect such a discrepancy in word association to be an indicator of error of lexical choice, driven by the L1 semantics.

We treat the task of lexico-semantic transfer error detection as a binary classification problem and train a classifier for this task. The classifier uses a combination of L1 and L2 semantic features. If our hypothesis holds, we expect to see an improvement in the classifier's performance when adding L1 semantic features.

4.1 L2 lexico-semantic features

We experiment with two types of L2 features: lexico-semantic features and semantic vector space features.

Lexico-semantic features include:

- `pmi` in L2: we estimate the association strength between the noun and verb using the combined BNC and UKWaC corpus;
- `verb` and `noun`: the identity of the verb and the noun in the pair, encoded in a numerical form in the range of $(0, 1)$. The motivation behind that step is that certain words are more error-prone than others and converting them into numerical features helps the classifier to use this information.

Semantic vector space features Kochmar and Briscoe (2014) obtained state-of-the-art results in error detection by using the semantic component of the content word combinations. We reimplement these features and test their impact on our task. We extracted the noun and verb vectors from the publicly available `word2vec` dataset of word embeddings for 3 million words and phrases.[2] The 300-dimensional vectors have been trained on a part of Google News dataset (about 100 billion words) using `word2vec` (Mikolov et al., 2013). The *dobj* and *subj* vectors are then built using element-wise addition on the vectors (Mitchell and Lapata, 2008; Mikolov et al., 2013; Kochmar and Briscoe, 2014).

Once the compositional vectors are created, the method relies on the idea that correct combinations can be distinguished from the erroneous ones by certain vector properties (Vecchi et al., 2011; Kochmar and Briscoe, 2014). We implement a set of numerical features based on the following properties of the vectors:

[2] `code.google.com/archive/p/word2vec/`

- `length` of the additive (vn) vector
- $\texttt{cos}_{vn \wedge n}$ – cosine between the vn vector and the noun vector
- $\texttt{cos}_{vn \wedge v}$ – cosine between the vn vector and the verb vector
- $\texttt{dist}_{10}$ – distance to the 10 nearest neighbours of the vn vector
- `lex-overlap` – proportion of the 10 nearest neighbours of the vn vector containing the verb/noun
- `comp-overlap` – overlap between the 10 neighbours of the vn vector and 10 neighbours of the verb/noun vector
- $\texttt{cos}_{v \wedge n}$ – cosine between the verb and the noun vectors.

The 10 nearest neighbours are retrieved in the combined semantic space containing word embeddings and additive phrase vectors. All features, except for the last one, have been introduced in previous work and showed promising results (Vecchi et al., 2011; Kochmar and Briscoe, 2014). For example, it has been shown that the distance from the constructed word combination vector to its nearest neighbours is one of the discriminative features of the error detection classifier. Manual inspection of the vectors and nearest neighbours shows that the closest neighbour to **find decision* is *see decision* with the similarity of 0.8735 while the closest one to *find solution* is *discover solution* with the similarity of 0.9048.

We implement an additional $\text{cos}_{v \wedge n}$ feature based on the intuition that the distance between the verb and noun vectors themselves may indicate a semantic mismatch and thus help in detecting lexical choice errors.

4.2 L1 lexico-semantic features

We first quantified the strength of association between the L1 verbs and nouns in the original L1 data, using PMI. We then generated a set of possible translations for each verb–noun pair in L1 using the translation dictionaries. Each verb–noun pair in the CLC was then mapped to one of the translated L1 pairs and its L1 features. We used the following L1 features in classification:

- `pmi` in L1: we estimate the strength of association on the original L1 corpora;
- `difference` between the PMI of the verb–noun pair in L1 and in L2.

4.3 Classification

Classifier settings We treat the task as a binary classification problem and apply a linear SVM classifier using `scikit-learn LinearSVC` implementation.[3] The error rates in Table 1 show that we are dealing with a two-class problem where one class (*correct* word combinations) significantly outnumbers the other class (*errors*) by up to 11:1 (on RU_{subj}). To address the problem of class imbalance, we use subsampling: we randomly split the set of correct word combinations in n samples keeping the majority class baseline under 0.60, and run n experiments over the samples. We apply 10-fold cross-validation within each sample. The results reported in the following sections are averaged across the samples for each dataset.

Evaluation The goal of the classifier is to detect errors, therefore we primarily focus on its performance on the *error* class and, in addition to accuracy, report *precision* (P), *recall* (R) and F_1 on this class. Previous studies (Nagata and Nakatani, 2010) suggest that systems with high precision in detecting errors are more helpful for L2 learning than systems with high recall as non-native speakers find misidentified errors very misleading. In line with this research, we focus on maximising precision on the *error* class.

Baseline We compare the performance of our different feature sets to the baseline classifier which uses L2 co-occurrence frequency of the verb and noun in the pair as a single feature. Frequency sets a competitive baseline as it is often judged to be the measure of acceptability of an expression and many previous works relied on the frequency of occurrence as an evidence of acceptability (Shei and Pain, 2000; Futagi et al., 2008).

5 Experimental Results

To test our hypothesis that lexico-semantic models are transferred from L1 to L2, we first run the set of experiments on the L1 subsets of the CLC data, that is $\text{RU} \rightarrow \text{RU}_{CLC}$ and $\text{ES} \rightarrow \text{ES}_{CLC}$, where the left-hand side of the notation denotes the lexico-semantic model and the right-hand side the L1 of the speakers that produced the word pairs extracted from the CLC. We incrementally add the features, starting with the set of lexico-semantic

[3] `scikit-learn.org/`

L1	Features	Acc	$\mathbf{P}_e$	R_e	F_{1_e}
RU_{dobj}	baseline	55.68	47.77	61.44	53.55
	pmi_{En}	64.74	59.76	47.55	52.96
	+verb	64.79	59.87	47.56	53.01
RU_{subj}	baseline	54.48	46.30	63.96	53.17
	pmi_{En}	67.02	58.86	62.74	60.74
	+verb	67.64	59.84	62.17	60.98
ES_{dobj}	baseline	56.74	52.25	74.44	61.36
	pmi_{En}	64.28	61.75	59.55	60.63
	+verb	64.34	61.80	59.67	60.71
ES_{subj}	baseline	54.45	46.71	70.31	56.00
	pmi_{En}	69.22	61.35	68.83	64.87
	+verb	69.51	61.79	68.58	65.00

Table 2: System performance (in %) using L2 lexico-semantic features, L1 $\rightarrow$ L1$_{CLC}$.

features in L2 that are readily available without reference to the L1, and later adding L1 semantic features, and measure their contribution.

5.1 L2 lexico-semantic features

The first system configuration we experiment with uses the set of lexico-semantic features from L2. Table 2 reports the results. Our experiments show that a classifier that uses L2 PMI ($\mathtt{pmi}_{En}$) as a single feature performs with relatively high accuracy: on all four datasets it outperforms the baseline classifier achieving an increase from 7.54% (on ES_{dobj}) up to 14.77% (on ES_{subj}) in accuracy.

Adding the `noun` as a feature decreases performance of the classifier and we do not further use this feature. The `verb` used as an additional feature consistently improves classifier performance.

5.2 L2 semantic vector space features

Next, we test the combination of the semantic vector space features (`sem`) and combine them with two L2 lexico-semantic features including $\mathtt{pmi}_{En}$ and `verb` (denoted as $\mathtt{ft}_{En}$ hereafter for brevity). Table 3 reports the results.

We note that the semantic vector space features on their own yield precision of $50\% - 52\%$ on the *error* class in *dobj* combinations and lower than 50% on *subj* combinations. This suggests that the classifier misidentifies correct combinations as errors more frequently than it correctly detects errors. Moreover, recall of this system configuration is also low on all datasets. Adding the semantic vector space features to the other L2 semantic features, however, improves the performance, as shown in Table 3. As both groups of features refer to the phenomena in L2, the results suggest that they complement each other.

L1	Features	Acc	$\mathbf{P}_e$	R_e	F_{1_e}
RU_{dobj}	sem	58.36	50.72	6.98	12.22
	+ft_{En}	65.90	58.64	62.18	60.35
RU_{subj}	sem	58.62	36.07	3.40	6.12
	+ft_{En}	68.37	60.05	66.48	63.07
ES_{dobj}	sem	54.51	52.01	20.78	29.48
	+ft_{En}	66.87	63.36	67.08	65.16
ES_{subj}	sem	58.63	49.37	9.27	15.47
	+ft_{En}	70.75	62.21	74.31	67.72

Table 3: System performance (in %) using a combination of L2 semantic features, L1 $\rightarrow$ L1$_{CLC}$.

L1	Features	Acc	$\mathbf{P}_e$	R_e	F_{1_e}
RU_{dobj}	ft_{En}	64.79	59.87	47.56	53.01
	+pmi_{L1}	66.05	58.74	62.72	60.67
RU_{subj}	ft_{En}	67.64	59.88	62.17	60.98
	+pmi_{L1}	68.68	62.10	69.61	64.38
ES_{dobj}	ft_{En}	64.34	61.80	59.67	60.71
	+pmi_{L1}	66.89	63.01	68.61	65.68
ES_{subj}	ft_{En}	69.51	61.79	68.58	65.00
	+pmi_{L1}	71.19	62.10	77.66	69.00

Table 4: System performance (in %) using L1 and L2 lexico-semantic features, L1 $\rightarrow$ L1$_{CLC}$.

5.3 L1 lexico-semantic features

Finally, we add the L1 lexico-semantic features to the well-performing L2 features (`pmi` and `verb`). The combination of L1 lexico-semantic features with the L2 lexico-semantic and semantic vector space features achieves lower results, therefore we do not report them here. The use of L1 `pmi` improves both the accuracy and the F-score of the *error* class (see Table 4). For the ease of comparison, we also include the results obtained using a combination of L1 lexico-semantic features (denoted $\mathtt{ft}_{En}$). The addition of the explicit `difference` feature between the two PMIs has not yielded further improvement. This is likely to be due to the fact that the classifier already implicitly captures the knowledge of this difference in the form of individual L1 and L2 PMIs.

We note that the system using a combination of L1 and L2 lexico-semantic features gains an absolute improvement in accuracy from 1.04% for RU_{subj} to 2.55% on ES_{dobj}. The performance on the *error* class improves in all but one case (P_e on RU_{dobj}), with an absolute increase in F_1 up to 7.66%. The system has both a higher coverage in error detection (a rise in recall) and a higher precision. The improvement in performance across all four datasets is statistically significant at 0.05 level. These results demonstrate the effect of lexico-semantic model transfer from L1 to L2.

6 Effect on different L1s

Next, we test our second hypothesis that a lexico-semantic model from one L1 is portable across several L1s, in particular, typologically related ones. We first experiment with the data representing all L1s in the CLC and then with the data representing a specific language group. We compare the performance of the baseline system using verb–noun co-occurrence frequency as a single feature, the system that uses L2 semantic features only and the system that combines both L2 and L1 semantic features.

6.1 Experiments on all L1s

Table 1 shows that using the translated verb–noun combinations from our L1s (RU and ES) we are able to find a large amount of both correct and erroneous combinations in different L1s in the CLC including RU and ES (see ALL). This gives us an initial confirmation that the lexico-semantic models may be shared across multiple languages.

We then experiment with error detection across all L1s represented in the CLC. The results are shown in Table 5. The baseline system achieves similar performance on RU $\rightarrow$ ALL$_{CLC}$ as on RU $\rightarrow$ RU$_{CLC}$, and better performance on ES $\rightarrow$ ALL$_{CLC}$ than on ES $\rightarrow$ ES$_{CLC}$. The results obtained with the L2 lexico-semantic features are also comparable: the system achieves an absolute increase in accuracy of up to 9.86% for the model transferred from RU$_{subj}$, reaching an accuracy of around $65 - 66\%$ with balanced performance in terms of precision and recall on *errors*.

When the L1 lexico-semantic features are added to the model, we observe an absolute increase in the accuracy ranging from 0.57% (for RU$_{subj}$) to 1.43% (for ES$_{dobj}$). The Spanish lexico-semantic model has a higher positive effect on all measures, including precision on the *error* class. Although the addition of the L1 lexico-semantic features does not have a significant effect on the accuracy and precision, the system achieves an absolute improvement in recall of up to 12.71% (on RU$_{dobj}$). That is, the system that uses L1 lexico-semantic features is able to find more errors in the data originating with a set of different L1s. Generally, the results of the Spanish model are more stable and comparable to the results in the previous Section, which may be explained by the fact that Spanish is more well-represented in the CLC.

L1	Features	Acc	P_e	R_e	F_{1_e}
RU$_{dobj}$	baseline	55.13	50.17	72.14	58.99
	ft$_{En}$	63.58	59.73	57.98	58.85
	+pmi$_{L1}$	64.60	58.81	70.69	64.20
RU$_{subj}$	baseline	54.56	47.95	71.10	56.71
	ft$_{En}$	64.42	57.27	62.64	59.83
	+pmi$_{L1}$	64.99	57.24	68.17	62.21
ES$_{dobj}$	baseline	59.35	55.38	71.87	62.51
	ft$_{En}$	64.32	61.89	63.47	62.67
	+pmi$_{L1}$	65.75	61.90	71.37	66.30
ES$_{subj}$	baseline	58.34	50.90	66.97	57.48
	ft$_{En}$	65.57	58.32	64.09	61.06
	+pmi$_{L1}$	66.54	58.80	68.72	63.36

Table 5: System performance (in %) using L1 and L2 lexico-semantic features, L1 $\rightarrow$ all L1s.

6.2 Experiments on related L1s

The results on ALL L1s confirm our expectations: since we have extracted verb–noun combinations that originate with two particular L1s from the set of all different L1s in the CLC, and then used the L1 lexico-semantic features, the system is able to identify more errors thus we observe an improvement in recall. The precision, however, does not improve, possibly because the set of errors in ALL L1s is different from that in the two L1s we rely on to build the lexico-semantic models. The final question that we investigate is whether the lexico-semantic models of our L1s are directly portable to typologically related languages. If this is the case, we expect to see an effect on the precision of the classifier as well as on the recall.

We experiment with the following groups of related languages ordered by the number of verb–noun pairs we found in the CLC data:

- `Ru group`: Russian, Polish, Czech, Slovak, Serbian, Croatian, Bulgarian, Slovene;
- `Es group`: Spanish, Italian, Portuguese, French, Catalan, Romanian, Romansch.

In addition to investigating the effect of the L1 lexico-semantic model on the whole language group, we also consider its effects on individual languages. We chose Polish for the RU model, and Italian for the ES model as these two languages have the most data representing their native speakers in the CLC. Table 6 shows the number of verb–noun combinations and error rates for the language groups and these individual languages.

The results are presented in Tables 7 and 8. They exhibit similar trends in the change of the system performance on L1 $\rightarrow$ L1_GROUP as we

Source	Targets	Total	ERR
RU$_{dobj}$	Slavic	18,721	9.19
	Polish	11,327	8.16
RU$_{subj}$	Slavic	18,511	6.80
	Polish	11,204	6.42
ES$_{dobj}$	Romance	18,898	12.81
	Italian	6,375	10.92
ES$_{subj}$	Romance	15,871	7.57
	Italian	5,300	6.98

Table 6: Statistics on the L1 groups and related languages.

L1	Features	Acc	$\mathbf{P}_e$	R_e	F_{1_e}
RU$_{dobj}$	baseline	57.08	51.80	71.58	59.78
	ft$_{En}$	64.20	60.99	55.36	58.04
	+pmi$_{L1}$	65.77	61.06	64.78	62.86
RU$_{subj}$	baseline	56.43	49.52	62.04	54.24
	ft$_{En}$	62.26	55.84	50.02	52.76
	+pmi$_{L1}$	62.78	56.02	54.48	55.21
ES$_{dobj}$	baseline	59.18	51.44	72.31	59.97
	ft$_{En}$	65.14	59.82	53.83	56.66
	+pmi$_{L1}$	66.24	58.92	67.00	62.70
ES$_{subj}$	baseline	58.10	52.95	77.43	62.45
	ft$_{En}$	66.29	61.24	68.45	64.64
	+pmi$_{L1}$	67.00	61.68	70.50	65.78

Table 7: System performance (in %) using L1 and L2 lexico-semantic features, L1 → L1_GROUP.

see for L1 → ALL L1s. Adding the L1 lexico-semantic features has only a minor effect on accuracy and precision, and a more pronounced effect on recall. On the contrary, when we test the system on one particular related L1 (Table 8) we observe the opposite effect: with the exception of ES$_{subj}$ data, precision and accuracy improve, suggesting that the error detection system using L1-induced information identifies errors more precisely.

Overall, the observed gains in performance indicate that L1 semantic models contribute information to lexical choice error detection in L2 for the speakers of typologically related languages. This in turn suggests that there may be less semantic variation within a language group than across different language groups.

7 Discussion and data analysis

The best accuracy achieved in our experiments is 71.19% on ES$_{subj}$ combinations. However, previous research suggests that error detection in lexical choice is a difficult task. For instance, Kochmar and Briscoe (2014) report that the agreement between human annotators on error detection in adjective–noun combinations is 86.50%.

We then qualitatively assessed the performance of our systems by analysing what types of errors

L1	Features	Acc	$\mathbf{P}_e$	R_e	F_{1_e}
RU$_{dobj}$	baseline	55.04	47.68	63.87	53.81
	ft$_{En}$	64.73	59.76	46.05	52.01
	+pmi$_{L1}$	65.15	60.63	45.77	52.16
RU$_{subj}$	baseline	53.30	44.77	61.09	51.29
	ft$_{En}$	61.84	54.63	35.81	43.22
	+pmi$_{L1}$	62.53	57.24	35.11	43.18
ES$_{dobj}$	baseline	55.25	51.67	76.79	61.21
	ft$_{En}$	64.06	62.30	56.01	58.98
	+pmi$_{L1}$	65.21	63.44	58.13	60.66
ES$_{subj}$	baseline	54.34	47.76	68.73	56.23
	ft$_{En}$	62.71	58.80	43.09	49.69
	+pmi$_{L1}$	62.44	58.46	41.71	48.60

Table 8: System performance (in %) using L1 and L2 lexico-semantic features, L1 → REL_L1.

the classifiers reliably detect and what types of errors the classifiers miss across all runs over the samples. Some of the most reliably identified errors in both RU and ES datasets include:

- verbs *offer*, *propose* and *suggest* which are often confused with each other. Correctly identified errors include **offer plan* vs. *suggest plan*, **propose work* vs. *offer work* and **suggest cost* vs. *offer cost*;
- verbs *demonstrate* and *show* where *demonstrate* is often used instead of *show* as in **chart demonstrates*;
- verbs *say* and *tell* particularly well identified with the ES model. Examples include **say idea* instead of *tell idea* and **tell goodbye* instead of *say goodbye*.

These examples represent lexical choice errors when selecting among near-synonyms, and violations of verb subcategorization frames. The error in **find solution* discussed throughout the paper is also reliably identified by the classifier across all runs. It is interesting to note that in the pair of verbs *do* and *make*, which are often confused with each other by both Russian and Spanish L1 speakers, errors involving *make* are identified more reliably than errors involving *do*: for example, **make business* is correctly identified as an error, while **do joke* is missed by the classifier.

Many of the errors missed by the classifier are context-dependent. Some of the most problematic errors involve errors in combinations with verbs like *be* and *become*. Such errors do not result from an L1 lexico-semantic transfer and it is not surprising that the classifiers miss them.

8 Conclusion

We have investigated whether lexico-semantic models from the native language are transferred to the second language, and what effect this transfer has on lexical choice in L2. We focused on two typologically different L1s – Russian and Spanish, and experimentally confirmed the hypothesis that statistical semantic models learned from these L1s significantly improve automatic error detection in L2 data produced by the speakers of the respective L1s. We also investigated whether the semantic models learned from particular L1s are portable to other languages, and in particular to languages that are typologically close to the investigated L1s. Our results demonstrate that L1 models improve the coverage of the error detection system on a range of other L1s.

Acknowledgments

We are grateful to the ACL reviewers for their helpful feedback. Ekaterina Kochmar's research is supported by Cambridge English Language Assessment via the ALTA Institute. Ekaterina Shutova's research is supported by the Leverhulme Trust Early Career Fellowship.

References

Bach E. and Chao W. 2008. *Semantic universals and typology*. In Chris Collins, Morten Christiansen and Shimon Edelman, eds., Language Universals (Oxford: Oxford University Press).

Baldwin T. and Kim S. N. 2010. *Multiword Expressions*. In Handbook of Natural Language Processing, Second Edition, N. Indurkhya and F. J. Damerau (eds.), pp. 267–292.

Ballesteros M., Herrera J., Francisco V., and Gervás P. 2010. *A Feasibility Study on Low Level Techniques for Improving Parsing Accuracy for Spanish Using Maltparser*. In Proceedings of the 6th Hellenic Conference on Artificial Intelligence: Theories, Models and Applications, pp. 39–48.

Baroni M., Bernardini S., Ferraresi A., and Zanchetta E. 2009. *The WaCky Wide Web: A Collection of Very Large Linguistically Processed Web-Crawled Corpora*. Language Resources and Evaluation, 43(3): 209–226.

Briscoe E., Carroll J., and Watson R. 2006. *The Second Release of the RASP System*. In Proceedings of the COLING/ACL-2006 Interactive Presentation Sessions, pp. 59–68.

Burnard L. 2007. *The British National Corpus, version 3 (BNC XML Edition)*. Distributed by Oxford University Computing Services on behalf of the BNC Consortium. `http://www.natcorp.ox.ac.uk/`.

Chang Y.C., Chang J.S., Chen H.J., and Liou H.C. 2012. *An automatic collocation writing assistant for Taiwanese EFL learners: A case of corpus-based NLP technology*. Computer Assisted Language Learning, 21(3), pp. 283–299.

Dahlmeier D. and Ng H.T. 2011. *Correcting Semantic Collocation Errors with L1-induced Paraphrases*. In Proceedings of the EMNLP-2011, pp. 107–117.

Futagi Y., Deane P., Chodorow M., and Tetreault J. 2009. *A computational approach to detecting collocation errors in the writing of non-native speakers of English*. Computer Assisted Language Learning, 21(4), pp. 353–367.

Joachims T. 1999. *Making Large-Scale SVM Learning Practical. Advances in Kernel Methods – Support Vector Learning*. B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press.

Kochmar E. and Briscoe T. 2014. *Detecting Learner Errors in the Choice of Content Words Using Compositional Distributional Semantics*. In Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers, pp. 1740–1751

Kochmar E. and Briscoe T. 2015. *Using Learner Data to Improve Error Correction in AdjectiveNoun Combinations*. In Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications, pp. 233-242.

Leacock C., Chodorow M., Gamon M. and Tetreault J. 2014. *Automated Grammatical Error Detection for Language Learners*. Morgan and Claypool Publishers.

Liu A. L.-E., Wible D., and Tsao N.-L. 2009. *Automated suggestions for miscollocations*. In Proceedings of the 4th Workshop on Innovative Use of NLP for Building Educational Applications, pp. 47–50.

Madnani N., Tetreault J., and Chodorow M. 2012. *Exploring Grammatical Error Correction with Not-So-Crummy Machine Translation*. In Proceedings of the 7th Workshop on the Innovative Use of NLP for Building Educational Applications, pp. 44-53.

Mendonca A., Jaquette D., Graff D., and DiPersio D. 2011. *Spanish Gigaword Third Edition*. Linguistic Data Consortium, Philadelphia.

Mikolov T., Sutskever I., Chen K., Corrado G., and Dean J. 2013. *Distributed Representations of Words and Phrases and their Compositionality*. In Proceedings of NIPS.

Mikolov T., Yih W.-T., and Zweig G. 2013. *Linguistic Regularities in Continuous Space Word Representations*. In Proceedings of NAACL HLT.

Miller G. A. 1995. *WordNet: A Lexical Database for English*. Communications of the ACM, 38(11): 39–41.

Mitchell J. and Lapata M. 2008. *Vector-based models of semantic composition*. In Proceedings of ACL, pp. 236–244.

Mitchell J. and Lapata M. 2010. *Composition in distributional models of semantics*. Cognitive Science, 34, pp. 1388–1429.

Nagata, R. and Nakatani, K. 2010. *Evaluating Performance of Grammatical Error Detection to Maximize Learning Effect*. In Proceedings of COLING (Posters), pp. 894-900.

Ng, H.T., Wu, S. M., Briscoe, T., Hadiwinoto, C., Susanto, R. H., Bryant, C. 2014. *The CoNLL-2014 Shared Task on Grammatical Error Correction*. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pp. 1–14.

Nicholls D. 2003. *The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT*. In Proceedings of the Corpus Linguistics conference, pp. 572–581.

Nivre J., Hall J., Nilsson J., Chanev A., Eryigit G., Kübler S., Marinov S., and Marsi E. 2007. *MaltParser: A language-independent system for data-driven dependency parsing*. Natural Language Engineering, 2(13):95–135.

Odlin T. 1989. *Language transfer: Cross-linguistic influence in language learning*. Cambridge University Press.

Östling R. and Knutsson O. 2009. *A corpus-based tool for helping writers with Swedish collocations*. In Proceedings of the Workshop on Extracting and Using Constructions in NLP, NODALIDA, pp. 28–33.

Park T., Lank E., Poupart P., and Terry M. 2008. *Is the sky pure today? AwkChecker: an assistive tool for detecting and correcting collocation errors*. In Proceedings of the 21st annual ACM symposium on User interface software and technology, pp. 121–130.

Rozovskaya A. and Roth D. 2010. *Generating Confusion Sets for Context-Sensitive Error Correction*. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pp. 961-970.

Rozovskaya A. and Roth D. 2011. *Algorithm Selection and Model Adaptation for ESL Correction Tasks*. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies – Volume 1, pp. 924–933.

Sharoff S. 2006. *Creating General-Purpose Corpora Using Automated Search Engine Queries*. WaCky! Working papers on the Web as Corpus, Marco Baroni and Silvia Bernardini (ed.).

Sharoff S. and Nivre J. 2011. *The proper place of men and machines in language technology Processing Russian without any linguistic knowledge*. Dialogue 2011, Russian Conference on Computational Linguistics.

Shei C.C. and Pain H. 2000. *An ESL Writer's Collocation Aid*. Computer Assisted Language Learning, 13(2), pp. 167–182.

Vecchi E., Baroni M. and Zamparelli R. 2011. *(Linear) maps of the impossible: Capturing semantic anomalies in distributional space*. In Proceedings of the DISCO Workshop at ACL-2011, pp. 1–9.

Wible H., Kwo C.-H., Tsao N.-L., Liu A., and Lin H.-L. 2003. *Bootstrapping in a language-learning environment*. Journal of Computer Assisted Learning, 19(4), pp. 90–102.

A CALL System for Learning Preposition Usage

John Lee
Department of
Linguistics and Translation
City University of Hong Kong
jsylee@cityu.edu.hk

Donald Sturgeon
Fairbank Center
for Chinese Studies
Harvard University
djs@dsturgeon.net

Mengqi Luo
Department of
Linguistics and Translation
City University of Hong Kong
mengqluo@cityu.edu.hk

Abstract

Fill-in-the-blank items are commonly featured in computer-assisted language learning (CALL) systems. An item displays a sentence with a blank, and often proposes a number of choices for filling it. These choices should include one correct answer and several plausible distractors. We describe a system that, given an English corpus, automatically generates distractors to produce items for preposition usage.

We report a comprehensive evaluation on this system, involving both experts and learners. First, we analyze the difficulty levels of machine-generated carrier sentences and distractors, comparing several methods that exploit learner error and learner revision patterns. We show that the quality of machine-generated items approaches that of human-crafted ones. Further, we investigate the extent to which mismatched L1 between the user and the learner corpora affects the quality of distractors. Finally, we measure the system's impact on the user's language proficiency in both the short and the long term.

1 Introduction

Fill-in-the-blank items, also known as gap-fill or cloze items, are a common form of exercise in computer-assisted language learning (CALL) applications. Table 1 shows an example item designed for teaching English preposition usage. It contains a sentence, "The objective is to kick the ball into the opponent's goal", with the preposition "into" blanked out; this sentence serves as the *stem* (or *carrier sentence*). It is followed by four choices for the blank, one of which is the *key* (i.e., the correct answer), and the other three are distractors. These choices enable the CALL application to provide immediate and objective feedback to the learner.

A high-quality item must meet multiple requirements. It should have a stem that is fluent and matches the reading ability of the learner; a blank that is appropriate for the intended pedagogical goal; exactly one correct answer among the choices offered; and finally, a number of distractors that seem plausible to the learner, and yet would each yield an incorrect sentence. Relying on language teachers to author these items is time consuming. Automatic generation of these items would not only expedite item authoring, but also potentially provide personalized items to suit the needs of individual learners. This paper addresses two research topics:

- How do machine-generated items compare with human-crafted items in terms of their quality?
- Do these items help improve the users' language proficiency?

For the first question, we focus on automatic generation of preposition distractors, comparing three different methods for distractor generation. One is based on word co-occurrence in standard

The objective is to kick the ball _____ the opponent's goal. (A) in (B) into (C) to (D) with

Table 1: An automatically generated fill-in-the-blank item, where "into" is the key, and the other three choices are distractors.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 984–993,
Berlin, Germany, August 7-12, 2016.

corpora; a second leverages error annotations in learner corpora; the third, a novel method, exploits learners' revision behavior. Further, we investigate the effect of tailoring distractors to the user's native language (L1). For the second question, we measure users' performance in the short and in the long term, through an experiment involving ten subjects, in multiple sessions tailored to their proficiency and areas of weakness.

Although a previous study has shown that learner error statistics can produce competitive items for prepositions on a narrow domain (Lee and Seneff, 2007), a number of research questions still await further investigation. Through both expert and learner evaluation, we will compare the quality of carrier sentences and the plausibility of automatically generated distractors against human-crafted ones. Further, we will measure the effect of mismatched L1 between the user and the learner corpora, and the short- and long-term impact on the user's preposition proficiency. To the best of our knowledge, this paper offers the most detailed evaluation to-date covering all these aspects.

The rest of the paper is organized as follows. Section 2 reviews previous work. Section 3 outlines the algorithms for generating the fill-in-the-blank items. Section 4 gives details about the experimental setup and evaluation procedures. Section 5 analyzes the results. Section 6 concludes the paper.

2 Previous Work

2.1 Distractor generation

Most research effort on automatic generation of fill-in-the-blank items has focused on vocabulary learning. In these items, the key is typically from an open-class part-of-speech (POS), e.g., nouns, verbs, or adjectives.

To ensure that the distractor results in an incorrect sentence, the distractor must rarely, or never, collocate with other words in the carrier sentence (Liu et al., 2005). To ensure the plausibility of the distractor, most approaches require it to be semantically close to the key, as determined by a thesaurus (Sumita et al., 2005; Smith et al., 2010), an ontology (Karamanis et al., 2006), rules handcrafted by experts (Chen et al., 2006), or context-sensitive inference rules (Zesch and Melamud, 2014); or to have similar word frequency (Shei, 2001; Brown et al., 2005). Sakaguchi et al. (2013) applied machine learning methods to select verb distractors, and showed that they resulted in items that can better predict the user's English proficiency level.

Less attention has been paid to items for closed-class POS, such as articles, conjunctions and prepositions, which learners also often find difficult (Dahlmeier et al., 2013). For these POS, the standard algorithms based on semantic relatedness for open-class POS are not applicable. Lee and Seneff (2007) reported the only previous study on using learner corpora to generate items for a closed-class POS. They harvested the most frequent preposition errors in a corpus of Japanese learners of English (Izumi et al., 2003), but performed an empirical evaluation with native Chinese speakers on a narrow domain.

We expand on this study in several dimensions. First, carrier sentences, selected from the general domain rather than a specific one, will be analyzed in terms of their difficulty level. Second, distractor quality will be evaluated not only by learners but also by experts, who give scores based on their plausibility; in contrast to most previous studies, their quality will be compared with the human gold standard. Thirdly, the effect of mismatched L1 will also be measured.

2.2 Learner error correction

There has been much recent research on automatic correction of grammatical errors. Correction of preposition usage errors, in particular, has received much attention. Our task can be viewed as the inverse of error correction — ensuring that the distractor yields an incorrect sentence — with the additional requirement on the plausibility of the distractor.

Most approaches in automatic grammar correction can be classified as one of three types, according to the kind of statistics on which the system is trained. Some systems are trained on examples of correct usage (Tetreault and Chodorow, 2008; Felice and Pulman, 2009). Others are trained on examples of pairs of correct and incorrect usage, either retrieved from error-annotated learner corpora (Han et al., 2010; Dahlmeier et al., 2013) or simulated (Lee and Seneff, 2008; Foster and Andersen, 2009). More recently, a system has been trained on revision statistics from Wikipedia (Cahill et al., 2013). We build on all three paradigms, using standard English cor-

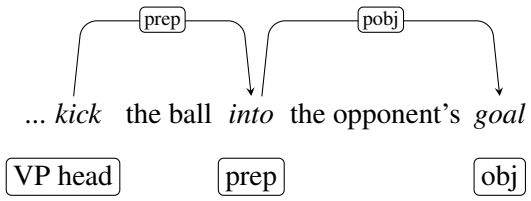


Figure 1: Parse tree for the carrier sentence in Table 1. Distractors are generated on the basis of the prepositional object ("obj") and the NP/VP head to which the prepositional phrase is attached (Section 3).

pora (Section 3.1), error-annotated learner corpora (Section 3.2) and learner revision corpora (Section 3.3) as resources to predict the most plausible distractors.

3 Item generation

The system assumes as input a set of English sentences, which are to serve as candidates for carrier sentences. In each candidate sentence, the system scans for prepositions, and extracts two features from the linguistic context of each preposition:

- The **prepositional object**. In Figure 1, for example, the word "goal" is the prepositional object of the key, "into".
- The head of the noun phrase or verb phrase (**NP/VP head**) to which the prepositional phrase (PP) is attached. In Figure 1, the PP "into the opponent's goal" is attached to the VP head "kick".

The system passes these two features to the following methods to generate distractors.[1] If all three methods are able to return a distractor, the preposition qualifies to serve as the key. If more than one key is found, the system randomly chooses one of them.

In the rest of this paper, we will sometimes abbreviate these three methods as the "Co-occur" (Section 3.1), "Error" (Section 3.2), and "Revision" (Section 3.3) methods, respectively.

3.1 Co-occurrence method

Proposed by Lee and Seneff (2007), this method requires co-occurrence statistics from a large corpus of well-formed English sentences.

[1]We do not consider errors where a preposition should be inserted or deleted.

Co-occurrence method ("Co-occur")
... *kicked* the chair *with* ...
... *kicked* the can *with* ...
... *with* the *goal* of ...
Learner error method ("Error")
... *kicked* it <error>*in*</error> the *goal*.
... *kick* the ball <error>*in*</error> the other team's *goal*.
Learner revision method ("Revision")
... *kick* the ball ~~to~~ into his own *goal*.
... *kick* the ball ~~to~~ towards his own *goal*.

Table 2: The Co-occurrence Method (Section 3.1) generates "with" as the distractor for the carrier sentence in Figure 1; the Learner Error Method (Section 3.2) generates "in"; the Learner Revision Method (Section 3.3) generates "to".

This method first retrieves all prepositions that co-occur with *both* the prepositional object and the NP/VP head in the carrier sentence. These prepositions are removed from consideration as distractors, since they would likely yield a correct sentence. The remaining candidates are those that co-occur with *either* the prepositional object *or* the NP/VP head, but not both. The more frequently the candidate co-occurs with either of these words, the more plausible it is expected to appear to a learner. Thus, the candidate with the highest co-occurrence frequency is chosen as the distractor. As shown in Table 2, this method generates the distractor "with" for the carrier sentence in Figure 1, since many instances of "kick ... with" and "with ... goal" are attested.

3.2 Learner error method

This method requires examples of English sentences from an error-annotated learner corpus. The corpus must mark wrong preposition usage, but does not need to provide corrections for the errors.

This method first retrieves all PPs that have the given prepositional object and are attached to the given NP/VP head. It then computes the frequency of prepositions that head these PPs and are marked as wrong. The one that is most frequently marked as wrong is chosen as the distractor. As shown in Table 2, this method generates the distractor "in" for the carrier sentence in Figure 1, since it is often marked as an error.

3.3 Learner revision method

It is expensive and time consuming to annotate learner errors. As an alternative, we exploit the revision behavior of learners in their English writing. This method requires draft versions of texts written by learners. In order to compute statistics on how often a preposition in an earlier draft ("draft n") is replaced with another one in the later draft ("draft $n + 1$"), the sentences in successive drafts must be sentence- and word-aligned.

This method scans for PPs that have the given prepositional object and are attached to the given NP/VP head. For all learner sentences in draft n that contain these PPs, it consults the sentences in draft $n+1$ to which they are aligned; it retains only those sentences whose prepositional object and the NP/VP head remain unchanged, but whose preposition has been replaced by another one. Among these sentences, the method selects the preposition that is most frequently edited between two drafts. Our assumption is that frequent editing implies a degree of uncertainty on the part of the learner as to which of these prepositions is in fact correct, thus suggesting that they may be effective distractors. As shown in Table 2, this method generates the distractor "to" for the carrier sentence in Figure 1, since it is most often edited in the given linguistic context. This study is the first to exploit a corpus of learner revision history for item generation.[2]

4 Experimental setup

In this section, we first describe our datasets (Section 4.1) and the procedure for item generation (Section 4.2). We then give details on the expert evaluation (Section 4.3) and the learner evaluation (Section 4.4).

4.1 Data

Carrier sentences. We used sentences in the English portion of the Wikicorpus (Reese et al., 2010) as carrier sentences. To avoid selecting stems with overly difficult vocabulary, we ranked the sentences in terms of their most difficult word. We measured the difficulty level of a word firstly with the graded English vocabulary lists compiled by the Hong Kong Education Bureau (EDB, 2012); and secondly, for words not occurring in any of these lists, with frequency counts derived from the Google Web Trillion Word Corpus.[3] In order to retrieve the prepositional object and the NP/VP head (cf. Section 3), we parsed the Wikicorpus, as well as the corpora mentioned below, with the Stanford parser (Manning et al., 2014).

Co-occurrence method ("Co-occur"). The statistics for the Co-occurrence method were also based on the English portion of Wikicorpus.

Learner Revision method ("Revision"). We used an 8-million-word corpus of essay drafts written by Chinese learners of English (Lee et al., 2015). This corpus contains over 4,000 essays, with an average of 2.7 drafts per essay. The sentences and words between successive drafts have been automatically aligned.

Learner Error method ("Error"). In addition to the corpus of essay drafts mentioned above, we used two other error-annotated learner corpora. The NUS Corpus of Learner English (NUCLE) contains one million words of academic writing by students at the National University of Singapore (Dahlmeier et al., 2013). The EF-Cambridge Open Language Database (EFCAMDAT) contains over 70 million words from 1.2 million assignments written by learners from a variety of linguistic background (Geertzen et al., 2013). A subset of the database has been error-annotated. We made use of the writings in this subset that were produced by students from China and Russia.

Human items ("Textbook"). To provide a comparison with human-authored items, we used the practise tests for preposition usage offered in an English exercise book designed for intermediate and advanced learners (Watcyn-Jones and Allsop, 2000). From the 50 tests in a variety of formats, we harvested 56 multiple-choice items, all of which had one key and three distractors.

4.2 Item generation procedure

We gathered three sets of 400 carrier sentences, for use in three evaluation sessions (see Section 4.4). Each sentence in Set 1 has one counterpart in Set 2 and one counterpart in Set 3 that have the same key, NP/VP head and prepositional object. We will refer to the items created from these counterpart carrier sentences as "similar" items. We will use these "similar" items to measure the learning impact on the subjects.

Each item has one key and distractors generated

[2] A similar approach, using revision statistics in Wikipedia, has been used for the purpose of correcting preposition errors (Cahill et al., 2013).

[3] http://norvig.com/ngrams/

by each of the three methods. For about half of the items, the three methods complemented one another to offer three distinct distractors. In the other half, two of the methods yielded the same distractor, resulting in only two distractors for those items. In Set 1, for control purposes, 56 of the items were replaced with the human items.

4.3 Expert evaluation procedure

Two professional English teachers (henceforth, the "experts") examined each of the 400 items in Set 1. They annotated each item, and each choice in the item, as follows.

For each item, the experts labeled its difficulty level in terms of the preposition usage being tested in the carrier sentence. They did not know whether the item was human-authored or machine-generated. Based on their experience in teaching English to native speakers of Chinese, they labeled each item as suitable for those in "Grades 1-3", "Grades 4-6", "Grades 7-9", "Grades 10-12", or ">Grade 12". We mapped these five categories to integers — 2, 5, 8, 11 and 13, respectively — for the purpose of calculating difficulty scores.

For each choice in the item, the experts judged whether it is correct or incorrect. They did not know whether each choice was the key or a distractor. They may judge one, multiple, or none of the choices as correct. For an incorrect choice, they further assessed its plausibility as a distractor, again from their experience in teaching English to native speakers of Chinese. They may label it as "Plausible", "Somewhat plausible", or "Obviously wrong".

4.4 Learner evaluation procedure

Ten university students (henceforth, the "learners") took part in the evaluation. They were all native Chinese speakers who did not major in English. The evaluation consisted of three one-hour sessions held on different days. At each session, the learner attempted 80 items on a browser-based application (Figure 2). The items were distributed in these sessions as follows.

Session 1. The 400 items in Set 1 were divided into 5 groups of 80 items, with 11 to 12 human items in each group. The items in each group had comparable difficulty levels as determined by the experts, with average scores ranging from 7.9 to 8.1. Each group was independently attempted by two learners. The system recorded the items to which the learner gave wrong answers; these will be referred to as the "wrong items". Among the items to which the learner gave correct answers, the system randomly set aside 10 items; these will be referred to as "control items".

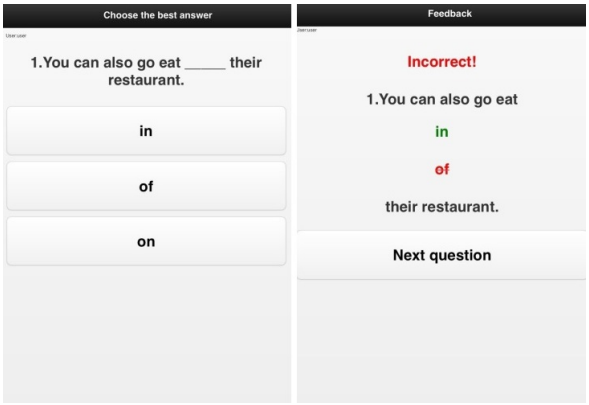


Figure 2: Interface for the learner evaluation. On the left, the learner selects a choice by tapping on it; on the right, the learner receives feedback.

Session 2. To measure the short-term impact, Session 2 was held on the day following Session 1. Each learner attempted 80 items, drawn from Set 2. These items were personalized according to the "wrong items" of the individual learner. For example, if a learner had 15 "wrong items" from Session 1, he or she then received 15 similar items[4] from Set 2. In addition, he or she also received ten items that were similar to the "control items" from Session 1. The remaining items were drawn randomly from Set 2. As in Session 1, the system noted the "wrong items" and set aside ten "control items".

Session 3. To test the long-term effect of these exercises, Session 3 was held two weeks after Session 2. Each learner attempted another 80 items, drawn from Set 3. These 80 items were chosen in the same manner as in Session 2.

5 Results

We first report inter-annotator agreement between the two experts on the difficulty levels of the carrier sentences and the distractors (Section 5.1). We then compare the difficulty levels of the human- and machine-generated items (Section 5.2). Next, we analyze the reliability and difficulty[5] of the

[4] See definition of "similar" in Section 4.2.

[5] Another metric, "validity", measures the ability of the distractor to discriminate between students of different proficiency levels. This metric is relevant for items intended for

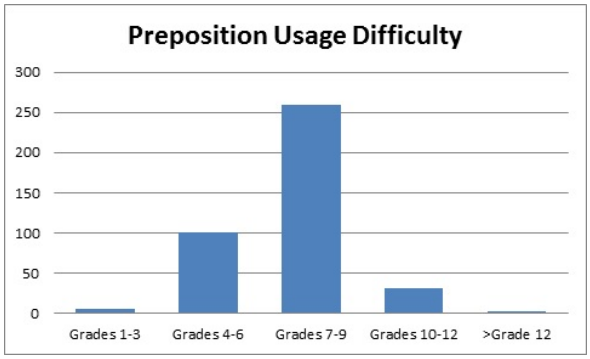


Figure 3: The difficulty level of the items in Set 1, as annotated by the experts.

automatically generated distractors (Sections 5.3 and 5.4), and the role of the native language (Section 5.5). Finally, we measure the impact on the learners' preposition proficiency (Section 5.6).

5.1 Inter-annotator agreement

For estimating the difficulty level of the preposition usage in the carrier sentences, the experts reached "substantial" agreement with kappa at 0.765 (Landis and Koch, 1977). In deciding whether a choice is correct or incorrect, the experts reached "almost perfect" agreement with kappa at 0.977. On the plausibility of the distractors, they reached "moderate" agreement with kappa at 0.537. The main confusion was between the categories "Obviously wrong" and "Somewhat plausible".

On the whole, expert judgment tended to correlate with actual behavior of the learners. For distractors considered "Plausible" by both experts, 63.6% were selected by the learners. In contrast, for those considered "Obviously wrong" by both experts, only 11.8% attracted any learner.

5.2 Carrier sentence difficulty

Figure 3 shows the distribution of difficulty level scores for the preposition usage in carrier sentences. Most items were rated as "Grades 7-9", with "Grades 4-6" being the second largest group.

A common concern over machine-generated items is whether the machine can create or select the kind of carrier sentences that illustrate challenging or advanced preposition usage, compared to those crafted by humans. In our system, the preposition errors and revisions in the learner corpora — as captured by the NP/VP head and the prepositional object — effectively served as the filter for selecting carrier sentences. Some of these errors and revisions may well be careless or trivial mistakes, and may not necessarily lead to the selection of appropriate carrier sentences.

To answer this question, we compared the difficulty levels of preposition usage in the machine-generated and human-crafted items. The average difficulty score for the human items was 8.7, meaning they were suitable for those in Grade 8. The average for the machine-generated items were lower, at 7.2. This result suggests that our system can select carrier sentences that illustrate challenging preposition usage, at a level that is only about 1.5 grade points below those designed by humans.

5.3 Distractor reliability

A second common concern over machine-generated items is whether their distractors might yield correct sentences. When taken out of context, a carrier sentence often admits multiple possible answers (Tetreault and Chodorow, 2008; Lee et al., 2009). In this section, we compare the performance of the automatic distractor generation methods against humans.

A distractor is called "reliable" if it yields an incorrect sentence. The Learner Revision method generated the most reliable distractors[6]; on average, 97.4% of the distractors were judged incorrect by both experts (Table 3). The Co-occurrence method ranked second at 96.1%, slightly better than those from the Learner Error method. Many distractors from the Learner Error method indeed led to incorrect sentences in their original contexts, but became acceptable when their carrier sentences were read in isolation. Items with unreliable distractors were excluded from the learner evaluation.

Surprisingly, both the Learner Revision and Co-occurrence methods outperformed the humans. Distractors in some of the human items did indeed yield sentences that were technically correct, and were therefore deemed "unreliable" by the experts. In many cases, however, these distractors were accompanied with keys that provided more natural choices. These items, therefore, remained valid.

assessment purposes (Brown et al., 2005; Sakaguchi et al., 2013) rather than self-learning.

[6]The difference with the Co-occurrence method is not statistically significant, in part due to the small sample size.

Method	Reliable distractor
Co-occur	96.1%
Error	95.6%
Revision	**97.4%**
Textbook	95.8%

Table 3: Distractors judged reliable by both experts.

5.4 Distractor difficulty

In the context of language learning, an item can be considered more useful if one of its distractors elicits a wrong choice from the learner, who would then receive corrective feedback. In this section, we compare the "difficulty" of the distractor generated by the various methods, in terms of their ability to attract the learners.

Expert evaluation. The two methods based on learner statistics produced the highest-quality distractors (Table 4). The Learner Error method had the highest rate of plausible distractors (51.2%) and the lowest rate of obviously wrong ones (22.0%). In terms of the number of distractors considered "Plausible", this method significantly outperformed the Learner Revision method.[7]

According to Table 4, all three automatic methods outperformed the humans in terms of the number of distractors rated "Plausible". This comparison, however, is not entirely fair, since the human items always supplied three distractors, whereas about half of the machine-generated items supplied only two, when two of the methods returned the same distractor.

An alternate metric is to compute the average number of distractors rated "Plausible" *per item*. On average, the human items had 0.91 plausible distractors; in comparison, the machine-generated items had 1.27. This result suggests that automatic generation of preposition distractors can perform at the human level.

Learner evaluation. The most direct way to evaluate the difficulty of a distractor is to measure how often a learner chose it. The contrast is less clear cut in this evaluation. Overall, the learners correctly answered 76.2% of the machine-generated items, and 75.5% of the human items, suggesting that the human distractors were more challenging. One must also take into account, however, the fact that the carrier sentences are more difficult in the human items than in the machine-generated ones. Broadly speaking, the machine-generated distractors were almost as successful as those authored by humans.

[7] $p < 0.05$ by McNemar's test, for both expert annotators.

Method	Plausible	Somewhat plausible	Obviously wrong
Co-occur	34.6%	31.5%	33.9%
Error	**51.2%**	**26.8%**	**22.0%**
Revision	45.4%	28.5%	26.1%
Textbook	31.4%	34.2%	34.5%

Table 4: Plausibility judgment of distractors by experts.

Consistent with the experts' opinion (Table 4), the Learner Error method was most successful among the three automatic methods (Table 5). The learner selection rate of its distractors was 13.5%, which was significantly higher[8] than its closest competitor, the Learner Revision method, at 9.5%. The Co-occurrence method ranked last, at 9.2%. It is unfortunately difficult to directly compare these rates with that of the human distractors, which they were offered in different carrier sentences.

5.5 Impact of L1

We now turn our attention to the relation between the native language (L1) of the user, and that of the learner corpora used for training the system. Specifically, we wish to measure the gain, if any, in matching the L1 of the user with the L1 of the learner corpora. To this end, for the Learner Error method, we generated distractors from the EF-Cambridge corpus with two sets of statistics: one harvested from the portion of the corpus with writings by Chinese students, the others from the portion by Russian students.

Expert evaluation. Table 6 contrasts the experts' plausibility judgment on distractors generated from these two sets. Chinese distractors were

[8] $p < 0.05$ by McNemar's test.

Method	Learner selection rate
Co-occur	9.2%
Error	**13.5%**
Revision	9.5%

Table 5: Percentage of distractors selected by learners.

Method	Plausible	Somewhat plausible	Obviously wrong
Chinese	**57.7%**	**24.0%**	**18.3%**
Russian	55.3%	22.0%	22.7%

Table 6: Plausibility judgment of distractors generated from the Chinese and Russian portions of the EF-Cambridge corpus, by experts.

Difficulty level	Retention rate
Below 6	74.0%
6-8	71.3%
9-11	60.0%
12 or above	25%

Table 7: Retention rate for items at different levels of difficulty.

slightly more likely to be rated "plausible" than the Russian ones, and less likely to be rated "obviously wrong".[9] The gap between the two sets of distractors was smaller than may be expected.

Learner evaluation. The difference was somewhat more pronounced in terms of the learners' behavior. The learners selected Chinese distractors, which matched their L1, 29.9% of the time over the three sessions. In contrast, they fell for the Russian distractors, which did not match their L1, only 25.1% of the time. This result confirms the intuition that matching L1 improves the plausibility of the distractors, but the difference was nonetheless relatively small. This result suggets that it might be worth paying the price for mismatched L1s, in return for a much larger pool of learner statistics.

5.6 Impact on learners

In this section, we consider the impact of these exercises on the learners. The performance of the learners was rather stable across all sessions; their average scores in the three sessions were 73.0%, 73.6% and 69.9%, respectively. It is difficult, however, to judge from these scores whether the learners benefited from the exercises, since the composition of the items differed for each session.

Instead, we measured how often the learners retain the system feedback. More specifically, if the learner chose a distractor and received feedback (cf. Figure 2), how likely would he or she succeed in choosing the key in a "similar"[10] item in a subsequent session.

We compared the learners' responses between Sessions 1 and 2 to measure the short-term impact, and between Sessions 2 and 3 to measure the long-term impact. In Session 2, when the learners attempted items that were "similar" to their "wrong items" from Session 1, they succeeded in choosing the key in 72.4% of the cases.[11] We refer to this figure as the "retention rate", in this case over the one-day period between the two sessions. The retention rate deteriorated over a longer term. In Session 3, when the learners attempted items that were "similar" to their "wrong items" from Session 2, which took place two weeks before, they succeeded only in 61.5% of the cases.[12]

Further, we analyzed whether the difficulty level of the items affected their retention rate. Statistics in Table 7 show that the rate varied widely according to the difficulty level of the "wrong items". Difficult items, at Grade 12 or beyond, proved hardest to learn, with a retention rate of only 25%. At the other end of the spectrum, those below Grade 6 were retained 74% of the time. This points to the need for the system to reinforce difficult items more frequently.

6 Conclusions

We have presented a computer-assisted language learning (CALL) system that automatically creates fill-in-the-blank items for prepositions. We found that the preposition usage tested in automatically selected carrier sentences were only slightly less challenging than those crafted by humans. We compared the performance of three methods for distractor generation, including a novel method that exploits learner revision statistics. The method based on learner error statistics yielded the most plausible distractors, followed by the one based on learner revision statistics. The items produced jointly by these automatic methods, in both expert and learner evaluations, rivalled the quality of human-authored items. Further, we evaluated the extent to which mismatched

[9] Data sparseness prevented us from generating both Chinese and Russian distractors for the same carrier sentences for evaluation. These statistics are therefore not controlled with regard to the difficulty level of the sentences.

[10] See definition of "similar" in Section 4.2.

[11] As a control, the retention rate for correctly answered items in Session 1 was 80% in Session 2.

[12] As a control, the retention rate for correctly answered items in Session 2 was 69.0% in Session 3.

native language (L1) affects distractor plausibility. Finally, in a study on the short- and long-term impact on the learners, we showed that difficult items had lower retention rate. In future work, we plan to conduct larger-scale evaluations to further validate these results, and to apply these methods on other common learner errors.

Acknowledgments

We thank NetDragon Websoft Holding Limited for their assistance with system evaluation, and the reviewers for their very helpful comments. This work was partially supported by an Applied Research Grant (Project no. 9667115) from City University of Hong Kong.

References

Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. 2005. Automatic Question Generation for Vocabulary Assessment. In *Proc. HLT-EMNLP*.

Aoife Cahill, Nitin Madnani, Joel Tetreault, and Diane Napolitano. 2013. Robust Systems for Preposition Error Correction using Wikipedia Revisions. In *Proc. NAACL-HLT*.

Chia-Yin Chen, Hsien-Chin Liou, and Jason S. Chang. 2006. FAST: An Automatic Generation System for Grammar Tests. In *Proc. COLING/ACL Interactive Presentation Sessions*.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proc. 8th Workshop on Innovative Use of NLP for Building Educational Applications*.

EDB. 2012. *Enhancing English Vocabulary Learning and Teaching at Secondary Level*. http://www.edb.gov.hk/vocab_learning_sec.

Rachele De Felice and Stephen Pulman. 2009. Automatic Detection of Preposition Errors in Learner Writing. *CALICO Journal*, 26(3):512–528.

Jennifer Foster and Øistein E. Andersen. 2009. GenERRate: Generating Errors for Use in Grammatical Error Detection. In *Proc. 4th Workshop on Innovative Use of NLP for Building Educational Applications*.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic Linguistic Annotation of Large Scale L2 Databases: The EF-Cambridge Open Language Database (EFCAMDAT). In *Proc. 31st Second Language Research Forum (SLRF)*.

Na-Rae Han, Joel Tetreault, Soo-Hwa Lee, and Jin-Young Ha. 2010. Using Error-annotated ESL Data to Develop an ESL Error Correction System. In *Proc. LREC*.

Emi Izumi, Kiyotaka Uchimoto, Toyomi Saiga, Thepchai Supnithi, and Hitoshi Isahara. 2003. Automatic Error Detection in the Japanese Learners' English Spoken Data. In *Proc. ACL*.

Nikiforos Karamanis, Le An Ha, and Ruslan Mitkov. 2006. Generating Multiple-Choice Test Items from Medical Text: A Pilot Study. In *Proc. 4th International Natural Language Generation Conference*.

J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33:159–174.

John Lee and Stephanie Seneff. 2007. Automatic generation of cloze items for prepositions. In *Proc. Interspeech*.

John Lee and Stephanie Seneff. 2008. Correcting Misuse of Verb Forms. In *Proc. ACL*.

John Lee, Joel Tetreault, and Martin Chodorow. 2009. Human Evaluation of Article and Noun Number Usage: Influences of Context and Construction Variability. In *Proc. Linguistic Annotation Workshop*.

John Lee, Chak Yan Yeung, Amir Zeldes, Marc Reznicek, Anke Lüdeling, and Jonathan Webster. 2015. CityU Corpus of Essay Drafts of English Language Learners: a Corpus of Textual Revision in Second Language Writing. *Language Resources and Evaluation*, 49(3):659–683.

Chao-Lin Liu, Chun-Hung Wang, Zhao-Ming Gao, and Shang-Ming Huang. 2005. Applications of Lexical Information for Algorithmically Composing Multiple-Choice Cloze Items. In *Proc. 2nd Workshop on Building Educational Applications Using NLP*, pages 1–8.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proc. ACL System Demonstrations*, pages 55–60.

Samuel Reese, Gemma Boleda, Montse Cuadros, Lluís Padró, and German Rigau. 2010. Wikicorpus: A Word-Sense Disambiguated Multilingual Wikipedia Corpus. In *Proc. LREC*.

Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative Approach to Fill-in-the-Blank Quiz Generation for Language Learners. In *Proc. ACL*.

Chi-Chiang Shei. 2001. FollowYou!: An Automatic Language Lesson Generation System. *Computer Assisted Language Learning*, 14(2):129–144.

Simon Smith, P. V. S. Avinesh, and Adam Kilgarriff. 2010. Gap-fill Tests for Language Learners: Corpus-Driven Item Generation. In *Proc. 8th International Conference on Natural Language Processing (ICON)*.

Eiichiro Sumita, Fumiaki Sugaya, and Seiichi Yamamoto. 2005. Measuring Non-native Speakers Proficiency of English by Using a Test with Automatically-Generated Fill-in-the-Blank Questions. In *Proc. 2nd Workshop on Building Educational Applications using NLP*.

Joel Tetreault and Martin Chodorow. 2008. The Ups and Downs of Preposition Error Detection in ESL Writing. In *Proc. COLING*.

Peter Watcyn-Jones and Jake Allsop. 2000. *Test Your Prepositions*. Penguin Books Ltd.

Torsten Zesch and Oren Melamud. 2014. Automatic Generation of Challenging Distractors Using Context-Sensitive Inference Rules. In *Proc. Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*.

A Persona-Based Neural Conversation Model

Jiwei Li[1*] **Michel Galley**[2] **Chris Brockett**[2]
Georgios P. Spithourakis[3*] **Jianfeng Gao**[2] **Bill Dolan**[2]

[1]Stanford University, Dept. of Computer Science
jiweil@stanford.edu
[2]Microsoft Research, Redmond
{mgalley,chrisbkt,jfgao,billdol}@microsoft.com
[3]University College London, Dept. of Computer Science
g.spithourakis@cs.ucl.ac.uk

Abstract

We present persona-based models for handling the issue of speaker consistency in neural response generation. A speaker model encodes personas in distributed embeddings that capture individual characteristics such as background information and speaking style. A dyadic speaker-addressee model captures properties of interactions between two interlocutors. Our models yield qualitative performance improvements in both perplexity and BLEU scores over baseline sequence-to-sequence models, with similar gains in speaker consistency as measured by human judges.

1 Introduction

As conversational agents gain traction as user interfaces, there has been growing research interest in training naturalistic conversation systems from large volumes of human-to-human interactions (Ritter et al., 2011; Sordoni et al., 2015; Vinyals and Le, 2015; Li et al., 2016). One major issue for these data-driven systems is their propensity to select the response with greatest likelihood—in effect a consensus response of the humans represented in the training data. Outputs are frequently vague or non-committal (Li et al., 2016), and when not, they can be wildly inconsistent, as illustrated in Table 1.

message	Where do you live now?
response	I live in Los Angeles.
message	In which city do you live now?
response	I live in Madrid.
message	In which country do you live now?
response	England, you?
message	Where were you born?
response	I was born in Canada.
message	Where are you from?
response	England, you?
message	Where did you grow up?
response	I grew up in Texas.
message	How old are you?
response	16 and you?
message	What's your age?
response	18.
message	What is your major?
response	I'm majoring in psychology
message	What did you study in college?
response	English lit.

Table 1: Inconsistent responses generated by a 4-layer SEQ2SEQ model trained on 25 million Twitter conversation snippets.

In this paper, we address the challenge of consistency and how to endow data-driven systems with the coherent "persona" needed to model human-like behavior, whether as personal assistants, personalized avatar-like agents, or game characters.[1] For present purposes, we will define PERSONA as the character that an artificial agent, as actor, plays or performs during conversational interactions. A persona can be viewed as a composite of elements of identity (background facts or user profile), language behavior, and interaction style. A persona is also adaptive, since an agent may need to present different facets to different human interlocutors depending on the interaction.

Fortunately, neural models of conversation generation (Sordoni et al., 2015; Shang et al., 2015; Vinyals and Le, 2015; Li et al., 2016) provide a straightforward mechanism for incorporating personas as embeddings. We therefore explore two per-

* The entirety of this work was conducted at Microsoft.

[1](Vinyals and Le, 2015) suggest that the lack of a coherent personality makes it impossible for current systems to pass the Turing test.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 994–1003,
Berlin, Germany, August 7-12, 2016.

sona models, a single-speaker SPEAKER MODEL and a dyadic SPEAKER-ADDRESSEE MODEL, within a sequence-to-sequence (SEQ2SEQ) framework (Sutskever et al., 2014). The Speaker Model integrates a speaker-level vector representation into the target part of the SEQ2SEQ model. Analogously, the Speaker-Addressee model encodes the interaction patterns of two interlocutors by constructing an interaction representation from their individual embeddings and incorporating it into the SEQ2SEQ model. These persona vectors are trained on human-human conversation data and used at test time to generate personalized responses. Our experiments on an open-domain corpus of Twitter conversations and dialog datasets comprising TV series scripts show that leveraging persona vectors can improve relative performance up to 20% in BLEU score and 12% in perplexity, with a commensurate gain in consistency as judged by human annotators.

2 Related Work

This work follows the line of investigation initiated by Ritter et al. (2011) who treat generation of conversational dialog as a statistical machine translation (SMT) problem. Ritter et al. (2011) represents a break with previous and contemporaneous dialog work that relies extensively on hand-coded rules, typically either building statistical models on top of heuristic rules or templates (Levin et al., 2000; Young et al., 2010; Walker et al., 2003; Pieraccini et al., 2009; Wang et al., 2011) or learning generation rules from a minimal set of authored rules or labels (Oh and Rudnicky, 2000; Ratnaparkhi, 2002; Banchs and Li, 2012; Ameixa et al., 2014; Nio et al., 2014; Chen et al., 2013). More recently (Wen et al., 2015) have used a Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) to learn from unaligned data in order to reduce the heuristic space of sentence planning and surface realization.

The SMT model proposed by Ritter et al., on the other hand, is end-to-end, purely data-driven, and contains no explicit model of dialog structure; the model learns to converse from human-to-human conversational corpora. Progress in SMT stemming from the use of neural language models (Sutskever et al., 2014; Gao et al., 2014; Bahdanau et al., 2015; Luong et al., 2015) has inspired efforts to extend these neural techniques to SMT-based conversational response generation. Sordoni et al. (2015) augments Ritter et al. (2011) by rescoring outputs using a SEQ2SEQ model conditioned on conversation history. Other researchers have recently used SEQ2SEQ to directly generate responses in an end-to-end fashion without relying on SMT phrase tables (Serban et al., 2015; Shang et al., 2015; Vinyals and Le, 2015). Serban et al. (2015) propose a hierarchical neural model aimed at capturing dependencies over an extended conversation history. Recent work by Li et al. (2016) measures mutual information between message and response in order to reduce the proportion of generic responses typical of SEQ2SEQ systems. Yao et al. (2015) employ an intention network to maintain the relevance of responses.

Modeling of users and speakers has been extensively studied within the standard dialog modeling framework (e.g., (Wahlster and Kobsa, 1989; Kobsa, 1990; Schatztnann et al., 2005; Lin and Walker, 2011)). Since generating meaningful responses in an open-domain scenario is intrinsically difficult in conventional dialog systems, existing models often focus on generalizing character style on the basis of qualitative statistical analysis (Walker et al., 2012; Walker et al., 2011). The present work, by contrast, is in the vein of the SEQ2SEQ models of Vinyals and Le (2015) and Li et al. (2016), enriching these models by training persona vectors directly from conversational data and relevant side-information, and incorporating these directly into the decoder.

3 Sequence-to-Sequence Models

Given a sequence of inputs $X = \{x_1, x_2, ..., x_{n_X}\}$, an LSTM associates each time step with an input gate, a memory gate and an output gate, respectively denoted as i_t, f_t and o_t. We distinguish e and h where e_t denotes the vector for an individual text unit (for example, a word or sentence) at time step t while h_t denotes the vector computed by the LSTM model at time t by combining e_t and h_{t-1}. c_t is the cell state vector at time t, and σ denotes the sigmoid function. Then, the vector representation h_t for each time step t is given by:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \end{bmatrix} \quad (1)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \quad (2)$$

$$h_t^s = o_t \cdot \tanh(c_t) \quad (3)$$

where W_i, W_f, W_o, $W_l \in \mathbb{R}^{K \times 2K}$. In SEQ2SEQ generation tasks, each input X is paired with a sequence of outputs to predict: $Y = \{y_1, y_2, ..., y_{n_Y}\}$. The LSTM defines a distribution over outputs and sequentially predicts tokens using a softmax function:

$$p(Y|X) = \prod_{t=1}^{n_y} p(y_t|x_1, x_2, ..., x_t, y_1, y_2, ..., y_{t-1}) = \prod_{t=1}^{n_y} \frac{\exp(f(h_{t-1}, e_{y_t}))}{\sum_{y'} \exp(f(h_{t-1}, e_{y'}))}$$

where $f(h_{t-1}, e_{y_t})$ denotes the activation function between h_{t-1} and e_{y_t}. Each sentence terminates with a special end-of-sentence symbol *EOS*. In keeping with common practices, inputs and outputs use different LSTMs with separate parameters to capture different compositional patterns.

During decoding, the algorithm terminates when an *EOS* token is predicted. At each time step, either a greedy approach or beam search can be adopted for word prediction.

4 Personalized Response Generation

Our work introduces two persona-based models: the Speaker Model, which models the personality of the respondent, and the Speaker-Addressee Model which models the way the respondent adapts their speech to a given addressee — a linguistic phenomenon known as lexical entrainment (Deutsch and Pechmann, 1982).

4.1 Notation

For the response generation task, let M denote the input word sequence (message) $M = \{m_1, m_2, ..., m_I\}$. R denotes the word sequence in response to M, where $R = \{r_1, r_2, ..., r_J, \textit{EOS}\}$ and J is the length of the response (terminated by an *EOS* token). r_t denotes a word token that is associated with a K dimensional distinct word embedding e_t. V is the vocabulary size.

4.2 Speaker Model

Our first model is the Speaker Model, which models the respondent alone. This model represents each individual speaker as a vector or embedding, which encodes speaker-specific information (e.g., dialect, register, age, gender, personal information) that influences the content and style of her responses. Note that these attributes are not explicitly annotated, which would be tremendously expensive for our datasets. Instead, our model manages to cluster users along some of these traits (e.g., age, country of residence) based on the responses alone.

Figure 1 gives a brief illustration of the Speaker Model. Each speaker $i \in [1, N]$ is associated with a user-level representation $v_i \in \mathbb{R}^{K \times 1}$. As in standard SEQ2SEQ models, we first encode message S into a vector representation h_S using the source LSTM. Then for each step in the target side, hidden units are obtained by combining the representation produced by the target LSTM at the previous time step, the word representations at the current time step, and the speaker embedding v_i:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \\ v_i \end{bmatrix} \quad (4)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \quad (5)$$

$$h_t^s = o_t \cdot \tanh(c_t) \quad (6)$$

where $W \in \mathbb{R}^{4K \times 3K}$. In this way, speaker information is encoded and injected into the hidden layer at each time step and thus helps predict personalized responses throughout the generation process. The Speaker embedding $\{v_i\}$ is shared across all conversations that involve speaker i. $\{v_i\}$ are learned by back propagating word prediction errors to each neural component during training.

Another useful property of this model is that it helps *infer* answers to questions even if the evidence is not readily present in the training set. This is important as the training data does not contain explicit information about every attribute of each user (e.g., gender, age, country of residence). The model learns speaker representations based on conversational content produced by different speakers, and speakers producing similar responses tend to have similar embeddings, occupying nearby positions in the vector space. This way, the training data of speakers nearby in vector space help increase the generalization capability of the speaker model. For example, consider two speakers i and j who sound distinctly British, and who are therefore close in speaker embedding space. Now, suppose that, in the training data, speaker i was asked *Where do you live?* and responded *in the UK*. Even if speaker j was never asked the same question, this answer can help influence a good response from speaker j, and this without explicitly labeled geo-location information.

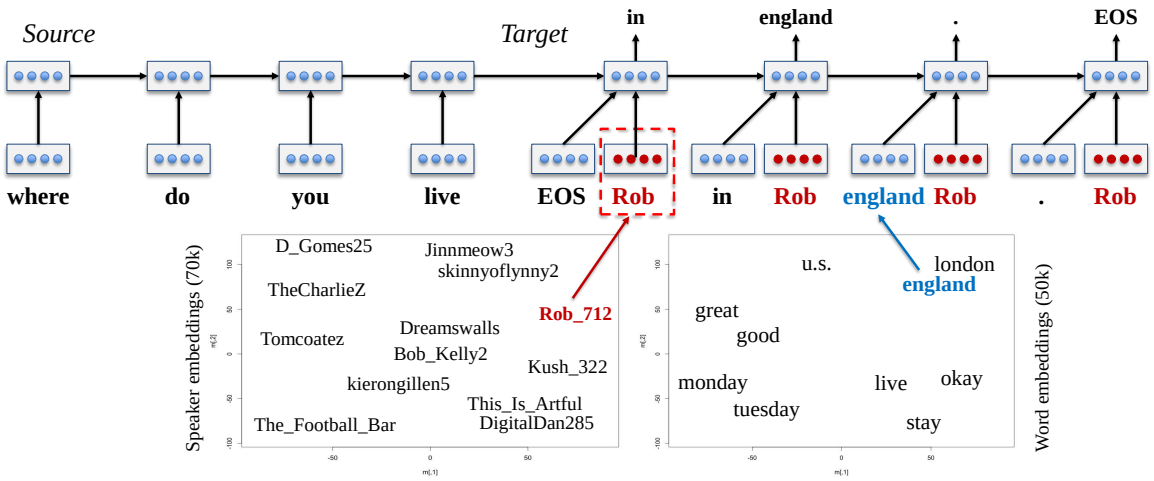


Figure 1: Illustrative example of the Speaker Model introduced in this work. Speaker IDs close in embedding space tend to respond in the same manner. These speaker embeddings are learned jointly with word embeddings and all other parameters of the neural model via backpropagation. In this example, say Rob is a speaker clustered with people who often mention England in the training data, then the generation of the token 'england' at time $t = 2$ would be much more likely than that of 'u.s.'. A non-persona model would prefer generating *in the u.s.* if 'u.s.' is more represented in the training data across all speakers.

4.3 Speaker-Addressee Model

A natural extension of the Speaker Model is a model that is sensitive to speaker-addressee interaction patterns within the conversation. Indeed, speaking style, register, and content does not vary only with the identity of the speaker, but also with that of the addressee. For example, in scripts for the TV series *Friends* used in some of our experiments, the character Ross often talks differently to his sister Monica than to Rachel, with whom he is engaged in an on-again off-again relationship throughout the series.

The proposed Speaker-Addressee Model operates as follows: We wish to predict how speaker i would respond to a message produced by speaker j. Similarly to the Speaker model, we associate each speaker with a K dimensional speaker-level representation, namely v_i for user i and v_j for user j. We obtain an interactive representation $V_{i,j} \in \mathbb{R}^{K\times 1}$ by linearly combining user vectors v_i and v_j in an attempt to model the interactive style of user i towards user j,

$$V_{i,j} = \tanh(W_1 \cdot v_i + W_2 \cdot v_2) \tag{7}$$

where $W_1, W_2 \in \mathbb{R}^{K\times K}$. $V_{i,j}$ is then linearly incorporated into LSTM models at each step in the target:

$$\begin{bmatrix} i_t \\ f_t \\ o_t \\ l_t \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} W \cdot \begin{bmatrix} h_{t-1} \\ e_t^s \\ V_{i,j} \end{bmatrix} \tag{8}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \tag{9}$$

$$h_t^s = o_t \cdot \tanh(c_t) \tag{10}$$

$V_{i,j}$ depends on both speaker and addressee and the same speaker will thus respond differently to a message from different interlocutors. One potential issue with Speaker-Addressee modelling is the difficulty involved in collecting a large-scale training dataset in which each speaker is involved in conversation with a wide variety of people. Like the Speaker Model, however, the Speaker-Addressee Model derives generalization capabilities from speaker embeddings. Even if the two speakers at test time (i and j) were never involved in the same conversation in the training data, two speakers i' and j' who are respectively close in embeddings may have been, and this can help modelling how i should respond to j.

4.4 Decoding and Reranking

For decoding, the N-best lists are generated using the decoder with beam size $B = 200$. We set a maximum length of 20 for the generated candidates. Decoding operates as follows: At each time step, we first examine all $B \times B$ possible next-word candidates, and add all hypothesis ending with an *EOS* token to the N-best list. We then preserve the top-B unfinished hypotheses and move to the next word position.

To deal with the issue that SEQ2SEQ models tend to generate generic and commonplace responses such as *I don't know*, we follow Li et al. (2016) by reranking the generated N-best list using

a scoring function that linearly combines a length penalty and the log likelihood of the source given the target:

$$\log p(R|M, v) + \lambda \log p(M|R) + \gamma |R| \quad (11)$$

where $p(R|M, v)$ denotes the probability of the generated response given the message M and the respondent's speaker ID. $|R|$ denotes the length of the target and γ denotes the associated penalty weight. We optimize γ and λ on N-best lists of response candidates generated from the development set using MERT (Och, 2003) by optimizing BLEU. To compute $p(M|R)$, we train an inverse SEQ2SEQ model by swapping messages and responses. We trained standard SEQ2SEQ models for $p(M|R)$ with no speaker information considered.

5 Datasets

5.1 Twitter Persona Dataset

Data Collection Training data for the Speaker Model was extracted from the Twitter FireHose for the six-month period beginning January 1, 2012. We limited the sequences to those where the responders had engaged in at least 60 (and at most 300) 3-turn conversational interactions during the period, in other words, users who reasonably frequently engaged in conversation. This yielded a set of 74,003 users who took part in a minimum of 60 and a maximum of 164 conversational turns (average: 92.24, median: 90). The dataset extracted using responses by these "conversationalists" contained 24,725,711 3-turn sliding-window (context-message-response) conversational sequences.

In addition, we sampled 12000 3-turn conversations from the same user set from the Twitter FireHose for the three-month period beginning July 1, 2012, and set these aside as development, validation, and test sets (4000 conversational sequences each). Note that development, validation, and test sets for this data are single-reference, which is by design. Multiple reference responses would typically require acquiring responses from different people, which would confound different personas.

Training Protocols We trained four-layer SEQ2SEQ models on the Twitter corpus following the approach of (Sutskever et al., 2014). Details are as follows:

- 4 layer LSTM models with 1,000 hidden cells for each layer.
- Batch size is set to 128.
- Learning rate is set to 1.0.
- Parameters are initialized by sampling from the uniform distribution $[-0.1, 0.1]$.
- Gradients are clipped to avoid gradient explosion with a threshold of 5.
- Vocabulary size is limited to 50,000.
- Dropout rate is set to 0.2.

Source and target LSTMs use different sets of parameters. We ran 14 epochs, and training took roughly a month to finish on a Tesla K40 GPU machine.

As only speaker IDs of responses were specified when compiling the Twitter dataset, experiments on this dataset were limited to the Speaker Model.

5.2 Twitter Sordoni Dataset

The Twitter Persona Dataset was collected for this paper for experiments with speaker ID information. To obtain a point of comparison with prior state-of-the-art work (Sordoni et al., 2015; Li et al., 2016), we measure our baseline (non-persona) LSTM model against prior work on the dataset of (Sordoni et al., 2015), which we call the Twitter Sordoni Dataset. We only use its test-set portion, which contains responses for 2114 context and messages. It is important to note that the Sordoni dataset offers up to 10 references per message, while the Twitter Persona dataset has only 1 reference per message. Thus BLEU scores cannot be compared across the two Twitter datasets (BLEU scores on 10 references are generally much higher than with 1 reference). Details of this dataset are in (Sordoni et al., 2015).

5.3 Television Series Transcripts

Data Collection For the dyadic Speaker-Addressee Model we used scripts from the American television comedies *Friends*[2] and *The Big Bang Theory*,[3] available from Internet Movie Script Database (IMSDb).[4] We collected 13 main characters from the two series in a corpus of 69,565 turns. We split the corpus into training/development/testing sets, with development and testing sets each of about 2,000 turns.

Training Since the relatively small size of the dataset does not allow for training an open domain dialog model, we adopted a domain adaption strategy where we first trained a standard SEQ2SEQ

[2] https://en.wikipedia.org/wiki/Friends

[3] https://en.wikipedia.org/wiki/The_Big_Bang_Theory

[4] http://www.imsdb.com

System	BLEU
MT baseline (Ritter et al., 2011)	3.60%
Standard LSTM MMI (Li et al., 2016)	5.26%
Standard LSTM MMI (our system)	5.82%
Human	*6.08%*

Table 2: BLEU on the Twitter Sordoni dataset (10 references). We contrast our baseline against an SMT baseline (Ritter et al., 2011), and the best result (Li et al., 2016) on the established dataset of (Sordoni et al., 2015). The last result is for a human oracle, but it is not directly comparable as the oracle BLEU is computed in a leave-one-out fashion, having one less reference available. We nevertheless provide this result to give a sense that these BLEU scores of 5-6% are not unreasonable.

models using a much larger OpenSubtitles (OSDb) dataset (Tiedemann, 2009), and then adapting the pre-trained model to the TV series dataset.

The OSDb dataset is a large, noisy, open-domain dataset containing roughly 60M-70M scripted lines spoken by movie characters. This dataset does not specify which character speaks each subtitle line, which prevents us from inferring speaker turns. Following Vinyals et al. (2015), we make the simplifying assumption that each line of subtitle constitutes a full speaker turn.[5] We trained standard SEQ2SEQ models on OSDb dataset, following the protocols already described in Section 5.1. We run 10 iterations over the training set.

We initialize word embeddings and LSTM parameters in the Speaker Model and the Speaker-Addressee model using parameters learned from OpenSubtitles datasets. User embeddings are randomly initialized from $[-0.1, 0.1]$. We then ran 5 additional epochs until the perplexity on the development set stabilized.

6 Experiments

6.1 Evaluation

Following (Sordoni et al., 2015; Li et al., 2016) we used BLEU (Papineni et al., 2002) for parameter tuning and evaluation. BLEU has been shown to correlate well with human judgment on the response generation task, as demonstrated in (Galley et al., 2015). Besides BLEU scores, we also report perplexity as an indicator of model capability.

6.2 Baseline

Since our main experiments are with a new dataset (the Twitter Persona Dataset), we first show that our LSTM baseline is competitive with the state-of-the-art (Li et al., 2016) on an established dataset, the Twitter Sordoni Dataset (Sordoni et al., 2015). Our baseline is simply our implementation of the LSTM-MMI of (Li et al., 2016), so results should be relatively close to their reported results. Table 2 summarizes our results against prior work. We see that our system actually does better than (Li et al., 2016), and we attribute the improvement to a larger training corpus, the use of dropout during training, and possibly to the "conversationalist" nature of our corpus.

[5]This introduces a degree of noise as consecutive lines are not necessarily from the same scene or two different speakers.

Model	Standard LSTM	Speaker Model
Perplexity	47.2	42.2 (-10.6%)

Table 3: Perplexity for standard SEQ2SEQ and the Speaker model on the Twitter Persona development set.

Model	Objective	BLEU
Standard LSTM	MLE	0.92%
Speaker Model	MLE	1.12% (+21.7%)
Standard LSTM	MMI	1.41%
Speaker Model	MMI	1.66% (+11.7%)

Table 4: BLEU on the Twitter Persona dataset (1 reference), for the standard SEQ2SEQ model and the Speaker model using as objective either maximum likelihood (MLE) or maximum mutual information (MMI).

6.3 Results

We first report performance on the Twitter Persona dataset. Perplexity is reported in Table 3. We observe about a 10% decrease in perplexity for the Speaker model over the standard SEQ2SEQ model. In terms of BLEU scores (Table 4), a significant performance boost is observed for the Speaker model over the standard SEQ2SEQ model, yielding an increase of 21% in the maximum likelihood (MLE) setting and 11.7% for mutual information setting (MMI). In line with findings in (Li et al., 2016), we observe a consistent performance boost introduced by the MMI objective function over a standard SEQ2SEQ model based on the MLE objective function. It is worth noting that our persona models are more beneficial to the MLE models than to the MMI models. This result is intuitive as the persona models help make Standard LSTM MLE outputs more informative and less bland, and thus make the use of MMI less critical.

For the TV Series dataset, perplexity and BLEU scores are respectively reported in Table 5 and Table 6. As can be seen, the Speaker and Speaker-Addressee models respectively achieve perplexity values of 25.4 and 25.0 on the TV-series dataset,

Model	Standard LSTM	Speaker Model	Speaker-Addressee Model
Perplexity	27.3	25.4 (−7.0%)	25.0 (−8.4%)

Table 5: Perplexity for standard SEQ2SEQ and persona models on the TV series dataset.

Model	Standard LSTM	Speaker Model	Speaker-Addressee Model
MLE	1.60%	1.82% (+13.7%)	1.83% (+14.3%)
MMI	1.70%	1.90% (+10.6%)	1.88% (+10.9%)

Table 6: BLEU on the TV series dataset (1 reference), for the standard SEQ2SEQ and persona models.

7.0% and 8.4% percent lower than the correspondent standard SEQ2SEQ models. In terms of BLEU score, we observe a similar performance boost as on the Twitter dataset, in which the Speaker model and the Speaker-Addressee model outperform the standard SEQ2SEQ model by 13.7% and 10.6%. By comparing the Speaker-Addressee model against the Speaker model on the TV Series dataset, we do not observe a significant difference. We suspect that this is primarily due to the relatively small size of the dataset where the interactive patterns might not be fully captured. Smaller values of perplexity are observed for the Television Series dataset than the Twitter dataset, the perplexity of which is over 40, presumably due to the noisier nature of Twitter dialogues.

6.4 Qualitative Analysis

Diverse Responses by Different Speakers Table 7 represents responses generated by persona models in response to three different input questions. We randomly selected 10 speakers (without cherry-picking) from the original Twitter dataset. We collected their user level representations from a speaker look-up table and integrated them into the decoding models. The model tends to generate specific responses for different people in response to the factual questions.[6]

Table 8 shows responses generated from the *Speaker-Addressee Model* using the TV-series dataset. Interestingly, we regularly observe that this model is sensitive to the identity of the addressee, generating words specifically targeted at that addressee (e.g., her name). For example, the model produces *Of course, I love you, Emily* in response to an input from *Emily*. Also, the model generates *Of course I love you. (kisses him)*, where the pronoun "him" accurately identifies the gender of the addressee.

[6]There appears to be a population bias in the training set that favors British users.

message	Where is your hometown?
baseline	I was born in Canada.
user1	I'm from England.
user2	I'm from Manchester.
user3	I'm from Liverpool.
user4	England. You?
user5	My hometown.
user6	I'm from Texas.
user7	I'm from LA.
user8	England.
user9	I was born here.
user10	I was born in the us.
message	What company do you work for?
baseline	I work for a company.
user1	I don't have a job.
user2	I'm a manager.
user3	I'm working. #
user4	I work for a company.
user5	I'm working. #
user6	Customer service.
user7	I work in retail.
user8	I work in a cafe.
user9	I work at a gas station.
user10	I don't know #
message	What did you have for dinner?
baseline	I had fish and chips.
user1	I had spag bol.
user2	Chicken and chips.
user3	Chicken and rice.
user4	Fish and chips.
user5	I had spag bol.
user6	I had Mexican food.
user7	Salad...
user8	I had chicken and chips.
user9	I had spag bol.
user10	Pizza.

Table 7: Responses generated by the baseline (LSTM-MMI) and the Speaker Model for ten randomly selected users, without cherry picking. # indicates poor-quality responses produced by the system.

Human Evaluation We conducted a human evaluation of outputs from the Speaker Model, using a crowdsourcing service. Since we cannot expect crowdsourced human judges to know or attempt to learn the ground truth of Twitter users who are not well-known public figures, we designed our experiment to evaluate the consistency of outputs associated with the speaker IDs. To this end, we collected 24 pairs of questions for which we would expect

message	Do you love me?
baseline	Of course, I love you.
	addressee: Emily, speaker: Sheldon
response	Of course, I love you, Emily.
	addressee: Emily, speaker: Penny
response	Of course, I love you, Emily.
	addressee: Leonard, speaker: Penny
response	Of course, I love you, Leonard.
	addressee: Monica, speaker: Chandler
response	Of course I love you. I want to marry you.
	addressee: Rachel, speaker: Chandler
response	Of course I love you.
	addressee: Ross, speaker: Chandler
response	Of course I love you.
	addressee: Ross, speaker: Rachel
response	Of course I love you. (kisses him)
	addressee: Emily, speaker: Rachel
response	Of course I love you.

Table 8: Responses to *Do you love me?* from the *Speaker-Addressee* model on the TV-series dataset using different addressees and speakers.

responses to be consistent if the persona model is coherent. For example, responses to the questions *What country do you live in?* and *What city do you live in?* would be considered consistent if the answers were *England* and *London* respectively, but not if they were *UK* and *Chicago*. Similarly, the responses to *Are you vegan or vegetarian?* and *Do you eat beef?* are consistent if the answers generated are *vegan* and *absolutely not*, but not if they are *vegan* and *I love beef*. We collected 20 pairs of outputs for randomly-selected personas provided by the Speaker Model for each question pair (480 response pairs total). We also obtained the corresponding outputs from the baseline MMI-enhanced SEQ2SEQ system.

Since our purpose is to measure the gain in consistency over the baseline system, we presented the pairs of answers system-pairwise, i.e., 4 responses, 2 from each system, displayed on the screen, and asked judges to decide which of the two systems was more consistent. The position in which the system pairs were presented on the screen was randomized. The two systems were judged on 5-point zero-sum scale, assigning a score of 2 (-2) if one system was judged more (less) consistent than the other, and 1 (-1) if one was rated "somewhat" more (less) consistent. Ties were assigned a score of zero. Five judges rated each pair and their scores were averaged and remapped into 5 equal-width bins. After discarding ties, we found the persona model was judged either "more consistent" or "somewhat more consistent" in 56.7% of cases. If we ignore the "somewhat more consistent" judgments, the persona model wins in 6.1% of cases, compared with only 1.6% for the baseline model. It should be emphasized that the baseline model is a strong baseline, since it represents the consensus of all 70K Twitter users in the dataset[7].

User1	
message	Where is your hometown?
response	I'm from England.
message	Where are you from?
response	I'm from England.
message	In which city do you live now?
response	I live in London.
message	In which country do you live now?
response	I live in England.
message	What is your major?
response	Business. You?
message	What did you study in college?
response	I did business studies.
message	How old are you?
response	I'm 18.
message	What is your age?
response	I'm 18.
User2	
message	Where is your hometown?
response	I live in Indonesia.
message	Where are you from?
response	England, you?
message	In which city do you live now?
response	I live in Jakarta.
message	In which country do you live now?
response	I live in Indonesia.
message	What is your major?
response	Business, you?
message	What did you study in college?
response	Psychology, you?
message	How old are you?
response	I'm 18.
message	What is your age?
response	I'm 16.

Table 9: Examples of speaker consistency and inconsistency generated by the Speaker Model

Table 9 illustrates how consistency is an emergent property of two arbitrarily selected users. The model is capable of discovering the relations between different categories of location such as London and the UK, Jakarta and Indonesia. However, the model also makes inconsistent response decisions, generating different answers in the second example in response to questions asking about age or major. Our proposed persona models integrate user embeddings into the LSTM, and thus can be viewed as encapsulating a trade-off between a persona-specific generation model and a general conversational model.

[7] *I'm not pregnant* is an excellent consensus answer to the question *Are you pregnant?*, while *I'm pregnant* is consistent as a response only in the case of someone who also answers the question *Are you a guy or a girl?* with something in the vein of *I'm a girl*.

7 Conclusions

We have presented two persona-based response generation models for open-domain conversation generation. There are many other dimensions of speaker behavior, such as mood and emotion, that are beyond the scope of the current paper and must be left to future work.

Although the gains presented by our new models are not spectacular, the systems outperform our baseline SEQ2SEQ systems in terms of BLEU, perplexity, and human judgments of speaker consistency. We have demonstrated that by encoding personas in distributed representations, we are able to capture personal characteristics such as speaking style and background information. In the Speaker-Addressee model, moreover, the evidence suggests that there is benefit in capturing dyadic interactions.

Our ultimate goal is to be able to take the profile of an arbitrary individual whose identity is not known in advance, and generate conversations that accurately emulate that individual's persona in terms of linguistic response behavior and other salient characteristics. Such a capability will dramatically change the ways in which we interact with dialog agents of all kinds, opening up rich new possibilities for user interfaces. Given a sufficiently large training corpus in which a sufficiently rich variety of speakers is represented, this objective does not seem too far-fetched.

Acknowledgments

We with to thank Stephanie Lukin, Pushmeet Kohli, Chris Quirk, Alan Ritter, and Dan Jurafsky for helpful discussions.

References

David Ameixa, Luisa Coheur, Pedro Fialho, and Paulo Quaresma. 2014. Luke, I am your father: dealing with out-of-domain requests by using movies subtitles. In *Intelligent Virtual Agents*, pages 13–21. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of the International Conference on Learning Representations (ICLR)*.

Rafael E Banchs and Haizhou Li. 2012. IRIS: a chat-oriented dialogue system based on the vector space model. In *Proc. of the ACL 2012 System Demonstrations*, pages 37–42.

Yun-Nung Chen, Wei Yu Wang, and Alexander Rudnicky. 2013. An empirical investigation of sparse log-linear models for improved dialogue act classification. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8317–8321. IEEE.

Werner Deutsch and Thomas Pechmann. 1982. Social interaction and the development of definite descriptions. *Cognition*, 11:159–184.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. ΔBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proc. of ACL-IJCNLP*, pages 445–450, Beijing, China, July.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proc. of ACL*, pages 699–709, Baltimore, Maryland.

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

Alfred Kobsa. 1990. User modeling in dialog systems: Potentials and hazards. *AI & society*, 4(3):214–231.

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

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proc. of NAACL-HLT*.

Grace I Lin and Marilyn A Walker. 2011. All the world's a stage: Learning character models from film. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proc. of ACL*, pages 11–19, Beijing, China, July.

Lasguido Nio, Sakriani Sakti, Graham Neubig, Tomoki Toda, Mirna Adriani, and Satoshi Nakamura. 2014. Developing non-goal dialog system based on examples of drama television. In *Natural Interaction with Robots, Knowbots and Smartphones*, pages 355–361. Springer.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Alice H Oh and Alexander I Rudnicky. 2000. Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 27–32.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*, pages 311–318.

Roberto Pieraccini, David Suendermann, Krishna Dayanidhi, and Jackson Liscombe. 2009. Are we there yet? research in commercial spoken dialog systems. In *Text, Speech and Dialogue*, pages 3–13. Springer.

Adwait Ratnaparkhi. 2002. Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech & Language*, 16(3):435–455.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven response generation in social media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 583–593.

Jost Schatztnann, Matthew N Stuttle, Karl Weilhammer, and Steve Young. 2005. Effects of the user model on simulation-based learning of dialogue strategies. In *Automatic Speech Recognition and Understanding, 2005 IEEE Workshop on*, pages 220–225.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proc. of AAAI*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL-IJCNLP*, pages 1577–1586.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Meg Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proc. of NAACL-HLT*.

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

Jörg Tiedemann. 2009. News from OPUS – a collection of multilingual parallel corpora with tools and interfaces. In *Recent advances in natural language processing*, volume 5, pages 237–248.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proc. of ICML Deep Learning Workshop*.

Wolfgang Wahlster and Alfred Kobsa. 1989. *User models in dialog systems*. Springer.

Marilyn A Walker, Rashmi Prasad, and Amanda Stent. 2003. A trainable generator for recommendations in multimodal dialog. In *INTERSPEECH*.

Marilyn A Walker, Ricky Grant, Jennifer Sawyer, Grace I Lin, Noah Wardrip-Fruin, and Michael Buell. 2011. Perceived or not perceived: Film character models for expressive nlg. In *Interactive Storytelling*, pages 109–121. Springer.

Marilyn A Walker, Grace I Lin, and Jennifer Sawyer. 2012. An annotated corpus of film dialogue for learning and characterizing character style. In *LREC*, pages 1373–1378.

William Yang Wang, Ron Artstein, Anton Leuski, and David Traum. 2011. Improving spoken dialogue understanding using phonetic mixture models. In *FLAIRS Conference*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proc. of EMNLP*, pages 1711–1721, Lisbon, Portugal, September. Association for Computational Linguistics.

Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. 2015. Attention with intention for a neural network conversation model. *CoRR*, abs/1510.08565.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Discriminative Deep Random Walk for Network Classification

Juzheng Li, Jun Zhu, Bo Zhang
Dept. of Comp. Sci. & Tech., State Key Lab of Intell. Tech. & Sys.
Tsinghua University, Beijing, 100084, China
lijuzheng09@gmail.com; {dcszj,dcszb}@tsinghua.edu.cn

Abstract

Deep Random Walk (DeepWalk) can learn a latent space representation for describing the topological structure of a network. However, for relational network classification, DeepWalk can be suboptimal as it lacks a mechanism to optimize the objective of the target task. In this paper, we present Discriminative Deep Random Walk (DDRW), a novel method for relational network classification. By solving a joint optimization problem, DDRW can learn the latent space representations that well capture the topological structure and meanwhile are discriminative for the network classification task. Our experimental results on several real social networks demonstrate that DDRW significantly outperforms DeepWalk on multi-label network classification tasks, while retaining the topological structure in the latent space. DDRW is stable and consistently outperforms the baseline methods by various percentages of labeled data. DDRW is also an online method that is scalable and can be naturally parallelized.

1 Introduction

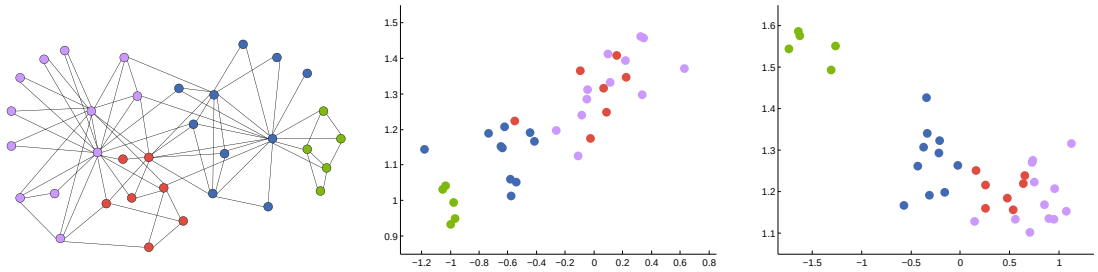
Categorization is an important task in natural language processing, especially with the growing scale of documents in the Internet. As the documents are often not isolated, a large amount of the linguistic materials present a network structure such as citation, hyperlink and social networks. The large size of networks calls for scalable machine learning methods to analyze such data. Recent efforts have been made in developing statistical models for various network analysis tasks, such as network classification (Neville and Jensen, 2000), content recommendation (Fouss et al., 2007), link prediction (Adamic and Adar, 2003), and anomaly detection (Savage et al., 2014). One common challenge of statistical network models is to deal with the sparsity of networks, which may prevent a model from generalizing well.

One effective strategy to deal with network sparsity is to learn a latent space representation for the entities in a network (Hoff et al., 2002; Zhu, 2012; Tang and Liu, 2011; Tang et al., 2015). Among various approaches, DeepWalk (Perozzi et al., 2014) is a recent method that embeds all the entities into a continuous vector space using deep learning methods. DeepWalk captures entity features like neighborhood similarity and represents them by Euclidean distances (See Figure 1(b)). Furthermore, since entities that have closer relationships are more likely to share the same hobbies or belong to the same groups, such an embedding by DeepWalk can be useful for network classification, where the topological information is explored to encourage a globally consistent labeling.

Although DeepWalk is effective on learning embeddings of the topological structure, when dealing with a network classification task, it lacks a mechanism to optimize the objective of the target task and thus often leads to suboptimal embeddings. In particular, for our focus of relational network classification, we would like the embeddings to be both representing the topological structure of the network actors and discriminative in predicting the class labels of actors.

To address the above issues, we present Discriminative Deep Random Walk (DDRW) for relational network classification. DDRW extends DeepWalk by jointly optimizing the classification objective and the objective of embedding entities in a latent space that maintains the topological structure. Under this joint learning framework, DDRM manages to learn the latent representations

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1004–1013,
Berlin, Germany, August 7-12, 2016.



(a) Karate Graph (b) DeepWalk Embedding (c) DDRW Embedding

Figure 1: Different experimental results of embedding a network into a two dimensional real space. We use Karate Graph (Macskassy and Provost, 1977) for this example. Four different colors stand for the classes of the vertices. In (b), vertices which have stronger relations in the network are more likely to be closer in the embedding latent space. While in (c), besides the above-mentioned property, DDRW makes vertices in different classes more separated.

that are strongly associated with the class labels (See Figure 1(c)), making it easy to find a separating boundary between the classes, and the actors that are connected in the original network are still close to each other in the latent social space. This idea of combining task-specific and representation objectives has been widely explored in other regions such as MedLDA (Zhu et al., 2012) and Supervised Dictionary Learning (Mairal et al., 2009).

Technically, to capture the topological structure, we follow the similar idea of DeepWalk by running truncated random walks on the original network to extract sequences of actors, and then building a language model (i.e., Word2Vec (Mikolov et al., 2013b)) to project the actors into a latent space. To incorporate the supervising signal in network classification, we build a classifier based on the latent space representations. By sharing the same latent social space, the two objectives are strongly coupled and the latent social space is guided by both the network topology and class labels. DDRW optimizes the joint objective by using stochastic gradient descent, which is scalable and embarrassingly parallizable.

We evaluate the performance on several real-world social networks, including BlogCatalog, Flickr and YouTube. Our results demonstrate that DDRW significantly boosts the classification accuracy of DeepWalk in multi-label network classification tasks, while still retaining the topological structure in the learnt latent social space. We also show that DDRW is stable and consistently outperforms the baseline methods by various percentages of labeled data. Although the networks we use only bring topological information for clarity, DDRW is flexible to consider additional attributes (if any) of vertices. For example, DDRW can be naturally extended to classify documents/webpages, which are often represented as a network (e.g., citation/hyperlink network), by conjoining with a word2vec component to embed the documents/webpages into the same latent space, similar as previous work on extending DeepWalk to incorporate attributes (Yang et al., 2015).

2 Problem Definition

We consider the network classification problem, which classifies entities from a given network into one or more categories from a set $\mathcal{Y}$. Let $G = (V, E, Y)$ denote a network, where V is the set of vertices, representing the entities of the network; $E \subseteq (V \times V)$ is the set of edges, representing the relations between the entities; and $Y \subseteq \mathbb{R}^{|V|\times|\mathcal{Y}|}$ denotes the labels of entities. We also consider Y_U as a set of unknown labels in the same graph G. The target of the classification task is to learn a model from labeled data and generate a label set Y_P to be the prediction of Y_U. The difference between Y_P and Y_U indicates the classification quality.

When classifying elements $X \in \mathbb{R}^n$, traditional machine learning methods learn a weight matrix H to minimize the difference between $Y_P = \mathcal{F}(X, H)$ and Y_U, where $\mathcal{F}$ is any known fixed function. In network aspect, we will be able to utilize well-developed machine learning methods if adequate information of G is embedded into a corresponding form as X. By this motivation, relational learning (Getoor and Taskar, 2007; Neville and Jensen, 2000) methods are pop-

ularly employed. In network classification, the internal structure of a network is resolved to extract the neighboring features of the entities (Macskassy and Provost, 2007; Wang and Sukthankar, 2013). Accordingly, the core problem is how to describe the irregular networks within formal feature spaces. A variety of approaches have been proposed with the purpose of finding effective statistical information through the network (Gallagher and Eliassi-Rad, 2008; Henderson et al., 2011; Tang and Liu, 2011).

DeepWalk (Perozzi et al., 2014) is an outstanding method for network embedding, which uses truncated random walks to capture the explicit structure of the network and applies language models to learn the latent relationships between the actors. When applied to the network classification task, DeepWalk first learns X which describes the topological structure of G and then learns a subsequent classifier H. One obvious shortcoming of this two-step procedure is that the embedding step is unaware of the target class label information and likely to learn embeddings that are suboptimal for classification.

We present Discriminative Deep Random Walk (DDRW) to enhance the effect of DeepWalk by learning $X \in \mathbb{R}^{|V|\times d}$ and $H \in \mathbb{R}^{d\times|\mathcal{Y}|}$ jointly. By using topological and label information of a certain network simultaneously, we will show that DDRW improves the classification accuracy significantly compared with most recent related methods. Furthermore, we will also show that the embedded result X produced by DDRW is able to retain the structure of G well.

3 Discriminative Deep Random Walk

In this section, we present the details of Discriminative Deep Random Walk (DDRW). DDRW has both embedding and classification objectives. We optimize the two objectives jointly to learn latent representations that are strongly associated with the class labels in the latent space. We use stochastic gradient descent (Mikolov et al., 1991) as our optimization method.

3.1 Embedding Objective

Let $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{|V|})$ denote the embedded vectors in the latent space, and $\boldsymbol{\alpha}$ denote the topological structure of the graph. The embedding objective can be described as an optimization problem as follows:

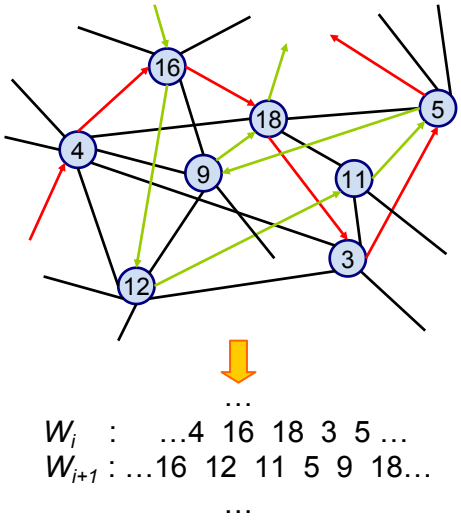


Figure 2: A part of Random Walk process in an undirected graph. Every time an adjacent vertex is chosen randomly (no matter visited or not) as the arrows indicate, until reaching the maximum length s.

$$\min_{\boldsymbol{\theta}} \ \mathcal{L}_r(\boldsymbol{\theta}, \boldsymbol{\alpha}), \tag{1}$$

where $\mathcal{L}_r$ indicates the difference between the embedded representations $\boldsymbol{\theta}$ and original topological structure $\boldsymbol{\alpha}$. For this objective, we use truncated random walks to capture the topological structure of the graph and the language model Word2Vec (Mikolov et al., 2013b) to learn the latent representations. Below, we explain each in turn.

3.1.1 Random Walk

Random Walk has been used in different regions in network analysis to capture the topological structure of graphs (Fouss et al., 2007; Andersen et al., 2006). As the name suggests, Random Walk chooses a certain vertex in the graph for the first step and then randomly migrates through the edges. Truncated random walk defines a maximum length s for all walk streams.

In our implementation, we shuffle the whole vertices V in the graph for τ times to build the sample set W. After each time of shuffling, we take the permutation list of vertices as the starting points of walks. Every time a walk stream starts at one element in order, randomly chooses an adjacent vertex to move, and ends when this stream reaches s vertices. By this procedure we get totally

$\tau|V|$ samples (i.e. walk streams) from the graph. Thus our sample set $W \in \mathbb{R}^{\tau|V| \times s}$ is obtained as the training materials.

3.1.2 Word2Vec

Existing work has shown that both the vertices in truncated random walks and the words in text articles follow similar power-law distributions in frequency, and then the idea of reshaping a social network into a form of corpus is very straightforward (Perozzi et al., 2014). Corresponding to linguistic analysis region, the objective is to find an embedding for a corpus to show the latent significances between the words. Words which have closer meanings are more likely to be embedded into near positions. Word2Vec (Mikolov et al., 2013b) is an appropriate tool for this problem. We use the Skip-gram (Mikolov et al., 2013a) strategy in Word2Vec, which uses the central word in a sliding window with radius R to predict other words in the window and make local optimizations. Specifically, let $\boldsymbol{\omega} = rw(\boldsymbol{\alpha})$ denote the full walk streams obtained from truncated random walks in Section 3.1.1. Then by Skip-gram we can get the objective function

$$\mathcal{L}_r(\boldsymbol{\theta}, \boldsymbol{\alpha}) = -\sum_{i=1}^{\tau} \frac{1}{s} \sum_{t=1}^{s} \sum_{-R \le j \le R, j \neq 0} \log p(\boldsymbol{\omega}_{i,t+j} | \boldsymbol{\omega}_{i,j}). \tag{2}$$

The standard Skip-gram method defines $p(\boldsymbol{\omega}_{i,t+j}|\boldsymbol{\omega}_{i,j})$ in Eq.(2) as follows:

$$p(\boldsymbol{\omega}_O|\boldsymbol{\omega}_I) = \frac{\exp(\boldsymbol{\theta}_{\boldsymbol{\omega}_O}^T \hat{\boldsymbol{\theta}}_{\boldsymbol{\omega}_I})}{\sum_{i=1}^{|V|} \exp(\boldsymbol{\theta}_i^T \hat{\boldsymbol{\theta}}_{\boldsymbol{\omega}_I})}, \tag{3}$$

where $\hat{\boldsymbol{\theta}}_i$ and $\boldsymbol{\theta}_i$ are the input and output representations of the ith vertex, respectively.

One shortcoming of the standard form is that the summation in Eq.(3) is very inefficient. To reduce the time consumption, we use the Hierarchical Softmax (Mnih and Hinton, 2009; Morin and Bengio, 2005) which is included in Word2Vec packages*. In Hierarchical Softmax, the Huffman binary tree is employed as an alternative representation for the vocabulary. The gradient descent step will be faster thanks to the Huffman tree structure which allows a reduction of output units necessarily evaluated.

*https://code.google.com/archive/p/word2vec/

3.2 Classification Objective

Let $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_{|V|})$ denote the labels, and $\boldsymbol{\beta}$ denote the subsequent classifier. The classification objective can be described as an optimization problem:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\beta}} \ \mathcal{L}_c(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{y}). \tag{4}$$

In DDRW, we use existing classifiers and do not attempt to extend them. Although SVM^{multicalss} (Crammer and Singer, 2002) often shows good performance in multi-class tasks empirically, we choose the classifier being referred to as L2-regularized and L2-loss Support Vector Classification (Fan et al., 2008) to keep pace with the baseline methods to be mentioned in Section 4.

In L2-regularized and L2-loss SVC, the loss function is

$$\begin{aligned} &\mathcal{L}_c(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{y}) \\ =&C \sum_{i=1}^{|V|} (\sigma(1 - \boldsymbol{y}_i \boldsymbol{\beta}^T \boldsymbol{\theta}_i))^2 + \frac{1}{2} \boldsymbol{\beta}^T \boldsymbol{\beta}, \end{aligned} \tag{5}$$

where C is the regularization parameter, $\sigma(x) = x$ if $x > 0$ and $\sigma(x) = 0$ otherwise. Eq.(5) is for binary classification problems, and is extended to multi-class problems following the *one-against-rest* strategy (Fan et al., 2008).

3.3 Joint Learning

The main target of our method is to classify the unlabeled vertices in the given network. We achieve this target with the help of intermediate embeddings which latently represent the network structure. We simultaneously optimize two objectives in Section 3.1 and 3.2. Specifically, let $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{y}) = \eta \mathcal{L}_r(\boldsymbol{\theta}, \boldsymbol{\alpha}) + \mathcal{L}_c(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{y})$, where η is a key parameter that balances the weights of the two objectives. We solve the joint optimization problem:

$$\min_{\boldsymbol{\theta}, \boldsymbol{\beta}} \ \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{y}). \tag{6}$$

We use stochastic gradient descent (Mikolov et al., 1991) to solve the optimization problem in Eq.(6). In each gradient descent step, we have

$$\begin{aligned} \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \delta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \boldsymbol{\theta} - \delta(\eta \frac{\partial \mathcal{L}_r}{\partial \boldsymbol{\theta}} + \frac{\partial \mathcal{L}_c}{\partial \boldsymbol{\theta}}), \\ \boldsymbol{\beta} &\leftarrow \boldsymbol{\beta} - \delta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \delta \frac{\partial \mathcal{L}_c}{\partial \boldsymbol{\beta}}, \end{aligned} \tag{7}$$

where δ is the learning rate for stochastic gradient descent. In our implementation, δ is initially set to

0.025 and linearly decreased with the steps, same as the default setting of Word2Vec. The derivatives in Eq.(7) are estimated by local slopes.

In Eq.(7), the latent representations adjust themselves according to both topological information ($\partial\mathcal{L}_r/\partial\boldsymbol{\theta}$) and label information ($\partial\mathcal{L}_c/\partial\boldsymbol{\theta}$). This process intuitively makes vertices in the same class closer and those in different classes farther, and this is also proved by experiments (See Figure 1). Thus by joint learning, DDRW can learn the latent space representations that well capture the topological structure and meanwhile are discriminative for the network classification task.

We take each sample W_i from walk streams W to estimate the local derivatives of the loss function for a descent step. Stochastic gradient descent enables DDRW to be an online algorithm, and thus our method is easy to be parallelized. Besides, a vertex may repeatedly appear for numerous times in W produced by random walks. This repeat is superfluous for classifiers and there is a considerable possibility to arise overfitting. Inspired from DropOut (Hinton et al., 2012) ideas, we randomly ignore the label information to control the optimization process in an equilibrium state.

4 Experimental Setup

In this section we present an overview of the datasets and baseline methods which we will compare with in the experiments.

4.1 Datasets

We use three popular social networks, which are exactly same with those used in some of the baseline methods. Table 1 summarizes the statistics of the data.

- **BlogCatalog**: a network of social relationships provided by blog authors. The labels of this graph are the topics specified by the uploading users.
- **Flickr**: a network of the contacts between users of the Flickr photo sharing website. The labels of this graph represent the interests of users towards certain categories of photos.
- **YouTube**: a network between users of the Youtube video sharing website. The labels stand for the groups of the users interested in different types of videos.

Dataset	BlogCatalog	Flickr	YouTube
Actors $\|V\|$	10,312	80,513	1,138,499
Links $\|E\|$	333,983	5,899,882	2,990,443
Labels $\|\mathcal{Y}\|$	29	195	47
Sparsity	6.3×10^{-3}	1.8×10^{-3}	4.6×10^{-6}
Max Degree	3,992	5,706	28,754
Average Degree	65	146	5

Table 1: Statistics of the three networks. Sparsity indicates the ratio of the actual links and links in a complete graph.

4.2 Baseline Methods

We evaluate our proposed method by comparing it with some significantly related methods.

- **LINE** (Tang et al., 2015)[†]: This method takes the edges of a graph as samples to train the *first-order* and *second-order* proximity seprately and integrate the results as an embedding of the graph. This method can handle both graphs with unweighted and weighted and is especially efficient in large networks.
- **DeepWalk** (Perozzi et al., 2014): This method employs language models to learn latent relations between the vertices in the graph. The basic assumption is that the closer two vertices are in the embedding space, the deeper relationships they have and there is higher possibility that they are in the same categories.
- **SpectralClustering** (Tang and Liu, 2011): This method finds out that graph cuts are useful for the classification task. This idea is implemented by finding the eigenvectors of a normalized graph Laplacian of the original graph.
- **EdgeCluster** (Tang and Liu, 2009b): This method uses k-means clustering algorithm to segment the edges of the graph into pieces. Then it runs iterations on the small clusters to find the internal relationships separately. The core idea is to scale time-consuming work into tractable sizes.
- **Majority**: This baseline method simply chooses the most frequent labels. It does not use any structural information of the graph.

[†]Although LINE also uses networks from Flickr and YouTube in its experiments, the networks are different from this paper.

As the datasets are not only multi-class but also multi-label, we usually need a thresholding method to test the results. But literature gives a negative opinion of arbitrarily choosing thresholding methods because of the considerably different performances. To avoid this, we assume that the number of the labels is already known in all the test processes.

5 Experiments

In this section, we present the experimental results and analysis on both network classification and latent space learning. We thoroughly evaluate the performance on the three networks and analyze the sensitivity to key parameters.

5.1 Classification Task

We first represent the results on multi-class classification and compare with the baseline methods. To have a direct and fair comparison, we use the same data sets, experiment procedures and testing points as in the reports of our relevant baselines (Perozzi et al., 2014; Tang and Liu, 2011; Tang and Liu, 2009b). The training set of a specified graph consists of the vertices, the edges and the labels of a certain percentage of labeled vertices. The testing set consists of the rest of the labels. We employ Macro-F_1 and Micro-F_1 (Yang, 1999) as our measurements. Micro-F_1 computes F_1 score globally while Macro-F_1 caculates F_1 score locally and then average them globally. All the results reported are averaged from 10 repeated processes.

5.1.1 BlogCatalog

BlogCatalog is the smallest dataset among the three. In BlogCatalog we vary the percentage of labeled data from 10% to 90%. Our results are presented in Table 2. We can see that DDRW performs consistently better than all the baselines on both Macro-F_1 and Micro-F_1 with the increasing percentage of labeled data. When compared with DeepWalk, DDRW obtains larger improvement when the percentage of labeled nodes is high. This improvement demonstrates the significance of DDRW on learning discriminative latent embeddings that are good for classification tasks.

5.1.2 Flickr

Flickr is a larger dataset with quite a number of classes. In this experiment we vary the percentage of labeled data from 1% to 10%. Our results are presented in Table 3. We can see that DDRW still performs better than the baselines significantly on both Macro-F_1 and Micro-F_1, and the results are consistent with what in BlogCatalog.

5.1.3 YouTube

YouTube is an even larger dataset with fewer classes than Flickr. In YouTube we vary the percentage of labeled data from 1% to 10%. Our results are presented in Table 4. In YouTube, LINE shows its strength in large sparse networks, probably because the larger scale of samples reduces the discrepancy from actual distributions. But from a general view, DDRW still performs better at most of the test points thanks to the latent representations when links are not sufficient.

5.2 Parameter Sensitivity

We now present an analysis of the sensitivity with respect to several important parameters. We measure our method with changing parameters to evaluate its stability. Despite the parameters which are unilateral to classification performance, the two main bidirectional parameters are η and the dimension d of embedding space in different percentages of labeled data. We use BlogCatalog and Flickr networks for the experiments, and fix parameters of random walks ($\tau = 30, s = 40, R = 10$). We do not represent the effects of changing parameters of random walks because results usually show unilateral relationships with them.

5.2.1 Effect of η

The key parameter η in our algorithm adjusts the weights of two objectives (Section 3.3). We represent the effect of changing η in Figure 3(a) and 3(b). We fix $d = 128$ in these experiments. Although rapid gliding can be observed on either sides, there are still sufficient value range where DDRW keeps the good performance. These experiments also show that η is not very sensitive towards the percentage of labeled data.

5.2.2 Effect of Dimensionality

We represent the effect of changing dimension d of the embedding space in Figure 3(c) and 3(d). We fix $\eta = 1.0$ in these experiments. There is decline when the dimension is high, but this decrease is not very sharp. Besides, when the dimension is high, the percentage of labeled data has more effect on the performance.

	Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-$F_1(\%)$	DDRW	**37.13**	**39.31**	**41.08**	**41.76**	**42.64**	**43.17**	**43.80**	**44.11**	**44.79**
	LINE	35.42	37.89	39.71	40.62	41.46	42.09	42.55	43.26	43.68
	DeepWalk	36.00	38.20	39.60	40.30	41.00	41.30	41.50	41.50	42.00
	SpecClust	31.06	34.95	37.27	38.93	39.97	40.99	41.66	42.42	42.62
	EdgeClust	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-$F_1(\%)$	DDRW	**21.69**	**24.33**	**26.28**	**27.78**	**28.76**	**29.53**	**30.47**	**31.40**	**32.04**
	LINE	20.98	23.44	24.91	26.06	27.19	27.89	28.43	29.10	29.45
	DeepWalk	21.30	23.80	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpecClust	19.14	23.57	25.97	27.46	28.31	29.46	30.13	31.38	31.78
	EdgeClust	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

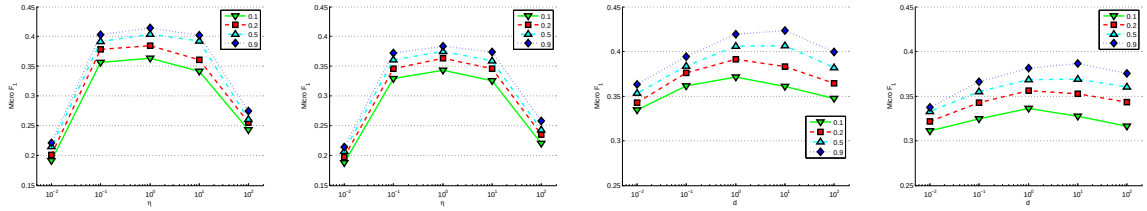
Table 2: Multi-class classification results in BlogCatalog.

	Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-$F_1(\%)$	DDRW	**33.61**	**35.20**	**36.72**	**37.43**	**38.31**	**38.89**	**39.33**	**39.64**	**39.85**	**40.02**
	LINE	31.65	33.98	35.46	36.63	37.53	38.20	38.47	38.74	39.07	39.25
	DeepWalk	32.40	34.60	35.90	36.70	37.20	37.70	38.10	38.30	38.50	38.70
	SpecClust	27.43	30.11	31.63	32.69	33.31	33.95	34.46	34.81	35.14	35.41
	EdgeClust	25.75	28.53	29.14	30.31	30.85	31.53	31.75	31.76	32.19	32.84
	Majority	16.34	16.31	16.34	16.46	16.65	16.44	16.38	16.62	16.67	16.71
Macro-$F_1(\%)$	DDRW	**14.49**	**17.81**	**20.05**	**21.40**	**22.91**	**23.84**	**25.12**	**25.79**	**26.28**	**26.43**
	LINE	13.69	17.77	19.88	21.07	22.36	23.62	24.78	25.11	25.69	25.90
	DeepWalk	14.00	17.30	19.60	21.10	22.10	22.90	23.60	24.10	24.60	25.00
	SpecClust	13.84	17.49	19.44	20.75	21.60	22.36	23.01	23.36	23.82	24.05
	EdgeClust	10.52	14.10	15.91	16.72	18.01	18.54	19.54	20.18	20.78	20.85
	Majority	0.45	0.44	0.45	0.46	0.47	0.44	0.45	0.47	0.47	0.47

Table 3: Multi-class classification results in Flickr.

	Labeled Nodes	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Micro-$F_1(\%)$	DDRW	**38.18**	**39.46**	40.17	41.09	**41.76**	**42.31**	**42.80**	**43.29**	**43.81**	**44.12**
	LINE	38.06	39.36	**40.30**	**41.14**	41.58	41.93	42.22	42.67	43.09	43.55
	DeepWalk	37.95	39.28	40.08	40.78	41.32	41.72	42.12	42.48	42.78	43.05
	SpecClust	26.61	35.16	37.28	38.35	38.90	39.51	40.02	40.49	40.86	41.13
	EdgeClust	23.90	31.68	35.53	36.76	37.81	38.63	38.94	39.46	39.92	40.07
	Majority	24.90	24.84	25.25	25.23	25.22	25.33	25.31	25.34	25.38	25.38
Macro-$F_1(\%)$	DDRW	**29.35**	**32.07**	**33.56**	**34.41**	**34.89**	**35.38**	**35.80**	**36.15**	**36.36**	**36.72**
	LINE	27.36	31.08	32.51	33.39	34.26	34.81	35.27	35.52	35.95	36.14
	DeepWalk	29.22	31.83	33.06	33.90	34.35	34.66	34.96	35.22	35.42	35.67
	SpecClust	24.62	29.33	31.30	32.48	33.24	33.89	34.15	34.47	34.77	34.98
	EdgeClust	19.48	25.01	28.15	29.17	29.82	30.65	30.75	31.23	31.45	31.54
	Majority	6.12	5.86	6.21	6.10	6.07	6.19	6.17	6.16	6.18	6.19

Table 4: Multi-class classification results in YouTube.



(a) BlogCatalog, η (b) Flickr, η (c) BlogCatalog, d (d) Flickr, d

Figure 3: Parameter Sensitivity in BlogCatalog and Flickr

K	1	5	10	20	50
DDRW(10%)	**91.3**	71.0	58.3	44.3	31.2
DDRW(50%)	90.9	69.8	**62.0**	44.7	30.7
DDRW(90%)	90.2	72.8	59.7	43.4	31.1
DeepWalk	91.2	**73.2**	59.8	**46.5**	**31.2**
Random	0.7	0.7	0.7	0.6	0.6

Table 5: Adjacency Predict Accuracy(%) in BlogCatalog.

5.3 Representation Efficiency

Finally, we examine the quality of the latent embeddings of entities discovered by DDRW. For network data, our major expectation is that the embedded social space should maintain the topological structure of the network. A visualization of the topological structure in a social space is showed in Figure 1. Besides, we examine the neighborhood structure of the vertices. Specifically, we check the top-K nearest vertices for each vertex in the embedded social space and calculate how many of the vertex pairs have edges between them in the observed network. We call this Adjacency Predict Accuracy. Table 5 shows the results, where DDRW with different percentages of labeled data, DeepWalk and Random are compared in BlogCatalog dataset. The baseline method Random maps all the vertices equably randomly into a fixed-size space. The experiments show that although DeepWalk outperforms on the whole, the performance of DDRW is approximate. DDRW is proved to inherit some important properties in latent representations of the network.

6 Related Work

Relational classification (Geman and Geman, 1984; Neville and Jensen, 2000; Getoor and Taskar, 2007) is a class of methods which involve the data item relation links during classification. A number of researchers have studied different methods for network relational learning. (Macskassy and Provost, 2003) present a simple weighted vote relational neighborhood classifier. (Xu et al., 2008) leverage the nonparametric infinite hidden relational model to analyze social networks. (Neville and Jensen, 2005) propose a latent group model for relational data, which discovers and exploits the hidden structures responsible for the observed autocorrelation among class labels. (Tang and Liu, 2009a) propose the latent social dimensions which are represented as continuous values and allow each node to involve at different dimensions in a flexible manner. (Gallagher et al., 2008) propose a method that learn sparsely labeled network data by adding ghost edges between neighbor vertices, and (Lin and Cohen, 2010) by using PageRank. (Wang and Sukthankar, 2013) extend the conventional relational classification to consider more additional features. (Gallagher and Eliassi-Rad, 2008) propose a complimentary approach to within-network classification based on the use of label-independent features. (Henderson et al., 2011) propose a regional feature generating method and demonstrate the usage of the regional feature in within-network and across-network classification. (Tang and Liu, 2009b) propose an edge-centric clustering scheme to extract sparse social dimensions for collective behavior prediction. (Tang and Liu, 2011) propose the concept of social dimensions to represent the latent affiliations of the entities. (Vishwanathan et al., 2010) propose Graph Kernels to use relational data during classification process and (Kang et al., 2012) propose a faster approximated method of Graph Kernels.

7 Conclusion

This paper presents Discriminative Deep Random Walk (DDRW), a novel approach for relational multi-class classification on social networks. By simultaneously optimizing embedding and classification objectives, DDRW gains significantly better performances in network classification tasks

than baseline methods. Experiments on different real-world datasets represent adequate stability of DDRW. Furthermore, the representations produced by DDRW is both an intermediate variable and a by-product. Same as other embedding methods like DeepWalk, DDRW can provide well-formed inputs for statistical analyses other than classification tasks. DDRW is also naturally an online algorithm and thus easy to parallel.

The future work has two main directions. One is semi-supervised learning. The low proportion of labeled vertices is a good platform for semi-supervised learning. Although DDRW has already combined supervised and unsupervised learning together, better performance can be expected after introducing well-developed methods. The other direction is to promote the random walk step. Literature has represented the good combination of random walk and language models, but this combination may be unsatisfactory for classification. It would be great if a better form of random walk is found.

Acknowledgments

The work was supported by the National Basic Research Program (973 Program) of China (No. 2013CB329403), National NSF of China (Nos. 61322308, 61332007), the Youngth Top-notch Talent Support Program, Tsinghua TNList Lab Big Data Initiative, and Tsinghua Initiative Scientific Research Program (No. 20141080934).

References

Lada A. Adamic and Eytan Adar. 2003. Friends and neighbors on the web. *Social Networks*, 25:211–230.

Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. 2006. Local graph partitioning using pagerank vectors. In *Foundations of Computer Science*, pages 476–486.

Koby Crammer and Yoram Singer. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19:355–369.

Brian Gallagher and Tina Eliassi-Rad. 2008. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. In *Proceedings of the Second International Conference on Advances in Social Network Mining and Analysis*, pages 1–19.

Brian Gallagher, Hanghang Tong, Tina Eliassi-Rad, and Christos Faloutsos. 2008. Using ghost edges for classification in sparsely labeled networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 256–264.

Stuart Geman and Donald Geman. 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6:721–741.

Lise Getoor and Ben Taskar. 2007. *Introduction to statistical relational learning*. The MIT Press.

Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 663–671.

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

Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. 2002. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97:1090–1098.

U. Kang, Hanghang Tong, and Jimeng Sun. 2012. Fast random walk graph kernel. In *SDM*, pages 828–838.

Frank Lin and William W. Cohen. 2010. Semi-supervised classification of network data using very few labels. In *Proceedings of the 2010 International Conference on Advances in Social Networks Analysis and Mining*, pages 192–199.

Sofus A. Macskassy and Foster J. Provost. 1977. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473.

Sofus A. Macskassy and Foster Provost. 2003. A simple relational classifier. In *Proceedings of the Multi-Relational Data Mining Workshop at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sofus A. Macskassy and Foster J. Provost. 2007. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8:935–983.

Julien Mairal, Jean Ponce, Guillermo Sapiro, Andrew Zisserman, and Francis R. Bach. 2009. Supervised dictionary learning. In *Advances in Neural Information Processing Systems*, pages 1033–1040.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 1991. Stochastic gradient learning in neural networks. In *Proceedings of Neuro-Nîmes 91*.

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

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

Andriy Mnih and Geoffrey E. Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems*, pages 1081–1088.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 246–252.

Jennifer Neville and David Jensen. 2000. Iterative classification in relational data. In *Proceedings of AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pages 13–20.

Jennifer Neville and David Jensen. 2005. Leveraging relational autocorrelation with latent group models. In *Proceedings of the 4th International Workshop on Multi-relational Mining*, pages 49–55.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 701–710.

David Savage, Xiuzhen Zhang, Xinghuo Yu, Pauline Lienhua Chou, and Qingmai Wang. 2014. Anomaly detection in online social networks. *Social Networks*, 39:62–70.

Lei Tang and Huan Liu. 2009a. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 817–826.

Lei Tang and Huan Liu. 2009b. Scalable learning of collective behavior based on sparse social dimensions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1107–1116.

Lei Tang and Huan Liu. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery*, 23:447–478.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research*, 11:1201–1242.

Xi Wang and Gita Sukthankar. 2013. Multi-label relational neighbor classification using social context features. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 464–472.

Zhao Xu, Volker Tresp, Shipeng Yu, and Kai Yu. 2008. Nonparametric relational learning for social network analysis. In *the 2nd SNA-KDD Workshop on Social Network Mining and Analysis*.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y. Chang. 2015. Network representation learning with rich text information. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 2111–2117.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1:69–90.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2012. MedLDA: maximum margin supervised topic models. *The Journal of Machine Learning Research*, 13:2237–2278.

Jun Zhu. 2012. Max-margin nonparametric latent feature models for link prediction. In *Proceedings of the 29th International Conference on Machine Learning*, pages 719–726.

Normalising Medical Concepts in Social Media Texts by Learning Semantic Representation

Nut Limsopatham and **Nigel Collier**
Language Technology Lab
Department of Theoretical and Applied Linguistics
University of Cambridge
Cambridge, UK
{nl347,nhc30}@cam.ac.uk

Abstract

Automatically recognising medical concepts mentioned in social media messages (e.g. tweets) enables several applications for enhancing health quality of people in a community, e.g. real-time monitoring of infectious diseases in population. However, the discrepancy between the type of language used in social media and medical ontologies poses a major challenge. Existing studies deal with this challenge by employing techniques, such as lexical term matching and statistical machine translation. In this work, we handle the medical concept normalisation at the semantic level. We investigate the use of neural networks to learn the transition between layman's language used in social media messages and formal medical language used in the descriptions of medical concepts in a standard ontology. We evaluate our approaches using three different datasets, where social media texts are extracted from Twitter messages and blog posts. Our experimental results show that our proposed approaches significantly and consistently outperform existing effective baselines, which achieved state-of-the-art performance on several medical concept normalisation tasks, by up to 44%.

1 Introduction

Existing studies (O'Connor et al., 2014; Limsopatham and Collier, 2015a; Limsopatham and Collier, 2015b) have shown that data from social media (e.g. Twitter[1] and Facebook[2]) can be leveraged to improve the understanding of patients' experience in healthcare, such as the spread of infectious diseases and side-effects of drugs. However, the lexical and grammatical variability of the language used in social media poses a key challenge for extracting information (Baldwin et al., 2013; O'Connor et al., 2014). In particular, the frequent use of informal language, non-standard grammar and abbreviation forms, as well as typos in social media messages has to be taken into account by effective information extraction systems.

The task of medical concept normalisation for social media text, which aims to map a variable length social media message to a medical concept in some external coding system, is faced with a similar challenge (Limsopatham and Collier, 2015b). Traditional approaches, e.g. (Ristad and Yianilos, 1998; Aronson, 2001; Lu et al., 2011; McCallum et al., 2012), used proximity matching or heuristic string matching rules based on dictionary lookup when mapping texts to medical concepts. For example, Ristad and Yianilos (1998) incorporated edit-distance when mapping similar texts. The MetaMap system of Aronson (2001) applied a rule-based approach using pre-defined variants of terms when mapping texts to medical concepts in the UMLS Metathesaurus[3]. However, as shown in Table 1, existing string matching techniques may not be able to map the social media message "moon face and 30 lbs in 6 weeks" to the medical concept 'Weight Gain', or map "head spinning a little" to 'Dizziness', as no words in the social media messages and the description of the medical concepts correspond. Recent studies, e.g. (Leaman et al., 2013; Leaman and Lu, 2014; Limsopatham and Collier, 2015a), applied machine learning techniques to take into account relationships between different words (e.g. synonyms) when performing normal-

[1] http://twitter.com
[2] http://facebook.com
[3] https://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1014–1023,
Berlin, Germany, August 7-12, 2016.

Social media message	Description of corresponding medical concept
lose my appetite	Loss of appetite
i don't hunger or thirst	Loss of Appetite
hungry	Hunger
moon face and 30 lbs in 6 weeks	Weight Gain
gained 7 lbs	Weight Gain
lose the 10 lbs	Body Weight Decreased
feeling dizzy ...	Dizziness
head spinning a little	Dizziness
terrible headache!!	Headache

Table 1: Examples of social media messages and their related medical concepts.

isation. For instance, the *DNorm* system of Leaman et al. (2013), which achieved state-of-the-art performance on several medical concept normalisation tasks for medical articles (Doğan et al., 2014) and patient records (Suominen et al., 2013), used a pairwise learning-to-rank technique to learn the similarity between different terms when performing concept normalisation. Limsopatham and Collier (2015a) leveraged translations between the informal language used in social media and the formal language used in the description of medical concepts in an ontology. However, we argue that effective concept normalisation requires a system to take into account the semantics of social media messages and medical concepts. For example, to be able to map from the social media message "i don't hunger or thirst" to the medical concept 'Loss of Appetite', a normalisation system has to take into account the semantics of the whole message; otherwise, "i don't hunger or thirst" may be mapped to the medical concept 'Hunger', because they contain the term "hunger" in common.

In this work, we go beyond string matching. We propose to learn and exploit the semantic similarity between texts from social media messages and medical concepts using deep neural networks. In particular, we investigate the use of techniques from two families of deep neural networks, i.e. a convolutional neural network (CNN) and a recurrent neural network (RNN), to learn the mapping between social media texts and medical concepts. We evaluate our approaches using three different datasets that contain messages from Twitter and blog posts. Our experimental results show that our proposed approaches significantly outperform existing strong baselines (e.g. DNorm) across all of the three datasets. The performance improvement is by up to 44%.

The main contributions of this paper are threefold:

1. We propose two novel approaches based on CNN and RNN for medical concept normalisation.

2. We introduce two datasets with the gold-standard mappings between medical concepts and social media texts extracted from tweets and blog posts, respectively.

3. We thoroughly evaluate our proposed approaches using these two datasets and an existing dataset of tweets related to the topic of adverse drug reactions (ADRs) (Limsopatham and Collier, 2015a).

The remainder of this paper is organised as follows. In Section 2, we discuss related work and position our paper in the literature. Section 3 introduces our neural network approaches for medical concept normalisation. We describe our experimental setup and empirically evaluate our proposed approaches in Sections 4 and 5, respectively. We provide further analysis and discussion of our approaches in Section 6. Finally, Section 7 provides concluding remarks.

2 Related Work

Existing techniques for concept normalisation are mostly based on string matching (e.g. (Tsuruoka et al., 2007; Ristad and Yianilos, 1998; Lu et al., 2011; McCallum et al., 2012). For example, McCallum et al. (2012) used conditional random field to learn edit distance between phrases. In the medical domain, Tsuruoka et al. (2007) learned mappings between phrases in medical documents and medical concepts by using string matching features, such as character bigrams and

common tokens. Meanwhile, Metke-Jimenez and Karimi (2015), and O'Connor et al. (2014) used term weighting techniques, such as TF-IDF and BM25 (Robertson and Zaragoza, 2009) to retrieve relevant concepts. We tackle the concept normalisation task in a different manner. In particular, we use deep neural networks to capture the similarity and/or dependency between terms and effectively represent a given social media message in a low dimensional vector representation, before mapping it to a medical concept.

Another research area related to this work is the exploitation of word embeddings (i.e. distributed vector representation of words). It has been empirically shown that word embeddings can capture semantic and syntactic similarities between words (Turian et al., 2010; Mikolov et al., 2013b; Pennington et al., 2014; Levy and Goldberg, 2014). The cosine similarity between vectors of words has a positive correlation with the semantic similarity between them (Mikolov et al., 2013b; Pennington et al., 2014). Importantly, word embeddings have been effectively used for several NLP tasks, such as named entity recognition (Passos et al., 2014), machine translation (Mikolov et al., 2013a) and part-of-speech tagging (Turian et al., 2010). In the context of concept normalisation, Limsopatham and Collier (2015a) showed that effective performance could be achieved by mapping the processed social media messages and medical concepts using the similarity of their embeddings. In this work, we use word embeddings as inputs of deep neural networks, which would allow an effective representation of words when learning the concept normalisation.

Neural networks, such as convolutional neural networks (CNN) and recurrent neural networks (RNN), have been effectively applied to NLP tasks, such as NER, sentiment classifications and machine translation (Collobert et al., 2011; Kim, 2014; Bahdanau et al., 2014). For example, Collobert et al. (2011) effectively used a multilayer neural network for chunking, part-of-speech tagging, NER and semantic role labelling. Kim (2014) effectively used CNN with pre-built word embeddings when performing sentence classifications. Kalchbrenner et al. (2014) learned representation of sentences by using CNN. Meanwhile, Bahdanau et al. (2014) used RNN to encode a sentence written in one language (e.g. French) into a fixed length vector before decoding it to a sentence in another language (e.g. English) for translation. Socher et al. used recursive neural networks to model sentences for different tasks, including paraphrase detection (Socher et al., 2011) and sentence classification (Socher et al., 2013). In this paper, we investigate only the use of CNN and RNN for medical concept normalisation, as recursive neural networks require parse trees of input sentences while grammatical rules are typically ignored in social media messages.

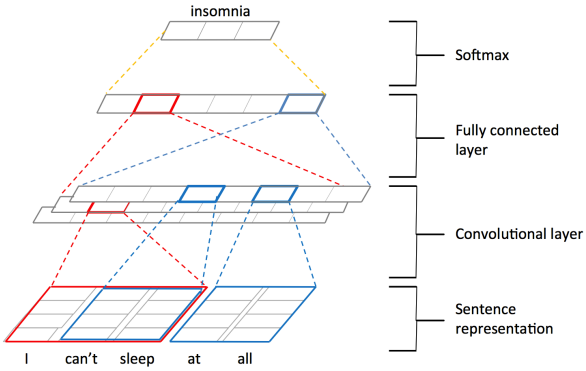


Figure 1: Our CNN architecture for medical concept normalisation.

3 Neural Networks for Concept Normalisation

Next, we introduce our medical concept normalisation approaches based on CNN and RNN in Sections 3.1 and 3.2, respectively.

3.1 CNN for Concept Normalisation

Our first approach uses CNN to learn the semantic representation of a social media message before mapping it to an appropriate medical concept. We use a CNN architecture with a single convolutional and pooling layer, as shown in Figure 1. Specifically, we firstly represent a given social media message of length l words (padded where necessary) using a sentence matrix $\mathbf{S} \in \mathbb{R}^{d \times l}$:

$$\mathbf{S} = \begin{bmatrix} | & | & | & | & \\ \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \dots & \mathbf{x}_l \\ | & | & | & | & \end{bmatrix} \quad (1)$$

where each column of $\mathbf{S}$ is the d-dimensional vector (i.e. embedding) $\mathbf{x}_i \in \mathbb{R}^d$ of each word in the social media message, which can be retrieved from pre-built word embeddings. This allows the model to take into account semantic features from the embeddings of each word.

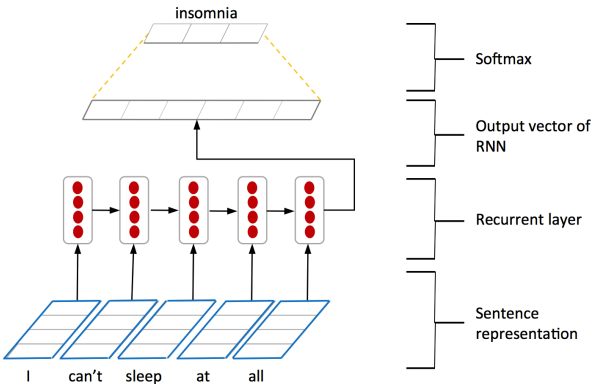


Figure 2: Our RNN architecture for medical concept normalisation.

We then apply a convolution operation using a filter $\mathbf{w} \in \mathbb{R}^{d \times h}$ to a window of h words. In particular, the filter $\mathbf{w}$ is convolved over the sequence of words in the sentence matrix $\mathbf{S}$ to create a feature matrix $\mathbf{C}$. Each feature c_i in $\mathbf{C}$ is extracted from a window of words $\mathbf{x}_{i:i+h-1}$, as follow:

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b) \quad (2)$$

where f is an activation function, such as sigmoid or tanh, and $b \in \mathbb{R}$ is a bias. Note that multiple filters (e.g. using different size h of window of words) can be used to extract multiple features.

This convolution operation enables the learning of dependencies between words from their semantic representation (i.e. word embeddings). In order to capture the most important features, max pooling (Collobert et al., 2011) is applied to take the maximum value of each row in the matrix $\mathbf{C}$:

$$\mathbf{c}_{max} = \begin{bmatrix} max(\mathbf{C}_{1,:}) \\ \vdots \\ max(\mathbf{C}_{d,:}) \end{bmatrix} \quad (3)$$

Finally, the fixed sized vector $\mathbf{c}_{max}$ forms a fully connected layer, which is used as inputs of softmax for multi-class classification. Indeed, the vector $\mathbf{c}_{max}$ provides a sentence representation that captures an extensional semantic information of the social media message for softmax to map to an appropriate medical concept.

3.2 RNN for Concept Normalisation

Our second approach uses RNN to capture the semantics of sequences of words in a social media message during normalisation. This approach is different from the CNN approach (introduced Section 3.1) in that instead of using the convolutional layer to learn the representation of social media messages (i.e. the vector representation at the fully connected layer), our RNN approach deploys a recurrent layer, as shown in Figure 2. Similar to the CNN approach, we initially represent a social media message of length l words using a sentence matrix $\mathbf{S} \in \mathbb{R}^{d \times l}$, as in Equation (1).

Then, the recurrent layer processes the vector $\mathbf{x}_i$ of each word in the social media message sequentially and produces a hidden state output $\mathbf{h}_i \in \mathbb{R}^k$, where $k \in \mathbb{Z}$ and $k > 0$. Importantly, when processing each input vector $\mathbf{x}_i$, the hidden state output $\mathbf{h}_{i-1}$ from the previous word is also recursively taken into account:

$$\mathbf{h}_i = f(\mathbf{h}_{i-1}, \mathbf{x}_i) \quad (4)$$

where f is a recurrent unit, such as long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho et al., 2014).

Finally, the hidden state output $\mathbf{h}_l$, which is the output from processing the last word of the social media message, is used as an input of the softmax for identifying the appropriate concept, in the same manner as the vector at the fully connected layer of the CNN approach in Section 3.1.

	TwADR-S	TwADR-L	AskAPatient
$\|Q\|$	201	1,436	8,662
$\|V_Q\|$	488	995	2,872
$\|C\|$	58	2,200	1,036
$\|V_C\|$	98	2,394	1,200
$\|Q \mapsto C\|_{avg}$	3.4655	0.6428	8.3610
$\|Q \mapsto C\|_{SD}$	5.6264	3.3168	39.2009
$\|Q \mapsto C\|_{min}$	1	0	1
$\|Q \mapsto C\|_{max}$	35	58	1,073

Table 2: Statistics of the datasets used in the experiments. $|Q|$: Number of queries. $|V_Q|$: Vocabulary size of queries. $|C|$: Number of target concepts. $|V_C|$: Vocabulary size of definition of target concepts. $|Q \mapsto C|_{avg}$ and $|Q \mapsto C|_{SD}$: Average number of queries mapped to each target concept, and its standard deviation (SD). $|Q \mapsto C|_{min}$ and $|Q \mapsto C|_{max}$: Mininum and maximum number of queries mapped to a given target concept, respectively.

4 Experimental Setup

4.1 Datasets

To evaluate our proposed approaches, we use three different datasets (namely, *TwADR-S*, *TwADR-L*

and *AskAPatient*)[4], where the task is to map a social media phrase to a relevant medical concept. In these datasets, a given social media phrase is mapped to only one medical concept. Table 2 shows statistics for the three datasets. In particular, TwADR-S is the dataset provided by Limsopatham and Collier (2015a), which contains 201 Twitter phrases and their corresponding SNOMED-CT[5] concept. The total number of target concepts is 58, while on average a medical concept can be mapped by 3.47 queries with the standard deviation of 5.63.

The TwADR-L dataset is our new dataset that we constructed from a collection of three months of tweets (between July and November 2015), downloaded using the Twitter Streaming API[6] by filtering using the name of a pre-defined set of drugs, which have been used in the literature for ADR profiling (e.g. cognitive enhancers) (Bender et al., 2007). These tweets were sampled and then annotated by undergraduate-level linguists. This collection contains 1,436 Twitter phrases that can be mapped to one of 2,220 medical concepts from the SIDER 4 database of drug profiles[7]. Note that 1,947 from the 2,220 concepts are not relevant to any of the Twitter phrases.

For the AskAPatient dataset, we extracted the gold-standard mappings of social media messages and medical concepts from the ADR annotation collection of Karimi et al. (2015). Our AskAPatient dataset contains 8,662 phrases[8], each of which can be mapped to one of the 1,036 medical concepts from SNOMED-CT and AMT (the Australian Medicines Terminology). We expect this dataset to be less difficult than TwADR-S and TwADR-L, as the nature of blog posts is less informal and ambiguous than Twitter messages.

For each of the datasets, we randomly divide it into ten equally folds, so that our approaches and the baselines would be trained on the same sets of data. We evaluate our approaches based on the accuracy performance, averaged across the ten folds. The significant difference between the performance of our approaches and the baselines is measured using the paired t-test ($p < 0.05$).

[4]TwADR-L and AskAPatient datasets are available on Zenodo.org (DOI:http://dx.doi.org/10.5281/zenodo.55013).

[5]http://www.ihtsdo.org/snomed-ct.

[6]https://dev.twitter.com/streaming/overview

[7]http://sideeffects.embl.de/

[8]From blog posts on http://www.askapatient.com website.

4.2 Pre-trained Word Embeddings

As our CNN (Section 3.1) and RNN (Section 3.2) approaches require word vectors as inputs, we investigate the use of two different pre-trained word embeddings. The first word embeddings (denoted, *GNews*) are the publicly available 300-dimension embeddings (vocabulary size of 3M) that were induced from 100 billion words from Google News using word2vec (Mikolov et al., 2013b)[9], which has been shown to be effective for several tasks (Baroni et al., 2014; Kim, 2014). The second word embeddings (denoted, *BMC*) induced from 854M words of medical articles downloaded from BioMed Central[10] by using the skip-gram model from word2vec (with default parameters). The BMC embeddings also have 300 dimension. For the words that do not existing in any embeddings, we use a vector of random values sampled from $[-0.25, 0.25]$.

As an alternative, we also use randomly generated embeddings (denoted, *Rand*) with 300 dimensions, where a vector representation of each word is randomly sampled from $[-0.25, 0.25]$. This allows the investigation of the effectiveness of our approaches when the semantic information from pre-built embeddings is not available.

4.3 Parameters of Our CNN and RNN Approaches

For our CNN approach, we use the filter $\mathbf{w}$ with the window size h of 3, 4 and 5, each of which with 100 feature maps, which have shown to be effective for modelling sentences in sentiment analysis (Kim, 2014). For the RNN, we use gated recurrent unit (GRU) (Cho et al., 2014) and set the size k of the output vector of each recurrent unit to 100.

In addition, for both CNN and RNN, we use rectifier linear unit (ReLU) (Nair and Hinton, 2010) as activation functions. We also apply L_2 regularisation of the weight vectors. We train the models over a mini-batch of size 50 to minimise the negative log-likelihood of correct predictions. The stochastic gradient descent with back-propagation is performed using Adadelta update rule (Zeiler, 2012). We initially set the number of epochs for training both CNN and RNN approaches to be 100, and allow the models to update the input

[9]https://code.google.com/p/word2vec/

[10]http://www.biomedcentral.com/about/datamining

embeddings in the sentence matrix $\mathbf{S}$. Later, in Sections 6.2 and 6.3, we discuss the performance achieved as we vary the number of epochs used for training the models, and the performance achieves when we allow and do not allow the models to update the input embeddings, respectively.

4.4 Baselines

We consider five different baselines as follows:

1. *TF-IDF*: A traditional term matching-based approach, using the TF-IDF score.

2. *BM25*: A traditional term matching-based approach, using the BM25 score, which has shown to be effective for several text retrieval tasks (Robertson and Zaragoza, 2009)

3. *EmbSim*: The cosine similarity between the word vector representation of a social media phrase and the description of a medical concept. If the phrase (or the description) contains several words, we represent it by adding up the values of the same dimension of the embedding of each word.

4. *DNorm*: A machine learning-based approach that exploits the relationships between words (e.g. synonyms) learned from training data (Leaman and Lu, 2014). This approach achieved state-of-the-art performance for several medical concept normalisation tasks (Suominen et al., 2013; Doğan et al., 2014). Note that we customise the open-source version[11] of DNorm to enable the mapping to a specific set of the target concepts for each dataset.

5. *P-MT*: The concept normalisation approach that translates social media texts to a formal medical text before mapping to appropriate medical concepts using the cosine similarity of their embeddings (Limsopatham and Collier, 2015a). We use the variant where the top-5 translations are used to map the medical concepts by taking the ranked position into account. We calculate the cosine similarity using either the GNews or the BMC embeddings.

6. *LogisticRegression*: A variant of our proposed approaches where we concatenate embeddings of terms (padded where necessary) in each social media phrase into a fixed-size sentence vector, before using this vector as input features for a multi-class logistic regression classifier.

Another possible baseline is a word-sense disambiguation system, such as IMS (Zhong and Ng, 2010). Nevertheless, the results from our initial experiments using IMS showed that it could not perform effectively on the three datasets. This is because the performance of IMS depends heavily on the contexts (i.e. words surrounding the input phrase); however, such contexts are not available in any of the three datasets. Therefore, we do not report the performance of IMS in this paper.

Note that for the baselines that require training data (i.e. DNorm and P-MT) and our two proposed approaches, apart from the training data provides with each fold of the datasets, we also train them using the descriptions of the target medical concepts, as these data are also used by the non-supervised baselines (i.e. TF-IDF, BM25 and EmbSim).

5 Experimental Results

In this section, we compare the performance of our CNN and RNN approaches for medical concept normalisation against the six baselines, introduced in Section 4.4. Table 3 compares the performances of our proposed approaches with the baselines in terms of accuracy on the three datasets (i.e. TwADR-S, TwADR-L, AskAPatient). Overall, as expected, the accuracy performance achieved by our approaches and the baselines on the AskAPatient dataset is higher than the TwADR-L and TwADR-S. This is due to nature use of language in Twitter, which is more ambiguous and informal than blog posts. When comparing among the existing baseline approaches, we observe that *DNorm* and *P-MT* are the most effective baselines. In particular, *DNorm* outperforms the other baselines for the TwADR-S (accuracy 0.2983) and AskAPatient (accuracy 0.7339) datasets, while *P-MT* with GNews embeddings is the most effective baseline for the TwADR-L dataset (accuracy 0.3371). In addition, term matching-based approaches, i.e. *TF-IDF* (accuracy 0.1638, 0.2293 and 0.5547, respectively) and *BM25* (accuracy 0.1638, 0.2300 and 0.5546), achieve almost similar performances, which are also comparable to the performances of *EmbSim* baselines. When comparing the effectiveness of different

[11] http://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/tmTools/#DNorm

Approach	Word Embeddings	Accuracy		
		TwADR-S	TwADR-L	AskAPatient
TF-IDF	-	0.1638	0.2293	0.5547
BM25	-	0.1638	0.2300	0.5546
EmbSim	GNews	0.2494	0.2326	0.5422
EmbSim	BMC	0.1348	0.2057	0.5141
DNorm	-	**0.2983**	0.3099	**0.7339**
P-MT	GNews	0.2346	**0.3371**	0.7235
P-MT	BMC	0.1248	0.3114	0.7126
LogisticRegression	GNews	0.3186	0.3409	0.7767
LogisticRegression	BMC	0.3036	0.3548	0.7752
CNN	Rand	0.3229$^{\bullet}$	0.4267$^{*\circ\bullet}$	0.8013$^{*\circ\bullet}$
CNN	GNews	**0.4174**$^{*\circ\bullet}$	**0.4478**$^{*\circ\bullet}$	**0.8141**$^{*\circ\bullet}$
CNN	BMC	0.3921$^{*\circ\bullet}$	0.4415$^{*\circ\bullet}$	0.8139$^{*\circ\bullet}$
RNN	Rand	0.2936$^{\bullet}$	0.3791$^{*\circ\bullet}$	0.7991$^{*\circ\bullet}$
RNN	GNews	**0.3529**$^{*\circ\bullet}$	**0.3882**$^{*\circ\bullet}$	**0.7998**$^{*\circ\bullet}$
RNN	BMC	0.3331$^{\bullet}$	0.3847$^{*\circ\bullet}$	0.7996$^{*\circ\bullet}$

Table 3: The accuracy performance of our proposed approaches and the baselines. Significant differences ($p < 0.05$, paired t-test) compared to the *DNorm*, *P-MT with GNews embeddings*, and *P-MT with BMC embeddings*, are denoted *, $^{\circ}$ and $^{\bullet}$, respectively.

pre-trained embeddings used in *EmbSim* and *P-MT*, we observe that GNews is more effective than BMC for both approaches, across all of the three datasets.

Next, we discuss the performance of our *CNN* and *RNN* approaches. From Table 3, we observe that both *CNN* and *RNN* markedly outperform all of the existing baselines for all of the three datasets. When compared with *DNorm* and *P-MT with GNews* baselines, which are the most effective existing baselines, we observe that both *CNN* and *RNN* significantly ($p < 0.05$, paired t-test) outperform the two baselines for all of the three datasets. Indeed, for the TwADR-L dataset, *CNN with GNews* (accuracy 0.4478) outperforms *DNorm* (accuracy 0.3099) by 44%. In addition, the choice of embeddings has a marked impact on the achieved performance. In particular, the GNews embeddings benefit both *CNN* and *RNN* more than the BMC embeddings, which is in line with the previous finding that GNews is more useful than BMC for the *EmbSim* and *P-MT* baselines. On the other than, the randomly generated embeddings (i.e. Rand) are less useful. These results show that the semantics captured in word embeddings are useful for both *CNN* and *RNN* approaches for medical concept normalisation. However, for both *CNN* and *RNN*, the choice of embeddings that are employed has less impact on the performance for the AskAPatient dataset, which has greater number of training data.

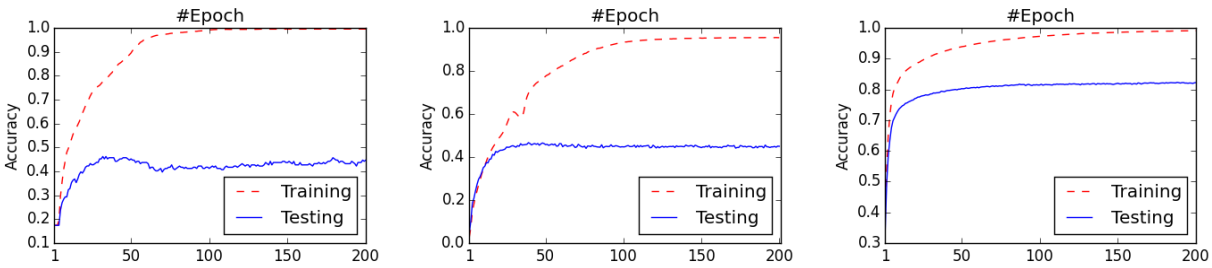
Furthermore, we observe that the *LogisticRegression* baseline, a variant of our proposed approach that uses the multi-class logistic regression instead of neural networks for identifying relevance concepts, also outperforms the all of the existing baselines. However, it performs worse than both *CNN* and *RNN* approaches. This shows that while logistic regression can exploit the semantics of embeddings of individual terms in social media texts (at the word level), it cannot learn the semantics of the whole phrase as effectively as CNN and RNN.

6 Analysis & Discussions

In this section, we further analyse the performance achieved by our proposed approaches. As the performance achieved by our CNN approach is better than that of our RNN approach, we discuss only our CNN approach in this section.

6.1 Failure Analysis

We first discuss the results achieved by the baselines and our CNN approach. As expected, we observe that all approaches perform very well for the social media phrases that lexically match with the definition of the medical concepts, e.g. the social media phrase "attention deficit disorder" is



(a) TwADR-S (b) TwADR-L (c) AskAPatient

Figure 3: The accuracy performance achieved by training with different numbers of epochs for the three datasets.

mapped to the medical concept 'Attention Deficit Disorder'. However, for a more complex phrases, such as "appetite on 10", "my appetite way up", "suppressed appetite", the baselines, including DNorm and P-MT, cannot effectively incorporate the modifiers of the word "appetite" in different phrases. For example, "appetite on 10", "my appetite way up" should be mapped to 'Increased Appetite', while "suppressed appetite" should be mapped to 'Loss of Appetite'. On the other hand, for social media phrases that do not have any terms in common with the definition of any medical concepts, all of the baselines performs poorly for most of the cases. For instance, even though DNorm can learn that the term "focusing" has some relationship with "concentration", it maps any phrases containing "focusing" to the 'Attention Concentration Difficulty' concept, including phrases, such as "focusing monster", which should be mapped to 'Consciousness Abnormal'. Our CNN approach could deal with most of these cases effectively, as it considers the semantic representation of the whole phrase during normalisation.

6.2 Impact of Number of Training Epochs

Next, we discuss the normalisation performance as we vary, between 1 and 200, the number of epochs used for training our CNN model. Figures 3(a), 3(b) and 3(c) show the performance in terms of accuracy achieved during training and testing for the TwADR-S, TWADR-L and AskAPatient datasets, respectively. We observe that training can be effectively achieved at around 60 - 70 epochs for the TwADR-S and TwADR-L datasets, and around 40 epochs for the AskAPatient dataset, before the performance becomes stable. We notice a gap between the performance achieved during training and testing, especially for the TwADR-S and TwADR-L datasets; however, this gap should be narrower if more training data are available.

Dataset	Accuracy	
	CNN with updated emb.	CNN with fixed emb.
TwADR-S	0.4174	**0.4369**
TwADR-L	0.4478	**0.4590**
AskAPatient	**0.8141**$^{\bullet}$	0.7869

Table 4: The accuracy performance of our CNN approach with the GNews embeddings, when allowing (*updated emb.*) and not allowing (*fixed emb.*) the model to update the input word embeddings. Significant difference ($p < 0.05$, paired t-test) between the performance achieved by the two variants, on each dataset, is denoted $^{\bullet}$.

6.3 Impact of Fixed Embeddings

In this section, we compare the performance of our CNN with GNews embeddings when we allow (*updated emb.*) and when we do not allow (*fixed emb.*) the input embeddings to be updated. Table 4 reports the accuracy performance of the two variants for the three datasets. We observe that for TwADR-S and TwADR-L datasets, which are smaller datasets (dataset size of 201 and 1,436, respectively), a better performance can be achieved if the model is not allowed to update the embeddings of the input phrases. In contrast, for the AskAPatient dataset (dataset size of 8,662), allowing the model to update the embeddings results in a significantly (paired t-test, $p < 0.05$) better performance. We observe the same trends of performance when using BMC embeddings. These results suggest that for small datasets, we should leverage semantics from pre-built word embeddings and do not allow the model to update the

embeddings. Meanwhile, for a larger dataset, further performance improvement can be achieved by allowing the model to update the embeddings.

7 Conclusions

We have motivated the importance of semantics when normalising medical concepts in social media messages. In particular, as social media messages are typically ambiguous, we argue that effective concept normalisation should deal with them at the semantic level. To do so, we introduced two neural network-based approaches for medical concept normalisation, which are based on convolutional and recurrent neural network architectures. Our experimental results evaluated on three different social media datasets showed that both of our approaches markedly and significantly outperformed several strong baselines, including an existing approach that achieved state-of-the-art performance on several medical concept normalisation tasks. From the analysis of the results, we found that while some existing approaches can capture synonyms of words, they could not leverage the semantic meaning of the social media message. Our approaches overcomes this by learning the semantic representation of the social media message before passing it to a classifier to match an appropriate concept.

Acknowledgements

The authors wish to thank funding support from the EPSRC (grant number EP/M005089/1).

References

Alan R Aronson. 2001. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *AMIA*, pages 17–21.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources. In *IJCNLP*, pages 356–364.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *ACL*, pages 238–247.

Andreas Bender, Josef Scheiber, Meir Glick, John W Davies, Kamal Azzaoui, Jacques Hamon, Laszlo Urban, Steven Whitebread, and Jeremy L Jenkins. 2007. Analysis of pharmacology data and the prediction of adverse drug reactions and off-target effects from chemical structure. *ChemMedChem*, 2(6):861–873.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537.

Rezarta Islamaj Doğan, Robert Leaman, and Zhiyong Lu. 2014. Ncbi disease corpus: a resource for disease name recognition and concept normalization. *Journal of biomedical informatics*, 47:1–10.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*, pages 655–665.

Sarvnaz Karimi, Alejandro Metke-Jimenez, Madonna Kemp, and Chen Wang. 2015. Cadec: A corpus of adverse drug event annotations. *Journal of biomedical informatics*, 55:73–81.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Robert Leaman and Zhiyong Lu. 2014. Automated disease normalization with low rank approximations. In *ACL*, pages 24–28.

Robert Leaman, Rezarta Islamaj Doğan, and Zhiyong Lu. 2013. Dnorm: disease name normalization with pairwise learning to rank. *Bioinformatics*, 29(22):2909–2917.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *ACL*, pages 302–308.

Nut Limsopatham and Nigel Collier. 2015a. Adapting phrase-based machine translation to normalise medical terms in social media messages. In *EMNLP*, pages 1675–1680.

Nut Limsopatham and Nigel Collier. 2015b. Towards the semantic interpretation of personal health messages from social media. In *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics*, UCUI '15, pages 27–30, New York, NY, USA. ACM.

Zhiyong Lu, Hung-Yu Kao, Chih-Hsuan Wei, Minlie Huang, Jingchen Liu, Cheng-Ju Kuo, Chun-Nan Hsu, Richard TH Tsai, Hong-Jie Dai, Naoaki Okazaki, et al. 2011. The gene normalization task in biocreative iii. *BMC bioinformatics*, 12(Suppl 8):S2.

Andrew McCallum, Kedar Bellare, and Fernando Pereira. 2012. A conditional random field for discriminatively-trained finite-state string edit distance. *arXiv preprint arXiv:1207.1406*.

Alejandro Metke-Jimenez and Sarvnaz Karimi. 2015. Concept extraction to identify adverse drug reactions in medical forums: A comparison of algorithms. *arXiv preprint arXiv:1504.06936*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814.

Karen O'Connor, Pranoti Pimpalkhute, Azadeh Nikfarjam, Rachel Ginn, Karen L Smith, and Graciela Gonzalez. 2014. Pharmacovigilance on twitter? mining tweets for adverse drug reactions. In *AMIA*, volume 2014, pages 924–933.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *arXiv preprint arXiv:1404.5367*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Eric Sven Ristad and Peter N Yianilos. 1998. Learning string-edit distance. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5):522–532.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, pages 801–809.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Hanna Suominen, Sanna Salanterä, Sumithra Velupillai, Wendy W Chapman, Guergana Savova, Noemie Elhadad, Sameer Pradhan, Brett R South, Danielle L Mowery, Gareth JF Jones, et al. 2013. Overview of the share/clef ehealth evaluation lab 2013. In *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, pages 212–231. Springer.

Yoshimasa Tsuruoka, John McNaught, Sophia Ananiadou, et al. 2007. Learning string similarity measures for gene/protein name dictionary look-up using logistic regression. *Bioinformatics*, 23(20):2768–2774.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *ACL*, pages 384–394.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL*, pages 78–83.

Agreement-based Learning of Parallel Lexicons and Phrases from Non-Parallel Corpora

Chunyang Liu[†], **Yang Liu**[†#*], **Huanbo Luan**[†], **Maosong Sun**[†#], and **Heng Yu**[‡]
[†] State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
[#] Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China
[‡] Samsung R&D Institute of China, Beijing 100028, China
{liuchunyang2012,liuyang.china,luanhuanbo}@gmail.com, sms@tsinghua.edu.cn
h0517.yu@samsung.com

Abstract

We introduce an agreement-based approach to learning parallel lexicons and phrases from non-parallel corpora. The basic idea is to encourage two asymmetric latent-variable translation models (i.e., source-to-target and target-to-source) to agree on identifying latent phrase and word alignments. The agreement is defined at both word and phrase levels. We develop a Viterbi EM algorithm for jointly training the two unidirectional models efficiently. Experiments on the Chinese-English dataset show that agreement-based learning significantly improves both alignment and translation performance.

1 Introduction

Parallel corpora, which are large collections of parallel texts, serve as an important resource for inducing translation correspondences, either at the level of words (Brown et al., 1993; Smadja and McKeown, 1994; Wu and Xia, 1994) or phrases (Kupiec, 1993; Melamed, 1997; Marcu and Wong, 2002; Koehn et al., 2003). However, the availability of large-scale, wide-coverage corpora still remains a challenge even in the era of big data: parallel corpora are usually only existent for resource-rich languages and restricted to limited domains such as government documents and news articles.

Therefore, intensive attention has been drawn to exploiting non-parallel corpora for acquiring translation correspondences. Most previous efforts have concentrated on learning parallel lexicons from non-parallel corpora, including parallel sentence and lexicon extraction via bootstrapping (Fung and Cheung, 2004), inducing parallel lexicons via canonical correlation analysis (Haghighi et al., 2008), training IBM models on monolingual corpora as decipherment (Ravi and Knight, 2011; Nuhn et al., 2012; Dou et al., 2014), and deriving parallel lexicons from bilingual word embeddings (Vulić and Moens, 2013; Mikolov et al., 2013; Vulić and Moens, 2015).

Recently, a number of authors have turned to a more challenging task: learning parallel phrases from non-parallel corpora (Zhang and Zong, 2013; Dong et al., 2015). Zhang and Zong (2013) present a method for retrieving parallel phrases from non-parallel corpora using a seed parallel lexicon. Dong et al. (2015) continue this line of research to further introduce an iterative approach to joint learning of parallel lexicons and phrases. They introduce a corpus-level latent-variable translation model in a non-parallel scenario and develop a training algorithm that alternates between (1) using a parallel lexicon to extract parallel phrases from non-parallel corpora and (2) using the extracted parallel phrases to enlarge the parallel lexicon. They show that starting from a small seed lexicon, their approach is capable of learning both new words and phrases gradually over time.

However, due to the structural divergence between natural languages as well as the presence of noisy data, only using asymmetric translation models might be insufficient to accurately identify parallel lexicons and phrases from non-parallel corpora. Dong et al. (2015) report that the accuracy on Chinese-English dataset is only around 40% after running for 70 iterations. In addition, their approach seems prone to be affected by noisy data in non-parallel corpora as the accuracy drops significantly with the increase of noise.

Since asymmetric word alignment and phrase alignment models are usually complementary, it is natural to combine them to make more accurate predictions. In this work, we propose to in-

*Corresponding author: Yang Liu.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1024–1033,
Berlin, Germany, August 7-12, 2016.

troduce agreement-based learning (Liang et al., 2006; Liang et al., 2008) into extracting parallel lexicons and phrases from non-parallel corpora. Based on the latent-variable model proposed by Dong et al. (2015), we propose two kinds of loss functions to take into account the agreement between both phrase alignment and word alignment in two directions. As the inference is intractable, we resort to a Viterbi EM algorithm to train the two models efficiently. Experiments on the Chinese-English dataset show that agreement-based learning is more robust to noisy data and leads to substantial improvements in phrase alignment and machine translation evaluations.

2 Background

Given a monolingual corpus of source language phrases $E = \{\mathbf{e}^{(s)}\}_{s=1}^{S}$ and a monolingual corpus of target language phrases $F = \{\mathbf{f}^{(t)}\}_{t=1}^{T}$, we assume there exists a parallel corpus $D = \{\langle \mathbf{e}^{(s)}, \mathbf{f}^{(t)} \rangle | \mathbf{e}^{(s)} \leftrightarrow \mathbf{f}^{(t)}\}$, where $\mathbf{e}^{(s)} \leftrightarrow \mathbf{f}^{(t)}$ denotes that $\mathbf{e}^{(s)}$ and $\mathbf{f}^{(t)}$ are translations of each other.

As a long sentence in E is usually unlikely to have an translation in F and vise versa, most previous efforts build on the assumption that *phrases* are more likely to have translational equivalents on the other side (Munteanu and Marcu, 2006; Cettolo et al., 2010; Zhang and Zong, 2013; Dong et al., 2015). Such a set of phrases can be constructed by collecting either constituents of parsed sentences or strings with hyperlinks on webpages (e.g., Wikipedia). Therefore, we assume the two monolingual corpora are readily available and focus on how to extract D from E and F.

To address this problem, Dong et al. (2015) introduce a corpus-level latent-variable translation model in a non-parallel scenario:

$$P(F|E;\boldsymbol{\theta}) = \sum_{\mathbf{m}} \underbrace{P(F,\mathbf{m}|E;\boldsymbol{\theta})}_{\textit{phrase alignment}}, \quad (1)$$

where $\mathbf{m}$ is *phrase alignment* and $\boldsymbol{\theta}$ is a set of model parameters. Each target phrase $\mathbf{f}^{(t)}$ is restricted to connect to exactly one source phrase: $\mathbf{m} = (\mathbf{m}_1, \ldots, \mathbf{m}_t, \ldots \mathbf{m}_T)$, where $\mathbf{m}_t \in \{0, 1, \ldots, S\}$. For example, $\mathbf{m}_t = s$ denotes that $\mathbf{f}^{(t)}$ is aligned to $\mathbf{e}^{(s)}$. Note that $\mathbf{e}^{(0)}$ represents an empty source phrase.

They follow IBM Model 1 (Brown et al., 1993) to further decompose the model as

$$P(F,\mathbf{m}|E;\boldsymbol{\theta}) = \frac{p(T|S)}{(S+1)^T} \prod_{t=1}^{T} P(\mathbf{f}^{(t)}|\mathbf{e}^{(\mathbf{m}_t)};\boldsymbol{\theta}), \quad (2)$$

where $P(\mathbf{f}^{(t)}|\mathbf{e}^{(\mathbf{m}_t)};\boldsymbol{\theta})$ is a *phrase translation* model that can be further defined as

$$\begin{aligned} & P(\mathbf{f}^{(t)}|\mathbf{e}^{(\mathbf{m}_t)};\boldsymbol{\theta}) \\ &= \delta(\mathbf{m}_t, 0)\epsilon + \\ & \quad (1-\delta(\mathbf{m}_t,0)) \sum_{\mathbf{a}} \underbrace{P(\mathbf{f}^{(t)},\mathbf{a}|\mathbf{e}^{(\mathbf{m}_t)};\boldsymbol{\theta})}_{\textit{word alignment}}. \end{aligned} \quad (3)$$

Dong et al. (2015) distinguish between *empty* and *non-empty* phrase translations. If a target phrase $\mathbf{f}^{(t)}$ is aligned to the empty source phrase $\mathbf{e}^{(0)}$ (i.e., $\mathbf{m}_t = 0$), they set the phrase translation probability to a fixed number ϵ. Otherwise, conventional word alignment models such as IBM Model 1 can be used for non-empty phrase translation:

$$\begin{aligned} & P(\mathbf{f}^{(t)},\mathbf{a}|\mathbf{e}^{(\mathbf{m}_t)};\boldsymbol{\theta}) \\ &= \frac{p(J^{(t)}|I^{(\mathbf{m}_t)})}{(I^{(\mathbf{m}_t)}+1)^{J^{(t)}}} \prod_{j=1}^{J^{(t)}} p(\mathbf{f}_j^{(t)}|\mathbf{e}_{\mathbf{a}_j}^{(\mathbf{m}_t)}), \end{aligned} \quad (4)$$

where $p(J|I)$ is a *length* model and $p(f|e)$ is a *translation* model. We use $J^{(t)}$ to denote the length of $\mathbf{f}^{(t)}$.

Therefore, the latent-variable model involves two kinds of latent structures: (1) *phrase alignment* $\mathbf{m}$ between source and target phrases, (2) *word alignment* $\mathbf{a}$ between source and target words within phrases.

Given the two monolingual corpora E and F, the training objective is to maximize the likelihood of the training data:

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \Big\{ \mathcal{L}(\boldsymbol{\theta}) \Big\}, \quad (5)$$

where

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) = & \log P(F|E;\boldsymbol{\theta}) - \\ & \sum_I \lambda_I \Big(\sum_J p(J|I) - 1 \Big) - \\ & \sum_e \gamma_e \Big(\sum_f p(f|e) - 1 \Big) - \\ & \sum_f \sum_e \sigma(f,e,\mathbf{d}) \log \frac{\sigma(f,e,\mathbf{d})}{p(f|e)}. \end{aligned} \quad (6)$$

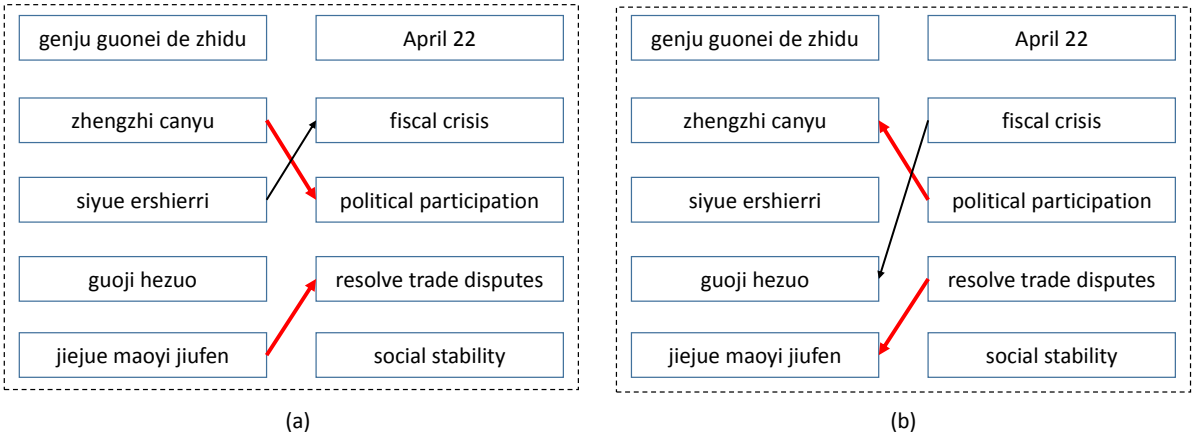


Figure 1: Agreement between (a) Chinese-to-English and (b) English-to-Chinese phrase alignments. The arrows indicate translation directions. The links on which two models agree are highlighted in bold red. The *outer agreement* loss function (see Eq. (14)) aims to encourage the agreement at the phrase level.

Note that $\mathbf{d}$ is a small seed parallel lexicon for initializing training [1] and $\sigma(f, e, \mathbf{d})$ checks whether an entry $\langle f, e \rangle$ exists in $\mathbf{d}$.

Given the monolingual corpora and the optimized model parameters, the Viterbi phrase alignment is calculated as

$$\mathbf{m}^* = \operatorname*{argmax}_{\mathbf{m}} \Big\{ P(F, \mathbf{m}|E; \boldsymbol{\theta}^*) \Big\} \quad (7)$$

$$= \operatorname*{argmax}_{\mathbf{m}} \Big\{ \prod_{t=1}^{T} P(\mathbf{f}^{(t)}|\mathbf{e}^{(\mathbf{m}_t)}; \boldsymbol{\theta}^*) \Big\}. \quad (8)$$

Finally, parallel lexicons can be derived from the translation probability table of IBM model 1 $\boldsymbol{\theta}^*$ and parallel phrases can be collected from the Viterbi phrase alignment $\mathbf{m}^*$. This process iterates and enlarges parallel lexicons and phrases gradually over time.

As it is very challenging to extract parallel phrases from non-parallel corpora, unidirectional models might only capture partial aspects of translation modeling on non-parallel corpora. Indeed, Dong et al. (2015) find that the accuracy of phrase alignment is only around 50% on the Chinese-English dataset. More importantly, their approach seems to be vulnerable to noise as the accuracy drops significantly with the increase of noise. As source-to-target and target-to-source translation models are usually complementary (Och and Ney, 2003; Koehn et al., 2003; Liang et al., 2006), it is appealing to combine them to improve alignment accuracy.

[1] Due to the difficulty of learning translation correspondences from non-parallel corpora, many authors have assumed that a small seed lexicon is readily available (Gaussier et al., 2004; Zhang and Zong, 2013; Vulić and Moens, 2013; Mikolov et al., 2013; Dong et al., 2015).

3 Approach

3.1 Agreement-based Learning

The basic idea of our work is to encourage the source-to-target and target-to-source translation models to agree on both phrase and word alignments.

For example, Figure 1 shows two example Chinese-to-English and English-to-Chinese phrase alignments on the same non-parallel data. As each model only captures partial aspects of translation modeling, our intuition is that the links on which two models agree (highlighted in red) are more likely to be correct.

More formally, let $P(F|E; \overrightarrow{\boldsymbol{\theta}})$ be a source-to-target translation model and $P(E|F; \overleftarrow{\boldsymbol{\theta}})$ be a target-to-source model, where $\overrightarrow{\boldsymbol{\theta}}$ and $\overleftarrow{\boldsymbol{\theta}}$ are corresponding model parameters. We use $\overrightarrow{\mathbf{m}} = (\overrightarrow{\mathbf{m}}_1, \ldots, \overrightarrow{\mathbf{m}}_t, \ldots, \overrightarrow{\mathbf{m}}_T)$ to denote source-to-target phrase alignment. Likewise, the target-to-source phrase alignment is denoted by $\overleftarrow{\mathbf{m}} = (\overleftarrow{\mathbf{m}}_1, \ldots, \overleftarrow{\mathbf{m}}_s, \ldots, \overleftarrow{\mathbf{m}}_S)$.

To ease the comparison between $\overrightarrow{\mathbf{m}}$ and $\overleftarrow{\mathbf{m}}$, we represent them as sets of non-empty links equivalently:

$$\overrightarrow{\mathbf{m}} = \Big\{ \langle \overrightarrow{\mathbf{m}}_t, t \rangle | \overrightarrow{\mathbf{m}}_t \neq 0 \Big\} \quad (9)$$

$$\overleftarrow{\mathbf{m}} = \Big\{ \langle s, \overleftarrow{\mathbf{m}}_s \rangle | \overleftarrow{\mathbf{m}}_s \neq 0 \Big\}. \quad (10)$$

For example, suppose the source-to-target and target-to-source phrase alignments are $\overrightarrow{\mathbf{m}} =$

1: **procedure** VITERBIEM(E, F, $\mathbf{d}$)
2: Initialize $\mathbf{\Theta}^{(0)}$
3: **for all** $k = 1, \ldots, K$ **do**
4: $\hat{\mathbf{m}}^{(k)} \leftarrow$ SEARCH($E, F, \mathbf{\Theta}^{(k-1)}$)
5: $\mathbf{\Theta}^{(k)} \leftarrow$ UPDATE($E, F, \mathbf{d}, \hat{\mathbf{m}}^{(k)}$)
6: **end for**
7: **return** $\hat{\mathbf{m}}^{(K)}, \mathbf{\Theta}^{(K)}$
8: **end procedure**

Figure 2: A Viterbi EM algorithm for agreement-based learning of parallel lexicons and phrases from non-parallel corpora. F and E are non-parallel corpora, $\mathbf{d}$ is a seed parallel lexicon, $\mathbf{\Theta}^{(k)}$ is the set of model parameters at the k-th iteration, $\hat{\mathbf{m}}^{(k)}$ is the Viterbi phrase alignment on which two models agree at the k-th iteration.

$(2, 3, 0, 0)$ and $\overleftarrow{\mathbf{m}} = (0, 1, 2)$. The equivalent link sets are $\overrightarrow{\mathbf{m}} = \{\langle 2, 1\rangle, \langle 3, 2\rangle\}$ and $\overleftarrow{\mathbf{m}} = \{\langle 2, 1\rangle, \langle 3, 2\rangle\}$. Therefore, $\overrightarrow{\mathbf{m}}$ is said to be *equal* to $\overleftarrow{\mathbf{m}}$ (i.e., $\delta(\overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}) = 1$).

Following Liang et al. (2006), we introduce a new training objective that favors the agreement between two unidirectional models:

$$\begin{aligned}
&\mathcal{J}(\overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}) \\
&= \log P(F|E; \overrightarrow{\boldsymbol{\theta}}) + \log P(E|F; \overleftarrow{\boldsymbol{\theta}}) - \\
&\quad \log \sum_{\overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}} P(\overrightarrow{\mathbf{m}}|E, F; \overrightarrow{\boldsymbol{\theta}}) P(\overleftarrow{\mathbf{m}}|F, E; \overleftarrow{\boldsymbol{\theta}}) \\
&\qquad \times \Delta(E, F, \overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}, \overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}), \qquad (11)
\end{aligned}$$

where the posterior probabilities in two directions are defined as

$$P(\overrightarrow{\mathbf{m}}|E, F; \overrightarrow{\boldsymbol{\theta}}) = \prod_{t=1}^{T} \frac{P(\mathbf{f}^{(t)}|\mathbf{e}^{(\overrightarrow{\mathbf{m}}_t)}; \overrightarrow{\boldsymbol{\theta}})}{\sum_{s=0}^{S} P(\mathbf{f}^{(t)}|\mathbf{e}^{(s)}; \overrightarrow{\boldsymbol{\theta}})} \quad (12)$$

$$P(\overleftarrow{\mathbf{m}}|F, E; \overleftarrow{\boldsymbol{\theta}}) = \prod_{s=1}^{S} \frac{P(\mathbf{e}^{(s)}|\mathbf{f}^{(\overleftarrow{\mathbf{m}}_s)}; \overleftarrow{\boldsymbol{\theta}})}{\sum_{t=0}^{T} P(\mathbf{e}^{(s)}|\mathbf{f}^{(t)}; \overleftarrow{\boldsymbol{\theta}})}. \quad (13)$$

The *loss function* $\Delta(E, F, \overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}, \overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}})$ measures the disagreement between the two models.

3.2 Outer Agreement

3.2.1 Definition

A straightforward loss function is to force the two models to generate identical phrase alignments:

$$\Delta_{\text{outer}}(E, F, \overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}, \overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}) = 1 - \delta(\overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}). \quad (14)$$

We refer to Eq. (14) as *outer agreement* since it only considers phrase alignment and ignores the word alignment within aligned phrases.

3.2.2 Training Objective

Since the outer agreement forces two models to generate identical phrase alignments, the training objective can be written as

$$\begin{aligned}
&\mathcal{J}_{\text{outer}}(\overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}) \\
&= \log P(F|E; \overrightarrow{\boldsymbol{\theta}}) + \log P(E|F; \overleftarrow{\boldsymbol{\theta}}) + \\
&\quad \log \sum_{\mathbf{m}} P(\mathbf{m}|E, F; \overrightarrow{\boldsymbol{\theta}}) P(\mathbf{m}|F, E; \overleftarrow{\boldsymbol{\theta}}), \qquad (15)
\end{aligned}$$

where $\mathbf{m}$ is a phrase alignment on which two models agree.

The partial derivatives of the training objective with respect to source-to-target model parameters $\overrightarrow{\boldsymbol{\theta}}$ are given by

$$\begin{aligned}
&\frac{\partial \mathcal{J}_{\text{outer}}(\overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}})}{\partial \overrightarrow{\boldsymbol{\theta}}} \\
&= \frac{\partial P(F|E; \overrightarrow{\boldsymbol{\theta}}) / \partial \overrightarrow{\boldsymbol{\theta}}}{P(F|E; \overrightarrow{\boldsymbol{\theta}})} + \\
&\quad \frac{\mathbb{E}_{\mathbf{m}|F,E;\overleftarrow{\boldsymbol{\theta}}} \left[\partial P(F|E; \overrightarrow{\boldsymbol{\theta}}) / \partial \overrightarrow{\boldsymbol{\theta}} \right]}{\sum_{\mathbf{m}} P(\mathbf{m}|E, F; \overrightarrow{\boldsymbol{\theta}}) P(\mathbf{m}|F, E; \overleftarrow{\boldsymbol{\theta}})}. \qquad (16)
\end{aligned}$$

The partial derivatives with respect to $\overleftarrow{\boldsymbol{\theta}}$ are defined likewise.

3.2.3 Training Algorithm

As the expectation in Eq. (16) is usually intractable to calculate due to the exponential search space of phrase alignment, we follow Dong et al. (2015) to use a Viterbi EM algorithm instead.

As shown in Figure 2, the algorithm takes a set of source phrases E, a set of target phrases F, and a seed parallel lexicon $\mathbf{d}$ as input (line 1). After initializing model parameters $\mathbf{\Theta} = \{\overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}\}$ (line 2), the algorithm calls the procedure ALIGN($F, E, \mathbf{\Theta}$) to compute the Viterbi phrase alignment between E and F on which two models agree. Then, the algorithm updates the two models by normalizing counts collected from the Viterbi phrase alignment. The process iterates for K iterations and returns the final Viterbi phrase alignment and model parameters.

3.2.4 Computing Viterbi Phrase Alignments

The procedure ALIGN($F, E, \mathbf{\Theta}$) computes the Viterbi phrase alignment $\hat{\mathbf{m}}$ between E and F on which two models agree as follows:

$$\begin{aligned}
\hat{\mathbf{m}} = \operatorname*{argmax}_{\mathbf{m}} \Big\{ &P(\mathbf{m}|E, F; \overrightarrow{\boldsymbol{\theta}}) \times \\
&P(\mathbf{m}|F, E; \overleftarrow{\boldsymbol{\theta}}) \Big\}. \qquad (17)
\end{aligned}$$

Unfortunately, due to the exponential search space of phrase alignment, computing $\hat{\mathbf{m}}$ is also intractable. As a result, we approximate it as the intersection of two unidirectional Viterbi phrase alignments:

$$\hat{\mathbf{m}} \approx \overrightarrow{\mathbf{m}}^* \cap \overleftarrow{\mathbf{m}}^*, \tag{18}$$

where the unidirectional Viterbi phrase alignments are calculated as

$$\overrightarrow{\mathbf{m}}^* = \underset{\overrightarrow{\mathbf{m}}}{\text{argmax}} \left\{ \prod_{t=1}^{T} P(\mathbf{f}^{(t)} | \mathbf{e}^{(\overrightarrow{\mathbf{m}}_t)}; \overrightarrow{\boldsymbol{\theta}}) \right\} \tag{19}$$

$$\overleftarrow{\mathbf{m}}^* = \underset{\overleftarrow{\mathbf{m}}}{\text{argmax}} \left\{ \prod_{s=1}^{S} P(\mathbf{e}^{(s)} | \mathbf{f}^{(\overleftarrow{\mathbf{m}}_s)}; \overleftarrow{\boldsymbol{\theta}}) \right\}. \tag{20}$$

The source-to-target Viterbi phrase alignment is calculated as

$$\overrightarrow{\mathbf{m}}^* = \underset{\overrightarrow{\mathbf{m}}}{\text{argmax}} \left\{ P(\overrightarrow{\mathbf{m}} | E, F; \overrightarrow{\boldsymbol{\theta}}) \right\} \tag{21}$$

$$= \underset{\overrightarrow{\mathbf{m}}}{\text{argmax}} \left\{ \prod_{t=1}^{T} P(\mathbf{f}^{(t)} | \mathbf{e}^{(\mathbf{m}_t)}; \overrightarrow{\boldsymbol{\theta}}) \right\}. \tag{22}$$

Dong et al. (2015) indicate that computing the Viterbi alignment for individual target phrases is *independent* and only need to focus on finding the most probable source phrase for each target phrase:

$$\overrightarrow{\mathbf{m}}_t^* = \underset{s \in \{0,1,\ldots,S\}}{\text{argmax}} \left\{ P(\mathbf{f}^{(t)} | \mathbf{e}^{(s)}; \overrightarrow{\boldsymbol{\theta}}) \right\}. \tag{23}$$

This can be cast as a translation retrieval problem (Zhang and Zong, 2013; Dong et al., 2014). Please refer to (Dong et al., 2015) for more details. The target-to-source Viterbi phrase alignment can be calculated similarly.

3.2.5 Updating Model Parameters

Following Liang et al. (2006), we collect counts of model parameters only from the agreement term.[2]

Given the agreed Viterbi phrase alignment $\hat{\mathbf{m}}$, the count of the source-to-target length model $p(J|I)$ is given by

$$c(J|I; E, F) = \sum_{\langle s,t \rangle \in \hat{\mathbf{m}}} \delta(J^{(t)}, J) \delta(I^{(s)}, I). \tag{24}$$

The new length probabilities can be obtained by

$$p(J|I) = \frac{c(J|I; E, F)}{\sum_{J'} c(J'|I; E, F)}. \tag{25}$$

[2]We experimented with collecting counts from both the unidirectional and agreement terms but obtained much worse results than counting only from the agreement term.

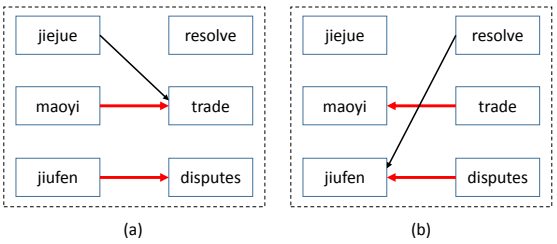


Figure 3: Agreement between (a) Chinese-to-English and (b) English-to-Chinese word alignments. The links on which two models agree are highlighted in red. The *inner agreement* loss function (see Eq. (28)) aims to encourage the agreement at both the phrase and word levels.

The count of the source-to-target translation model $p(f|e)$ is given by

$$\begin{aligned} & c(f|e; E, F) \\ = & \sum_{\langle s,t \rangle \in \hat{\mathbf{m}}} \frac{p(f|e)}{\sum_{i=0}^{I^{(s)}} p(f|\mathbf{e}_i^{(s)})} \times \\ & \sum_{j=1}^{J^{(t)}} \delta(f, \mathbf{f}_j^{(t)}) \sum_{i=0}^{I^{(s)}} \delta(e, \mathbf{e}_i^{(s)}) \\ & + \sigma(f, e, \mathbf{d}). \end{aligned} \tag{26}$$

The new translation probabilities can be obtained by

$$p(f|e) = \frac{c(f|e; E, F)}{\sum_{f'} c(f'|e; E, F)}. \tag{27}$$

Counts of target-to-source length and translation models can be calculated in a similar way.

3.3 Inner Agreement

3.3.1 Definition

As the outer agreement only considers the phrase alignment, the *inner agreement* takes both phrase alignment and word alignment into consideration:

$$\begin{aligned} & \Delta_{\text{inner}}(E, F, \overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}, \overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}) \\ = & -\delta(\overrightarrow{\mathbf{m}}, \overleftarrow{\mathbf{m}}) \times \\ & \sum_{\langle s,t \rangle \in \overrightarrow{\mathbf{m}}} \sum_{\overrightarrow{\mathbf{a}}, \overleftarrow{\mathbf{a}}} P(\overrightarrow{\mathbf{a}} | \mathbf{e}^{(s)}, \mathbf{f}^{(t)}; \overrightarrow{\boldsymbol{\theta}}) \times \\ & P(\overleftarrow{\mathbf{a}} | \mathbf{f}^{(t)}, \mathbf{e}^{(s)}; \overleftarrow{\boldsymbol{\theta}}) \times \\ & \delta(\overrightarrow{\mathbf{a}}, \overleftarrow{\mathbf{a}}). \end{aligned} \tag{28}$$

For example, Figure 3 shows two examples of Chinese-to-English and English-to-Chinese word alignments. The shared links are highlighted in

red. Our intuition is that a source phrase and a target phrase are more likely to be translations of each other if the two translation models also agree on word alignment within aligned phrases.

3.3.2 Training Objective and Algorithm

The training objective for inner agreement is given by

$$\begin{aligned}
&\mathcal{J}_{\text{inner}}(\overrightarrow{\boldsymbol{\theta}}, \overleftarrow{\boldsymbol{\theta}}) \\
= \;& \log P(F|E; \overrightarrow{\boldsymbol{\theta}}) + \log P(E|F; \overleftarrow{\boldsymbol{\theta}}) + \\
& \log \sum_{\mathbf{m}} P(\mathbf{m}|E, F; \overrightarrow{\boldsymbol{\theta}}) P(\mathbf{m}|F, E; \overleftarrow{\boldsymbol{\theta}}) \times \\
& \sum_{\langle s,t\rangle \in \mathbf{m}} \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e}^{(s)}, \mathbf{f}^{(t)}; \overrightarrow{\boldsymbol{\theta}}) \times \\
& P(\mathbf{a}|\mathbf{f}^{(t)}, \mathbf{e}^{(s)}; \overleftarrow{\boldsymbol{\theta}}). \quad (29)
\end{aligned}$$

We still use the Viterbi EM algorithm as shown in Figure 2 for training the two models.

3.3.3 Computing Viterbi Phrase Alignments

The agreed Viterbi phrase alignment is defined as

$$\begin{aligned}
\hat{\mathbf{m}} = \operatorname*{argmax}_{\mathbf{m}} \Big\{ & P(\mathbf{m}|E, F; \overrightarrow{\boldsymbol{\theta}}) P(\mathbf{m}|F, E; \overleftarrow{\boldsymbol{\theta}}) \\
& \times \sum_{\langle s,t\rangle \in \mathbf{m}} \sum_{\mathbf{a}} P(\mathbf{a}|\mathbf{e}^{(s)}, \mathbf{f}^{(t)}; \overrightarrow{\boldsymbol{\theta}}) \\
& \times P(\mathbf{a}|\mathbf{f}^{(t)}, \mathbf{e}^{(s)}; \overleftarrow{\boldsymbol{\theta}}) \Big\}. \quad (30)
\end{aligned}$$

As computing $\hat{\mathbf{m}}$ is intractable, we still approximate it using the intersection of two unidirectional Viterbi phrase alignments (see Eq. (18)). The source-to-target Viterbi phrase alignment is calculated as

$$\begin{aligned}
\overrightarrow{\mathbf{m}}^* = \operatorname*{argmax}_{\overrightarrow{\mathbf{m}}} \Big\{ & P(\overrightarrow{\mathbf{m}}|E, F; \overrightarrow{\boldsymbol{\theta}}) \times \\
& \sum_{\langle s,t\rangle \in \overrightarrow{\mathbf{m}}} \sum_{j=1}^{J^{(t)}} \sum_{i=1}^{I^{(s)}} P(\langle i, j\rangle|\mathbf{e}^{(s)}, \mathbf{f}^{(t)}; \overrightarrow{\boldsymbol{\theta}}) \times \\
& P(\langle i, j\rangle|\mathbf{f}^{(t)}, \mathbf{e}^{(s)}; \overleftarrow{\boldsymbol{\theta}}) \Big\}, \quad (31)
\end{aligned}$$

where $P(\langle i, j\rangle|\mathbf{e}^{(s)}, \mathbf{f}^{(t)}; \overrightarrow{\boldsymbol{\theta}})$ is source-to-target link posterior probability of the link $\langle i, j\rangle$ being present (or absent) in the word alignment according to the source-to-target model, $P(\langle i, j\rangle|\mathbf{f}^{(t)}, \mathbf{e}^{(s)}; \overleftarrow{\boldsymbol{\theta}})$ is target-to-source link posterior probability. We follow Liang et al. (2006) to use the product of link posteriors to encourage the agreement at the level of word alignment.

We use a coarse-to-fine approach (Dong et al., 2015) to compute the Viterbi alignment: first retrieving a coarse set of candidate source phrases using translation probabilities and then selecting the candidate with the highest score according to Eq. (31). The target-to-source Viterbi phrase alignment can be calculated similarly.

3.3.4 Updating Model Parameters

Given the agreed Viterbi phrase alignment $\hat{\mathbf{m}}$, the count of the source-to-target length model $p(J|I)$ is still given by Eq. (24). The count of the translation model $p(f|e)$ is calculated as

$$\begin{aligned}
& c(f|e; E, F) \\
= \;& \sum_{\langle s,t\rangle \in \hat{\mathbf{m}}} \sum_{i=1}^{I^{(s)}} \sum_{j=1}^{J^{(t)}} P(\langle i, j\rangle|\mathbf{e}^{(s)}, \mathbf{f}^{(t)}; \overrightarrow{\boldsymbol{\theta}}) \times \\
& \qquad P(\langle i, j\rangle|\mathbf{f}^{(t)}, \mathbf{e}^{(s)}; \overleftarrow{\boldsymbol{\theta}}) \times \\
& \qquad \delta(f, \mathbf{f}^{(t)}) \delta(e, \mathbf{e}^{(s)}) \\
& + \sigma(f, e, \mathbf{d}). \quad (32)
\end{aligned}$$

Counts of target-to-source length and translation models can be calculated in a similar way.

4 Experiments

In this section, we evaluate our approach in two tasks: phrase alignment (Section 4.1) and machine translation (Section 4.2).

4.1 Alignment Evaluation

4.1.1 Evaluation Metrics

Given two monolingual corpora E and F, we suppose there exists a ground truth parallel corpus G and denote an extracted parallel corpus as D. The quality of an extracted parallel corpus can be measured by F1 $= 2|D \cap G|/(|D| + |G|)$.

4.1.2 Data Preparation

Although it is appealing to apply our approach to dealing with real-world non-parallel corpora, it is time-consuming and labor-intensive to manually construct a ground truth parallel corpus. Therefore, we follow Dong et al. (2015) to build synthetic E, F, and G to facilitate the evaluation.

We first extract a set of parallel phrases from a sentence-level parallel corpus using the state-of-the-art phrase-based translation system Moses (Koehn et al., 2007) and discard low-probability parallel phrases. Then, E and F can be constructed by corrupting the parallel phrase set by

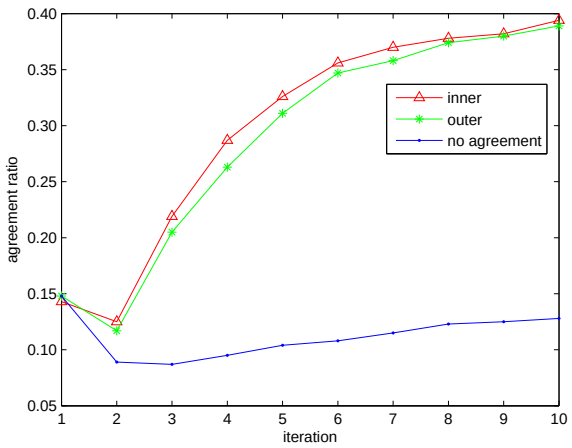


Figure 4: Comparison of agreement ratios on the development set.

seed	C → E	E → C	Outer	Inner
50	4.1	4.8	60.8	66.2
100	5.1	5.5	65.6	69.8
500	7.5	8.4	70.4	72.5
1,000	22.4	23.1	73.6	74.3

Table 1: Effect of seed lexicon size in terms of F1 on the development set.

adding irrelevant source and target phrases randomly. Note that the parallel phrase set can serve as the ground truth parallel corpus G. We refer to the non-parallel phrases in E and F as *noise*.

From LDC Chinese-English parallel corpora, we constructed a *development set* and a *test set*. The development set contains 20K parallel phrases, 20K noisy Chinese phrases, and 20K noisy English phrases. The test test contains 20K parallel phrases, 180K noisy Chinese phrases, and 180K noisy English phrases. The seed parallel lexicon contains 1K entries.

4.1.3 Comparison of Agreement Ratios

We introduce *agreement ratio* to measure to what extent two unidirectional models agree on phrase alignment:

$$\text{ratio} = \frac{2|\overrightarrow{\mathbf{m}}^* \cap \overleftarrow{\mathbf{m}}^*|}{|\overrightarrow{\mathbf{m}}^*| + |\overleftarrow{\mathbf{m}}^*|}. \quad (33)$$

Figure 4 shows the agreement ratios of independent training ("no agreement"), joint training with the outer agreement ("outer"), and joint training with the inner agreement ("inner"). We find that independently trained unidirectional models hardly agree on phrase alignment, suggesting that each model can only capture partial aspects of translation modeling on non-parallel corpora. In contrast, imposing the agreement term significantly increases the agreement ratios: after 10 iterations, about 40% of phrase alignment links are shared by two models.

noise		C → E	E → C	Outer	Inner
C	E				
0	0	58.5	61.2	86.5	86.1
0	10K	41.0	54.4	83.6	83.8
0	20K	28.3	48.3	80.1	81.2
10K	0	54.7	43.1	84.9	84.3
20K	0	50.4	31.4	83.8	83.6
10K	10K	34.9	34.4	80.0	79.7
20K	20K	22.4	23.1	73.6	74.3

Table 2: Effect of noise in terms of F1 on the development set.

4.1.4 Effect of Seed Lexicon Size

Table 1 shows the F1 scores of the Chinese-to-English model ("C → E"), the English-to-Chinese model ("E → C"), joint learning based on the outer agreement ("outer"), and jointing learning based on the inner agreement ("inner") over various sizes of seed lexicons on the development set.

We find that agreement-based learning obtains substantial improvements over independent learning across all sizes. More importantly, even with a seed lexicon containing only 50 entries, agreement-based learning is able to achieve F1 scores above 60%. The inner agreement performs better than the outer agreement by taking the consensus at the word level into account.

4.1.5 Effect of Noise

Table 2 demonstrates the effect of noise on the development set. In row 1, "0+0" denotes there is no noise, which can be seen as an upper bound. Adding noise, either on the Chinese side or on the English side, deteriorates the F1 scores for all methods. Adding noise on the English side makes predicting phrase alignment in the C → E direction more challenging due to the enlarged search space. The situation is similar in the reverse direction. It is clear that agreement-based learning is more robust to noise: while independent training suffers from a reduction of 40% in terms of F1 for the "20K + 20K" setting, agreement-based learning still achieves F1 scores over 70%.

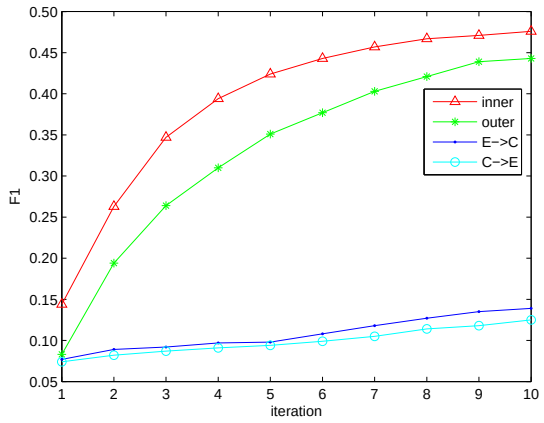


Figure 5: Comparison of F1 scores on the test set.

Chinese	jingji
English	*economy*
Chinese	*jialebi*
English	*caribbean*
Chinese	zhengzhi huanjing
English	political environment
Chinese	*jiaoyisuo* shichang *jiage* zhishu
English	*exchange* market *price* index
Chinese	*qianding bianjing maoyi xieding*
English	*signed border trade agreements*

Table 3: Example learned parallel lexicons and phrases. New words that are not included in the seed lexicon are highlighted in italic.

4.1.6 Results

Figure 5 gives the final results on the test set. We find that agreement-based training achieves significant improvements over independent training. By considering the consensus on both phrase and word alignments, the inner agreement significantly outperforms the outer agreement. Notice that Dong et al. (2015) only add noise on one side while we add noisy phrases on both sides, which makes phrase alignment more challenging.

Table 3 shows example learned parallel words and phrases. The lexicon is built from the translation table by retaining high-probability word pairs. Therefore, our approach is capable of learning both new words and new phrases unseen in the seed lexicon.

4.2 Translation Evaluation

Following Zhang and Zong (2013) and Dong et al. (2015), we evaluate our approach on domain adaptation for machine translation.

The data set consists of two in-domain non-parallel corpora and an out-domain parallel corpus. The in-domain non-parallel corpora consists of 2.65M Chinese phrases and 3.67M English phrases extracted from LDC news articles. We use a small out-domain parallel corpus extracted from financial news of FTChina which contains 10K phrase pairs. The task is to extract a parallel corpus from in-domain non-parallel corpora starting from a small out-domain parallel corpus.

We use the state-of-the-art translation system Moses (Koehn et al., 2007) and evaluate the performance on Chinese-English NIST datasets. The development set is NIST 2006 and the test set is NIST 2005. The evaluation metric is case-insensitive BLEU4 (Papineni et al., 2002). We use the SRILM toolkit (Stolcke, 2002) to train a 4-gram English language model on a monolingual corpus with 399M English words.

Table 4 shows the results. At iteration 0, only the out-domain corpus is used and the BLEU score is 5.61. All methods iteratively extract parallel phrases from non-parallel corpora and enlarge the extracted parallel corpus. We find that agreement-based learning achieves much higher BLEU scores while obtains a smaller parallel corpus as compared with independent learning. One possible reason is that the agreement-based learning rules out most unlikely phrase pairs by encouraging consensus between two models.

5 Conclusion

We have presented agreement-based training for learning parallel lexicons and phrases from non-parallel corpora. By modeling the agreement on both phrase alignment and word alignment, our approach achieves significant improvements in both alignment and translation evaluations.

In the future, we plan to apply our approach to real-world non-parallel corpora to further verify its effectiveness. It is also interesting to extend the phrase translation model to more sophisticated models such as IBM models 2-5 (Brown et al., 1993) and HMM (Vogel and Ney, 1996).

Acknowledgments

We sincerely thank the reviewers for their valuable suggestions. We also thank Meng Zhang, Yankai Lin, Shiqi Shen, Meiping Dong and Congyu Fu for their insightful discussions. Yang Liu is sup-

Iteration	Corpus Size				BLEU			
	E→ C	C→ E	Outer	Inner	E→ C	C→ E	Outer	Inner
0	10k				5.61			
1	145k	162k	59k	73k	8.65	8.90	13.53	13.74
2	195k	215k	69k	101k	8.82	9.47	15.26	15.61
3	209k	231k	88k	132k	8.42	9.29	16.88	16.94
4	214k	238k	106k	159k	8.46	9.27	17.15	17.83
5	217k	241k	123k	181k	8.87	9.40	17.94	18.89
6	219k	243k	137k	197k	8.52	9.30	18.56	19.47
7	222k	247k	140k	207k	8.81	9.22	18.72	19.46
8	224k	249k	153k	220k	8.71	9.26	18.84	19.50
9	227k	251k	159k	233k	8.92	9.35	19.05	19.63
10	229k	254k	163k	239k	8.33	9.06	19.39	19.78

Table 4: Results on domain adaptation for machine translation.

ported by the National Natural Science Foundation of China (No. 61522204), the 863 Program (2015AA011808), and Samsung R&D Institute of China. Huanbo Luan is supported by the National Natural Science Foundation of China (No. 61303075). Maosong Sun is supported by the Major Project of the National Social Science Foundation of China (13&ZD190).

References

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguisitics*.

Mauro Cettolo, Marcello Federico, and Nicola Bertoldi. 2010. Mining parallel fragments from comparable texts. In *Proceedings of IWSLT*.

Meiping Dong, Yong Cheng, Yang Liu, Jia Xu, Maosong Sun, Tatsuya Izuha, and Jie Hao. 2014. Query lattice for translation retrieval. In *Proceedings of COLING*.

Meiping Dong, Yang Liu, Huanbo Luan, Maosong Sun, Tatsuya Izuha, and Dakun Zhang. 2015. Iterative learning of parallel lexicons and phrases from non-parallel corpora. In *Proceedings of IJCAI*.

Qing Dou, Ashish Vaswani, and Kevin Knight. 2014. Beyond parallel data: Joint word alignment and decipherment improves machine translation. In *Proceedings of EMNLP*.

Pascale Fung and Percy Cheung. 2004. Mining very-non-parallel corpora: Parallel sentence and lexicon extraction via bootstrapping and em. In *Proceedings of EMNLP*.

Eric Gaussier, J.M. Renders, I. Matveeva, C. Goutte, and H. Dejean. 2004. A geometric view on bilingual lexicon extraction from comparable corpora. In *Proceedings of ACL*.

Aria Haghighi, Percy Liang, Taylor Berg-Kirkpatrick, and Dan Klein. 2008. Learning bilingual lexicons from monolingual corpora. In *Proceedings of ACL*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*.

Julian Kupiec. 1993. An algorithm for finding noun phrase correspondences in bilingual corpora. In *Proceedings of ACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *Proceedings of NAACL*.

Percy Liang, Dan Klein, and I. Jordan, Michael. 2008. Alignment-based learning. In *Proceedings of NIPS*.

Daniel Marcu and Daniel Wong. 2002. A phrase-based joint probability model for statistical machine translation. In *Proceedings of EMNLP*.

I. Dan Melamed. 1997. Automatic discovery of non-compositional compounds in parallel data. In *Proceedings of EMNLP*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013. Exploiting similarities among languages for machine translation. arXiv:1309.4168.

Dragos Stefan Munteanu and Daniel Marcu. 2006. Extracting parallel sub-sentential fragments from non-parallel corpora. In *Proceedings of ACL*.

Malte Nuhn, Arne Mauser, and Hermann Ney. 2012. Deciphering foreign language by combining language models and context vectors. In *Proceedings of ACL*.

Franz J. Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguisitics*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a methof for automatic evaluation of machine translation. In *Proceedings of ACL*.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of ACL*.

Frank Smadja and Kathleen McKeown. 1994. Translating collocations for use in bilingual lexicons. In *Proceedings of the ARPA Human Language Technology Workshop*.

Andreas Stolcke. 2002. Srilm - an extensible language modeling toolkit. In *Proceedings of ICSLP*.

Stephan Vogel and Hermann Ney. 1996. Hhm-based word alignment in statistical translation. In *Proceedings of COLING*.

Ivan Vulić and Marie-Francine Moens. 2013. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proceedings of EMNLP*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proceedings of ACL*.

Dekai Wu and Xuanyin Xia. 1994. Learning an english-chinese lexicon from a parallel corpus. In *Proceedings of the ARPA Human Language Technology Workshop*.

Jiajun Zhang and Chengqing Zong. 2013. Learning a phrase-based translation model from monolingual data with application to domain adaptation. In *Proceedings of ACL*.

Deep Fusion LSTMs for Text Semantic Matching

Pengfei Liu, Xipeng Qiu*, Jifan Chen, Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pfliu14,xpqiu,jfchen14,xjhuang}@fudan.edu.cn

Abstract

Recently, there is rising interest in modelling the interactions of text pair with deep neural networks. In this paper, we propose a model of deep fusion LSTMs (DF-LSTMs) to model the strong interaction of text pair in a recursive matching way. Specifically, DF-LSTMs consist of two interdependent LSTMs, each of which models a sequence under the influence of another. We also use external memory to increase the capacity of LSTMs, thereby possibly capturing more complicated matching patterns. Experiments on two very large datasets demonstrate the efficacy of our proposed architecture. Furthermore, we present an elaborate qualitative analysis of our models, giving an intuitive understanding how our model worked.

1 Introduction

Among many natural language processing (NLP) tasks, such as text classification, question answering and machine translation, a common problem is modelling the relevance/similarity of a pair of texts, which is also called text semantic matching. Due to the semantic gap problem, text semantic matching is still a challenging problem.

Recently, deep learning is rising a substantial interest in text semantic matching and has achieved some great progresses (Hu et al., 2014; Qiu and Huang, 2015; Wan et al., 2016). According to their interaction ways, previous models can be classified into three categories:

Weak interaction Models Some early works focus on sentence level interactions, such as ARC-I(Hu et al., 2014), CNTN(Qiu and Huang, 2015) and so on. These models first encode two sequences into continuous dense vectors by separated neural models, and then compute the matching score based on sentence encoding. In this paradigm, two sentences have no interaction until arriving final phase.

Semi-interaction Models Another kind of models use soft attention mechanism to obtain the representation of one sentence by depending on representation of another sentence, such as ABCNN (Yin et al., 2015), Attention LSTM (Rocktäschel et al., 2015; Hermann et al., 2015). These models can alleviate the weak interaction problem to some extent.

Strong Interaction Models Some models build the interaction at different granularity (word, phrase and sentence level), such as ARC-II (Hu et al., 2014), MultiGranCNN (Yin and Schütze, 2015), Multi-Perspective CNN (He et al., 2015), MV-LSTM (Wan et al., 2016), MatchPyramid (Pang et al., 2016). The final matching score depends on these different levels of interactions.

In this paper, we adopt a deep fusion strategy to model the strong interactions of two sentences. Given two texts $x_{1:m}$ and $y_{1:n}$, we define a matching vector $\mathbf{h}_{i,j}$ to represent the interaction of the subsequences $x_{1:i}$ and $y_{1:j}$. $\mathbf{h}_{i,j}$ depends on the matching vectors $\mathbf{h}_{s,t}$ on previous interactions $1 \leq s < i$ and $1 \leq t < j$. Thus, text matching can be regarded as modelling the interaction of two texts in a recursive matching way.

Following this idea, we propose deep fusion long short-term memory neural networks (DF-LSTMs) to model the interactions recursively. More concretely, DF-LSTMs consist of two interconnected conditional LSTMs, each of which models a piece of text under the influence of another. The output vector of DF-LSTMs is fed into a task-specific output layer to compute the match-

*Corresponding author

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1034–1043,
Berlin, Germany, August 7-12, 2016.

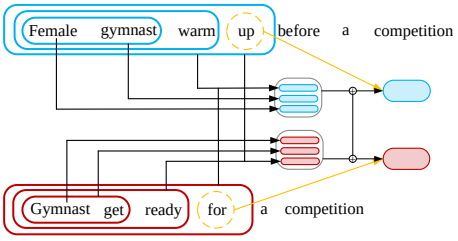


Figure 1: A motivated example to illustrate our recursive composition mechanism.

ing score.

The contributions of this paper can be summarized as follows.

1. Different with previous models, DF-LSTMs model the strong interactions of two texts in a recursive matching way, which consist of two inter- and intra-dependent LSTMs.

2. Compared to the previous works on text matching, we perform extensive empirical studies on two very large datasets. Experiment results demonstrate that our proposed architecture is more effective.

3. We present an elaborate qualitative analysis of our model, giving an intuitive understanding how our model worked.

2 Recursively Text Semantic Matching

To facilitate our model, we firstly give some definitions.

Given two sequences $X = x_1, x_2, \cdots, x_m$ and $Y = y_1, y_2, \cdots, y_n$, most deep neural models try to represent their semantic relevance by a **matching vector** $\mathbf{h}(X, Y)$, which is followed by a score function to calculate the matching score.

The weak interaction methods decompose matching vector by $\mathbf{h}(X, Y) = f(\mathbf{h}(X), \mathbf{h}(Y))$, where function $f(\cdot)$ may be one of some basic operations or the combination of them: concatenation, affine transformation, bilinear, and so on.

In this paper, we propose a strong interaction of two sequences to decompose matching vector $\mathbf{h}(X, Y)$ in a recursive way. We refer to the interaction of the subsequences $x_{1:i}$ and $y_{1:j}$ as $\mathbf{h}_{i,j}(X, Y)$, which depends on previous interactions $\mathbf{h}_{s,t}(X, Y)$ for $1 \leq s < i$ and $1 \leq t < j$.

Figure 1 gives an example to illustrate this. For sentence pair X ="`Female gymnast warm up before a competition`", Y="`Gymnast get ready for a competition`", considering the interaction ($\mathbf{h}_{4,4}$) between $x_{1:4}$ = "`Female gymnast warm up`" and $y_{1:4}$ = "`Gymnast get ready for`", which is composed by the interactions between their subsequences $(\mathbf{h}_{1,4}, \cdots, \mathbf{h}_{3,4}, \mathbf{h}_{4,1}, \cdots, \mathbf{h}_{4,3})$. We can see that a strong interaction between two sequences can be decomposed in recursive topology structure.

The matching vector $\mathbf{h}_{i,j}(X, Y)$ can be written as

$$\mathbf{h}_{i,j}(X, Y) = \mathbf{h}_{i,j}(X|Y) \oplus \mathbf{h}_{i,j}(Y|X), \quad (1)$$

where $\mathbf{h}_{i,j}(X|Y)$ refers to conditional encoding of subsequence $x_{1:i}$ influenced by $y_{1:j}$. Meanwhile, $\mathbf{h}_{i,j}(Y|X)$ is conditional encoding of subsequence $y_{1:j}$ influenced by subsequence $x_{1:i}$; $\oplus$ is concatenation operation.

These two conditional encodings depend on their history encodings. Based on this, we propose deep fusion LSTMs to model the matching of texts by recursive composition mechanism, which can better capture the complicated interaction of two sentences due to fully considering the interactions between subsequences.

3 Long Short-Term Memory Network

Long short-term memory neural network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) (Elman, 1990), and specifically addresses the issue of learning long-term dependencies. LSTM maintains a memory cell that updates and exposes its content only when deemed necessary.

While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz et al., 2015), which is similar to the architecture of (Graves, 2013) but without peep-hole connections.

We define the LSTM *units* at each time step t to be a collection of vectors in $\mathbb{R}^d$: an *input gate* $\mathbf{i}_t$, a *forget gate* $\mathbf{f}_t$, an *output gate* $\mathbf{o}_t$, a *memory cell* $\mathbf{c}_t$ and a hidden state $\mathbf{h}_t$. d is the number of the LSTM units. The elements of the gating vectors $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ are in $[0, 1]$.

The LSTM is precisely specified as follows.

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} T_{\mathbf{A},\mathbf{b}} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}, \quad (2)$$

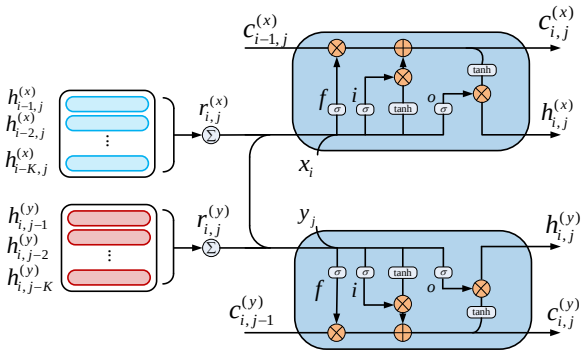


Figure 2: Illustration of DF-LSTMs unit.

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (3)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (4)$$

where $\mathbf{x}_t$ is the input at the current time step; $T_{\mathbf{A},\mathbf{b}}$ is an affine transformation which depends on parameters of the network $\mathbf{A}$ and $\mathbf{b}$. σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication. Intuitively, the forget gate controls the amount of which each unit of the memory cell is erased, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state.

The update of each LSTM unit can be written precisely as

$$(\mathbf{h}_t, \mathbf{c}_t) = \mathbf{LSTM}(\mathbf{h}_{t-1}, \mathbf{c}_{t-1}, \mathbf{x}_t). \quad (5)$$

Here, the function $\mathbf{LSTM}(\cdot,\cdot,\cdot)$ is a shorthand for Eq. (2-4).

LSTM can map the input sequence of arbitrary length to a fixed-sized vector, and has been successfully applied to a wide range of NLP tasks, such as machine translation (Sutskever et al., 2014), language modelling (Sutskever et al., 2011), text matching (Rocktäschel et al., 2015) and text classification (Liu et al., 2015).

4 Deep Fusion LSTMs for Recursively Semantic Matching

To deal with two sentences, one straightforward method is to model them with two separate LSTMs. However, this method is difficult to model local interactions of two sentences.

Following the recursive matching strategy, we propose a neural model of deep fusion LSTMs (DF-LSTMs), which consists of two interdependent LSTMs to capture the inter- and intra-interactions between two sequences. Figure 2 gives an illustration of DF-LSTMs unit.

To facilitate our model, we firstly give some definitions. Given two sequences $X = x_1, x_2, \cdots, x_n$ and $Y = y_1, y_2, \cdots, y_m$, we let $\mathbf{x}_i \in \mathbb{R}^d$ denotes the embedded representation of the word x_i. The standard LSTM has one temporal dimension. When dealing with a sentence, LSTM regards the position as time step. At position i of sentence $x_{1:n}$, the output $\mathbf{h}_i$ reflects the meaning of subsequence $x_{1:i} = x_1, \cdots, x_i$.

To model the interaction of two sentences in a recursive way, we define $\mathbf{h}_{i,j}$ to represent the interaction of the subsequences $x_{1:i}$ and $y_{1:j}$, which is computed by

$$\mathbf{h}_{i,j} = \mathbf{h}^{(x)}_{i,j} \oplus \mathbf{h}^{(y)}_{i,j}, \quad (6)$$

where $\mathbf{h}^{(x)}_{i,j}$ denotes the encoding of subsequence $x_{1:i}$ in the first LSTM influenced by the output of the second LSTM on subsequence $y_{1:j}$; $\mathbf{h}^{(y)}_{i,j}$ is the encoding of subsequence $y_{1:j}$ in the second LSTM influenced by the output of the first LSTM on subsequence $x_{1:i}$.

More concretely,

$$(\mathbf{h}^{(x)}_{i,j}, \mathbf{c}^{(x)}_{i,j}) = \mathbf{LSTM}(\mathcal{H}_{i,j}, \mathbf{c}^{(x)}_{i-1,j}, \mathbf{x}_i), \quad (7)$$

$$(\mathbf{h}^{(y)}_{i,j}, \mathbf{c}^{(y)}_{i,j}) = \mathbf{LSTM}(\mathcal{H}_{i,j}, \mathbf{c}^{(y)}_{i,j-1}, \mathbf{x}_j), \quad (8)$$

where $\mathcal{H}_{i,j}$ is information consisting of history states before position (i, j).

The simplest setting is $\mathcal{H}_{i,j} = \mathbf{h}^{(x)}_{i-1,j} \oplus \mathbf{h}^{(y)}_{i,j-1}$. In this case, our model can be regarded as grid LSTMs (Kalchbrenner et al., 2015).

However, there are total $m \times n$ interactions in recursive matching process, LSTM could be stressed to keep these interactions in internal memory. Therefore, inspired by recent neural memory network, such as neural Turing machine(Graves et al., 2014) and memory network (Sukhbaatar et al., 2015), we introduce two external memories to keep the history information, which can relieve the pressure on low-capacity internal memory.

Following (Tran et al., 2016), we use external memory constructed by history hidden states, which is defined as

$$\mathbf{M}_t = \{\mathbf{h}_{t-K}, \ldots, \mathbf{h}_{t-1}\} \in \mathbb{R}^{K \times d}, \quad (9)$$

where K is the number of memory segments, which is generally instance-independent and predefined as hyper-parameter; d is the size of each segment; and $\mathbf{h}_t$ is the hidden state at time t emitted by LSTM.

At position i, j, two memory blocks $\mathbf{M}^{(x)}, \mathbf{M}^{(y)}$ are used to store contextual information of x and y respectively.

$$\mathbf{M}_{i,j}^{(x)} = \{\mathbf{h}_{i-K,j}^{(x)}, \ldots, \mathbf{h}_{i-1,j}^{(x)}\}, \quad (10)$$

$$\mathbf{M}_{i,j}^{(y)} = \{\mathbf{h}_{i,j-K}^{(y)}, \ldots, \mathbf{h}_{i,j-1}^{(y)}\}, \quad (11)$$

where $\mathbf{h}^{(x)}$ and $\mathbf{h}^{(x)}$ are outputs of two conditional LSTMs at different positions.

The history information can be read from these two memory blocks. We denote a read vector from external memories as $\mathbf{r}_{i,j} \in \mathbb{R}^d$, which can be computed by soft attention mechanisms.

$$\mathbf{r}_{i,j}^{(x)} = \mathbf{a}_{i,j}^{(x)} \mathbf{M}_{i,j}^{(x)}, \quad (12)$$

$$\mathbf{r}_{i,j}^{(y)} = \mathbf{a}_{i,j}^{(y)} \mathbf{M}_{i,j}^{(y)}, \quad (13)$$

where $\mathbf{a}_{i,j} \in \mathbb{R}^K$ represents attention distribution over the corresponding memory $\mathbf{M}_{i,j} \in \mathbb{R}^{K \times d}$.

More concretely, each scalar $a_{i,j,k}$ in attention distribution $\mathbf{a}_{i,j}$ can be obtained:

$$a_{i,j,k}^{(x)} = \text{softmax}(g(\mathbf{M}_{i,j,k}^{(x)}, \mathbf{r}_{i-1,j}^{(x)}, \mathbf{x}_i)), \quad (14)$$

$$a_{i,j,k}^{(y)} = \text{softmax}(g(\mathbf{M}_{i,j,k}^{(y)}, \mathbf{r}_{i,j-1}^{(y)}, \mathbf{y}_j)), \quad (15)$$

where $\mathbf{M}_{i,j,k} \in \mathbb{R}^d$ represents the k-th row memory vector at position (i, j), and $g(\cdot)$ is an align function defined by

$$g(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{v}^T \tanh(\mathbf{W}_a[\mathbf{x}; \mathbf{y}, \mathbf{z}]), \quad (16)$$

where $\mathbf{v} \in \mathbb{R}^d$ is a parameter vector and $\mathbf{W}_a \in \mathbb{R}^{d \times 3d}$ is a parameter matrix.

The history information $\mathcal{H}_{i,j}$ in Eq (7) and (8) is computed by

$$\mathcal{H}_{i,j} = \mathbf{r}_{i,j}^{(x)} \oplus \mathbf{r}_{i,j}^{(y)}. \quad (17)$$

By incorporating external memory blocks, DF-LSTMs allow network to re-read history interaction information, therefore it can more easily capture complicated and long-distance matching patterns. As shown in Figure 3, the forward pass of DF-LSTMs can be unfolded along two dimensional ordering.

4.1 Related Models

Our model is inspired by some recently proposed models based on recurrent neural network (RNN).

One kind of models is multi-dimensional recurrent neural network (MD-RNN) (Graves et al., 2007; Graves and Schmidhuber, 2009; Byeon et al., 2015) in machine learning and computer vision communities. As mentioned above, if we just use the neighbor states, our model can be regarded as grid LSTMs (Kalchbrenner et al., 2015).

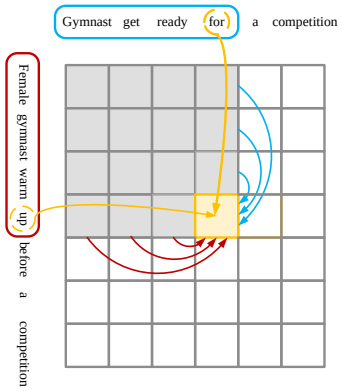


Figure 3: Illustration of unfolded DF-LSTMs.

What is different is the dependency relations between the current state and history states. Our model uses external memory to increase its memory capacity and therefore can store large useful interactions of subsequences. Thus, we can discover some matching patterns with long dependence.

Another kind of models is memory augmented RNN, such as long short-term memory-network (Cheng et al., 2016) and recurrent memory network (Tran et al., 2016), which extend memory network (Bahdanau et al., 2014) and equip the RNN with ability of re-reading the history information. While they focus on sequence modelling, our model concentrates more on modelling the interactions of sequence pair.

5 Training

5.1 Task Specific Output

There are two popular types of text matching tasks in NLP. One is ranking task, such as community question answering. Another is classification task, such as textual entailment.

We use different ways to calculate matching score for these two types of tasks.

1. For ranking task, the output is a scalar matching score, which is obtained by a linear transformation of the matching vector obtained by FD-LSTMs.

2. For classification task, the outputs are the probabilities of the different classes, which

is computed by a softmax function on the matching vector obtained by FD-LSTMs.

5.2 Loss Function

Accordingly, we use two loss functions to deal with different sentence matching tasks.

Max-Margin Loss for Ranking Task Given a positive sentence pair (X, Y) and its corresponding negative pair $(X, \hat{Y})$. The matching score $s(X, Y)$ should be larger than $s(X, \hat{Y})$.

For this task, we use the contrastive max-margin criterion (Bordes et al., 2013; Socher et al., 2013) to train our model on matching task.

The ranking-based loss is defined as

$$L(X, Y, \hat{Y}) = max(0, 1 - s(X, Y) + s(X, \hat{Y})). \quad (18)$$

where $s(X, Y)$ is predicted matching score for (X, Y).

Cross-entropy Loss for Classification Task Given a sentence pair (X, Y) and its label l. The output $\hat{l}$ of neural network is the probabilities of the different classes. The parameters of the network are trained to minimise the cross-entropy of the predicted and true label distributions.

$$L(X, Y; \boldsymbol{l}, \hat{\boldsymbol{l}}) = -\sum_{j=1}^{C} \boldsymbol{l}_j \log(\hat{\boldsymbol{l}}_j), \quad (19)$$

where $\boldsymbol{l}$ is one-hot representation of the ground-truth label l; $\hat{\boldsymbol{l}}$ is predicted probabilities of labels; C is the class number.

5.3 Optimizer

To minimize the objective, we use stochastic gradient descent with the diagonal variant of AdaGrad (Duchi et al., 2011).

To prevent exploding gradients, we perform gradient clipping by scaling the gradient when the norm exceeds a threshold (Graves, 2013).

5.4 Initialization and Hyperparameters

Orthogonal Initialization We use orthogonal initialization of our LSTMs, which allows neurons to react to the diverse patterns and is helpful to train a multi-layer network (Saxe et al., 2013).

Unsupervised Initialization The word embeddings for all of the models are initialized with the 100d GloVe vectors (840B token version, (Pennington et al., 2014)). The other parameters are initialized by randomly sampling from uniform distribution in $[-0.1, 0.1]$.

Hyper-parameters	MQA	RTE
K	9	9
Embedding size	100	100
Hidden layer size	50	100
Initial learning rate	0.05	0.005
Regularization	$5E{-}5$	$1E{-}5$

Table 1: Hyper-parameters for our model on two tasks.

Hyperparameters For each task, we used a stacked DF-LSTM and take the hyperparameters which achieve the best performance on the development set via an small grid search over combinations of the initial learning rate $[0.05, 0.0005, 0.0001]$, l_2 regularization $[0.0, 5E{-}5, 1E{-}5, 1E{-}6]$ and the values of K $[1, 3, 6, 9, 12]$. The final hyper-parameters are set as Table 1.

6 Experiment

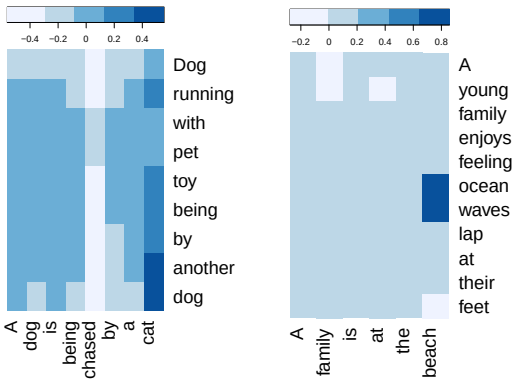
In this section, we investigate the empirical performances of our proposed model on two different text matching tasks: classification task (recognizing textual entailment) and ranking task (matching of question and answer).

6.1 Competitor Methods

- Neural bag-of-words (NBOW): Each sequence is represented as the sum of the embeddings of the words it contains, then they are concatenated and fed to a MLP.
- Single LSTM: Two sequences are encoded by a single LSTM, proposed by (Rocktäschel et al., 2015).
- Parallel LSTMs: Two sequences are first encoded by two LSTMs separately, then they are concatenated and fed to a MLP.
- Attention LSTMs: Two sequences are encoded by LSTMs with attention mechanism, proposed by (Rocktäschel et al., 2015).
- Word-by-word Attention LSTMs: An improved strategy of attention LSTMs, which introduces word-by-word attention mechanism and is proposed by (Rocktäschel et al., 2015).

Model	k	Train	Test
NBOW	100	77.9	75.1
single LSTM (Rocktäschel et al., 2015)	100	83.7	80.9
parallel LSTMs (Bowman et al., 2015)	100	84.8	77.6
Attention LSTM (Rocktäschel et al., 2015)	100	83.2	82.3
Attention(w-by-w) LSTM (Rocktäschel et al., 2015)	100	83.7	83.5
DF-LSTMs	100	85.2	**84.6**

Table 2: Accuracies of our proposed model against other neural models on SNLI corpus.



(a) 5-th neuron (b) 11-th neuron

Figure 4: Illustration of two interpretable neurons and some word-pairs captured by these neurons. The darker patches denote the corresponding activations are higher.

6.2 Experiment-I: Recognizing Textual Entailment

Recognizing textual entailment (RTE) is a task to determine the semantic relationship between two sentences. We use the Stanford Natural Language Inference Corpus (SNLI) (Bowman et al., 2015). This corpus contains 570K sentence pairs, and all of the sentences and labels stem from human annotators. SNLI is two orders of magnitude larger than all other existing RTE corpora. Therefore, the massive scale of SNLI allows us to train powerful neural networks such as our proposed architecture in this paper.

6.2.1 Results

Table 2 shows the evaluation results on SNLI. The 2nd column of the table gives the number of hidden states.

From experimental results, we have several experimental findings.

The results of DF-LSTMs outperform all the competitor models with the same number of hidden states while achieving comparable results to the state-of-the-art and using much fewer parameters, which indicate that it is effective to model the strong interactions of two texts in a recursive matching way.

All models outperform NBOW by a large margin, which indicate the importance of words order in semantic matching.

The strong interaction models surpass the weak interaction models, for example, compared with parallel LSTMs, DF-LSTMs obtain improvement by 7.0%.

6.2.2 Understanding Behaviors of Neurons in DF-LSTMs

To get an intuitive understanding of how the DF-LSTMs work on this problem, we examined the neuron activations in the last aggregation layer while evaluating the test set. We find that some cells are bound to certain roles.

We refer to $h_{i,j,k}$ as the activation of the k-th neuron at the position of (i, j), where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. By visualizing the hidden state $\mathbf{h}_{i,j,k}$ and analyzing the maximum activation, we can find that there exist multiple interpretable neurons. For example, when some contextualized local perspectives are semantically related at point (i, j) of the sentence pair, the activation value of hidden neuron $h_{i,j,k}$ tends to be maximum, meaning that the model could capture some reasoning patterns.

Figure 4 illustrates this phenomenon. In Figure 4(a), a neuron shows its ability to monitor the word pairs with the property of describing different things of the same type.

The activation in the patch, containing the word pair "`(cat, dog)`", is much higher than others. This is an informative pattern for the relation prediction of these two sentences, whose ground truth is contradiction. An interesting thing is there are two "`dog`" in sentence "`Dog running with pet toy being by another dog`". Our model ignores the useless word, which indicates this neuron selectively captures pattern by contextual understanding, not just word level interaction.

In Figure 4(b), another neuron shows that it can capture the local contextual interactions, such as "`(ocean waves, beach)`". These patterns can be easily captured by final layer and provide a strong support for the final prediction.

Index of Cell	Word or Phrase Pairs	Explanation
5-th	(jeans, shirt), (dog, cat) (retriever, cat), (stand, sitting)	different entities or events of the same type
11-th	(pool, swimming), (street, outside) (animal, dog), (grass,outside)	word pair related to lexical entailment
20-th	(skateboard, skateboarding), (running, runs) (advertisement, ad), (grassy, grass)	words with different morphology
49-th	(blue, blue), (wearing black, wearing white), (green uniform, red uniform)	words related to color
55-th	(a man, two other men), (a man, two girls) (Two women, No one)	subjects with singular or plural forms

Table 3: Multiple interpretable neurons and the word-pairs/phrase-pairs captured by these neurons. The third column gives the explanations of corresponding neuron's behaviours.

Table 3 illustrates multiple interpretable neurons and some representative word or phrase pairs which can activate these neurons. These cases show that our model can capture contextual interactions beyond word level.

6.2.3 Case Study for Attention Addressing Mechanism

External memory with attention addressing mechanism enables the network explicitly to utilize the history information of two sentences simultaneously. As a by-product, the obtained attention distribution over history hidden states also help us interpret the network and discover underlying dependencies present in the data.

To this end, we randomly sample two good cases with entailment relation from test data and visualize attention distributions over external memory constructed by last 9 hidden states. As shown in Figure 5(a), For the first sentence pair, when the word pair "`(competition, competition)`" are processed, the model simultaneously selects "`warm, before`" from one sentence and "`gymnast,ready,for`" from the other, which are informative patterns and indicate our model has the capacity of capturing phrase-phrase pair.

Another case in Figure 5(b) also shows by attention mechanism, the network can sufficiently utilize the history information and the fusion approach allows two LSTMs to share the history information of each other.

6.2.4 Error Analysis

Although our model DF-LSTMs are more sensitive to the discrepancy of the semantic capacity between two sentences, some cases still can not be solved by our model. For example, our model gives a wrong prediction of the sentence pair "`A golden retriever nurses puppies/Puppies next to their mother`", whose ground truth is entailment. The model fails to realize "`nurses`" means "`next to`".

Besides, despite the large size of the training corpus, it's still very difficult to solve some cases, which depend on the combination of the world knowledge and context-sensitive inferences. For example, given an entailment pair "`Several women are playing volleyball/The women are hitting a ball with their arms`", all models predict "`neutral`".

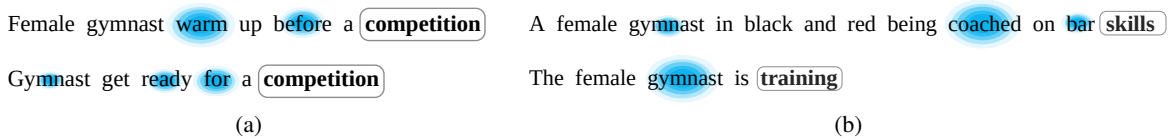
These analysis suggests that some architectural improvements or external world knowledge are necessary to eliminate all errors instead of simply scaling up the basic model.

6.3 Experiment-II: Matching Question and Answer

Matching question answering (MQA) is a typical task for semantic matching (Zhou et al., 2013). Given a question, we need select a correct answer from some candidate answers.

In this paper, we use the dataset collected from Yahoo! Answers with the getByCategory function provided in Yahoo! Answers API, which produces $963,072$ questions and corresponding best answers. We then select the pairs in which the length of questions and answers are both in the interval $[4, 30]$, thus obtaining $220,000$ question answer pairs to form the positive pairs.

For negative pairs, we first use each question's best answer as a query to retrieval top $1,000$ re-



(a) (b)

Figure 5: Examples of external memory positions attended when encoding the next word pair (bold and marked by a box)

Model	k	P@1(5)	P@1(10)
Random Guess	-	20.0	10.0
NBOW	50	63.9	47.6
single LSTM	50	68.2	53.9
parallel LSTMs	50	66.9	52.1
Attention LSTMs	50	73.5	62.0
Attention(w-by-w) LSTMs	50	75.1	64.0
DF-LSTMs	50	**76.5**	**65.0**

Table 4: Results of our proposed model against other neural models on Yahoo! question-answer pairs dataset.

sults from the whole answer set with Lucene, where 4 or 9 answers will be selected randomly to construct the negative pairs.

The whole dataset[1] is divided into training, validation and testing data with proportion $20:1:1$. Moreover, we give two test settings: selecting the best answer from 5 and 10 candidates respectively.

6.3.1 Results

Results of MQA are shown in the Table 4. we can see that the proposed model also shows its superiority on this task, which outperforms the state-of-the-arts methods on both metrics (P@1(5) and P@1(10)) with a large margin.

By analyzing the evaluation results of question-answer matching in Table 4, we can see strong interaction models (attention LSTMs, our DF-LSTMs) consistently outperform the weak interaction models (NBOW, parallel LSTMs) with a large margin, which suggests the importance of modelling strong interaction of two sentences.

7 Related Work

Our model can be regarded as a strong interaction model, which has been explored in previous methods.

One kind of methods is to compute similarities between all the words or phrases of the two sentences to model multiple-granularity interactions of two sentences, such as RAE (Socher et al., 2011), Arc-II (Hu et al., 2014),ABCNN (Yin et al., 2015),MultiGranCNN (Yin and Schütze, 2015), Multi-Perspective CNN (He et al., 2015), MV-LSTM (Wan et al., 2016).

Socher et al. (2011) firstly used this paradigm for paraphrase detection. The representations of words or phrases are learned based on recursive autoencoders.

Hu et al. (2014) proposed to an end-to-end architecture with convolutional neural network (Arc-II) to model multiple-granularity interactions of two sentences.

Wan et al. (2016) used LSTM to enhance the positional contextual interactions of the words or phrases between two sentences. The input of LSTM for one sentence does not involve another sentence.

Another kind of methods is to model the conditional encoding, in which the encoding of one sentence can be affected by another sentence. Rocktäschel et al. (2015) and Wang and Jiang (2015) used LSTM to read pairs of sequences to produce a final representation, which can be regarded as interaction of two sequences. By incorporating an attention mechanism, they got further improvements to the predictive abilities.

Different with these two kinds of methods, we model the interactions of two texts in a recursively matching way. Based on this idea, we propose a model of deep fusion LSTMs to accomplish recursive conditional encodings.

8 Conclusion and Future Work

In this paper, we propose a model of deep fusion LSTMs to capture the strong interaction for text semantic matching. Experiments on two large scale text matching tasks demonstrate the efficacy of our proposed model and its superiority to competitor models. Besides, our visualization analysis revealed that multiple interpretable neurons in our model can capture the contextual interactions of the words or phrases.

[1] http://nlp.fudan.edu.cn/data/.

In future work, we would like to investigate our model on more text matching tasks.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints*, September.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Wonmin Byeon, Thomas M Breuel, Federico Raue, and Marcus Liwicki. 2015. Scene labeling with lstm recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3547–3555.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

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

Alex Graves and Jürgen Schmidhuber. 2009. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 545–552.

Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. 2007. Multi-dimensional recurrent neural networks. In *Artificial Neural Networks–ICANN 2007*, pages 549–558. Springer.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1586.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*.

Nal Kalchbrenner, Ivo Danihelka, and Alex Graves. 2015. Grid long short-term memory. *arXiv preprint arXiv:1507.01526*.

PengFei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. 2015. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the Conference on EMNLP*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Andrew M Saxe, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

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

Ke M. Tran, Arianna Bisazza, and Christof Monz. 2016. Recurrent memory network for language modeling. *CoRR*, abs/1601.01272.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*.

Shuohang Wang and Jing Jiang. 2015. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*.

Wenpeng Yin and Hinrich Schütze. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 901–911.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jun Zhao. 2013. Improving question retrieval in community question answering using world knowledge. In *IJCAI*.

Understanding Discourse on Work and Job-Related Well-Being in Public Social Media

Tong Liu[1*], **Christopher M. Homan**[1*], **Cecilia Ovesdotter Alm**[1$],
Ann Marie White[2], **Megan C. Lytle**[2], **Henry A. Kautz**[3]
[1*] Golisano College of Computing and Information Sciences, Rochester Institute of Technology
[1$] College of Liberal Arts, Rochester Institute of Technology
[2] University of Rochester Medical Center
[3] Department of Computer Science, University of Rochester
tl8313@rit.edu, cmh@cs.rit.edu, coagla@rit.edu
annmarie_white|megan_lytle@urmc.rochester.edu
kautz@cs.rochester.edu

Abstract

We construct a humans-in-the-loop supervised learning framework that integrates crowdsourcing feedback and local knowledge to detect job-related tweets from individual and business accounts. Using data-driven ethnography, we examine discourse about work by fusing language-based analysis with temporal, geospatial, and labor statistics information.

1 Introduction

Work plays a major role in nearly every facet of our lives. Negative and positive experiences at work places can have significant social and personal impacts. Employment condition is an important social determinant of health. But how exactly do jobs influence our lives, particularly with respect to well-being? Many theories address this question (Archambault and Grudin, 2012; Schaufeli and Bakker, 2004), but they are hard to validate as well-being is influenced by many factors, including geography as well as social and institutional support.

Can computers help us understand the complex relationship between work and well-being? Both are broad concepts that are difficult to capture objectively (for instance, the unemployment rate as a statistic is continually redefined) and thus challenging subjects for computational research.

Our first contribution is to propose a classification framework for such broad concepts as work that alternates between humans-in-the-loop annotation and machine learning over multiple iterations to simultaneously clarify human understanding of these concepts and automatically determine whether or not posts from public social media sites are about work. Our framework balances the effectiveness of crowdsourced workers with local experience, evaluates the degree of subjectivity throughout the process, and uses an iterative post-hoc evaluation method to address the problem of discovering gold standard data. Our performance (on an open-domain problem) demonstrates the value of our humans-in-the-loop approach which may be of special relevance to those interested in discourse understanding, particularly settings characterized by high levels of subjectivity, where integrating human intelligence into active learning processes is essential.

Our second contribution is to use our classifiers to study job-related discourse on social media using data-driven ethnography. Language is fundamentally a social phenomenon, and social media gives us a lens through which to observe a very particular form of discourse in real time. We add depth to the NLP analysis by gathering data from specific geographical regions to study discourse along a broad spectrum of interacting social groups, using work as a framing device, and we fuse language-based analysis with temporal, geospatial and labor statistics dimensions.

2 Background and Related Work

Though not the first study of job-related social media, prior ones used data from large companies' internal sites, whose users were employees (De Choudhury and Counts, 2013; Yardi et al., 2008; Kolari et al., 2007; Brzozowski, 2009). An obvious limitation in that case is it excludes populations without access to such restricted networks. Moreover, workers may not disclose true feelings about their jobs on such sites, since their employ-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1044–1053,
Berlin, Germany, August 7-12, 2016.

ers can easily monitor them. On the other hand, we show that on Twitter, it is quite common for tweets to relate negative feelings about work ("*I don't wanna go to work today*"), unprofessional behavior ("*Got drunk as hell last night and still made it to work*"), or a desire to work elsewhere ("*I want to go work at Disney World so bad*").

Nonetheless, these studies inform our work. DeChoudhury et al. (2013) investigated the landscape of emotional expression of the employees via enterprise internal microblogging. Yardi et al. (2008) examined temporal aspects of blogging usage within corporate internal blogging community. Kolari et al. (2007) characterized comprehensively how behaviors expressed in posts impact a company's internal social networks. Brzozowski (2009) described a tool that aggregated shared internal social media which when combined with its enterprise directory added understanding the organization and employees connections.

From a theoretical perspective, the Job Demands-Resources Model (Schaufeli and Bakker, 2004) suggests that job demands (e.g., overworked, dissonance, and conflict) lead to burnout and disengagement while resources (e.g., appreciation, cohesion, and safety) often result in satisfaction and productivity. Although burnout and engagement have an inverse relationship, these states fluctuate and can vary over time. In 2014, more than two-thirds of U.S. workers were disengaged at work (Gallup, 2015a) and this disconnection costs the U.S. up to $398 billion annually in lost work and medical treatment (Gallup, 2015b). Indeed, job dissatisfaction poses serious health risks and has even been linked to suicide (Hazards Magazine, 2014). Thus, examining social media for job-related messages provides a novel opportunity to study job discourse and associated demands and resources. Moreover, the declarative and affective tone of these tweets may have important implications for understanding the relationship between burnout and engagement with such public health concerns as mental health.

3 Humans-in-the-Loop Classification

From July 2013 to June 2014 we collected over 7M geo-tagged tweets from around 85,000 public accounts in a 15-county around a midsized city using DataSift[1]. We removed punctuation and special characters, and used the Internet Slang Dictionary[2] to normalize nonstandard terms.

Figure 1 shows our humans-in-the-loop framework for learning classifiers to identify job-related posts. It consists of four rounds of machine classification – similar to that of Li et al. (2014) except that our rounds are not as uniform – where the classifier in each round acts as a filter on our training data, providing human annotators a sample of Twitter data to label and (except for the final round) using these labeled data to train the classifiers in later rounds.

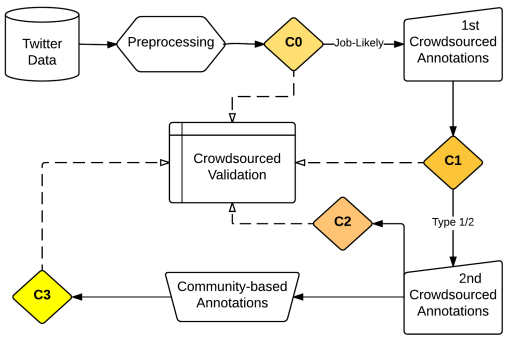


Figure 1: Flowchart of our humans-in-the-loop framework, laid out in Section 3.

The initial classifier C_0 is a simple term-matching filter; see Table 1 (number options were considered for some terms). The other classifiers (C_1, C_2, C_3) are SVMs that use a feature space of n-grams from the training set.

Include	job, jobless, manager, boss my/your/his/her/their/at work
Exclude	school, class, homework, student, course finals, good/nice/great job, boss ass[3]

Table 1: C_0 rules identifying *Job-Likely* tweets.

Round 1. We ran C_0 on our dataset. Approximately 40K tweets having at least five tokens passed this filter. We call them *Job-Likely tweets*. We randomly chose around 2,000 *Job-Likely* tweets and split them evenly into 50 AMT Human Intelligence Tasks (HITs), and further randomly duplicated five tweets in each HIT to evaluate each worker's consistency. Five crowdworkers assigned to each HIT[4] answered, for each tweet,

[1] http://datasift.com/

[2] http://www.noslang.com/dictionary

[3] Describe something awesome in a sense of utter dominance, magical superiority, or being ridiculously good.

[4] This is based on empirical insights for crowdsourced annotation tasks (Callison-Burch, 2009; Evanini et al., 2010).

the question: *Is this tweet about job or employment?* All crowdworkers lived in the U.S. and had an approval rating of 90% or better. They were paid $1.00 per HIT[5]. We assessed inter-annotator reliability among the five annotators in each HIT using Geertzen's tool (Geertzen, 2016).

This yielded 1,297 tweets where all 5 annotators agreed on the same label (Table 2). To balance our training data, we added 757 tweets chosen randomly from tweets outside the *Job-Likely* set that we labeled not job-related. C_1 trained on this set.

Round 2. Our goal was to collect 4,000 more labeled tweets that, when combined with the Round 1 training data, would yield a class-balanced set. Using C_1 to perform regression, we ranked the tweets in our dataset by the confidence score (Chang and Lin, 2011). We then spot-checked the tweets to estimate the frequency of job-related tweets as the confidence score increases. We discovered that among the top-ranked tweets about half, and near the separating hyperplane (i.e., where the confidence scores are near zero) almost none, are job-related.

Based on these estimates, we randomly sampled 2,400 tweets from those in the top 80th percentile of confidence scores (*Type-1*). We then randomly sampled about 800 tweets each from the first deciles of tweets greater and lesser than zero, respectively (*Type-2*).

The rationale for drawing from these two groups was that the false Type-1 tweets represent those on which the C_1 classifier most egregiously fails, and the Type-2 tweets are those closest to the feature vectors and those toward which the classifier is most sensitive.

Crowdworkers again annotated these tweets in the same fashion as in Round 1 (see Table 3), and cross-round comparisons are in Tables 2 and 4. We trained C_2 on all tweets from Round 1 and 2 with unanimous labels (bold in Table 2).

AMTs	job-related			not job-related		
	3	4	5	3	4	5
Round 1	104	389	**1027**	78	116	**270**
Round 2	140	287	**721**	66	216	**2568**

Table 2: Summary of both annotation rounds.

[5]We consulted with Turker Nation (`http://www.turkernation.com`) to ensure that the workers were treated and compensated fairly for their tasks. We also rewarded annotators based on the qualities of their work.

Round 2	job-related			not job-related		
	3	4	5	3	4	5
Type-1	129	280	713	50	149	1079
Type-2	11	7	8	16	67	1489

Table 3: Summary of tweet labels in Round 2 by confidence type (showing when 3/4/5 of 5 annotators agreed).

AMTs	Fleiss' kappa	Krippendorf's alpha
Round 1	0.62 ± 0.14	0.62 ± 0.14
Round 2	0.81 ± 0.09	0.81 ± 0.08

Table 4: Average $\pm$ stdev agreement from Round 1 and 2 are *Good*, *Very Good* (Altman, 1991).

Annotations	Sample Tweet
Y Y Y Y Y	Really bored....., no entertainment at work today
Y Y Y Y N	two more days of work then I finally get a day off.
Y Y Y N N	Leaving work at 430 and driving in this snow is going to be the death of me
Y Y N N N	Being a mommy is the hardest but most rewarding job a women can have #babyBliss #babybliss
Y N N N N	These refs need to DO THEIR FUCKING JOBS
N N N N N	One of the best Friday nights I've had in a while

Table 5: Inter-annotator agreement combinations with sample tweets. Y denotes job-related. Cases where the majority (not all) annotators agreed (3/4 out of 5) are underlined in bold.

Round 3. Two coauthors with prior experience from the local community reviewed instances from Round 1 and 2 on which crowdworkers disagreed (highlighted in Table 5) and provided labels. Cohen's kappa agreement was high: κ = 0.80. Combined with all labeled data from the previous rounds this yielded 2,670 gold-standard-labeled job-related and 3,250 not job-related tweets. We trained C_3 on this entire set. Since it is not strictly class-balanced, we grid-searched on a range of class weights and chose the estimator that optimized F1 score, using 10-fold cross validation[6]. Table 6 shows C_3's top-weighted features, which reflect the semantic field of work for the job-related class.

[6]These scores were determined respectively using the mean score over the cross-validation folds. The parameter settings that gave the best results on the left out data were a linear kernel with penalty parameter $C = 0.1$ and class weight ratio of 1:1.

job-related	weights	not job-related	weights
work	2.026	did	-0.714
job	1.930	amazing	-0.613
manager	1.714	nut	-0.600
jobs	1.633	hard	-0.571
managers	1.190	constr	-0.470
working	0.827	phone	-0.403
bosses	0.500	doing	-0.403
lovemyjob	0.500	since	-0.373
shift	0.487	brdg	-0.363
worked	0.410	play	-0.348
paid	0.374	its	-0.337
worries	0.369	think	-0.330
boss	0.369	thru	-0.329
seriously	0.368	hand	-0.321
money	0.319	awesome	-0.319

Table 6: Top 15 features for both classes of C_3.

Discovering Businesses. Manual examination of job-related tweets revealed patterns like: *Panera Bread: Baker – Night (#LOCATION) http://URL #Hospitality #VeteranJob #Job #Jobs #TweetMyJobs*. Nearly all tweets that contained at least one of these hashtags: *#veteranjob*, *#job*, *#jobs*, *#tweetmyjobs*, *#hiring*, *#retail*, *#realestate*, *#hr* also included a URL, which spot-checking revealed nearly always led to a recruitment website (see Table 7). This led to an effective heuristic to separate individual from business accounts only for posts that have first been classified as job-related: if an account had more job-related tweets with any of the above hashtags + URL patterns, we labeled it *business*; otherwise *individual*.

	hashtag only	hashtag + URL
#veteranjob	18,066	18,066
#job	79,362	79,327
#jobs	58,637	58,631
#tweetmyjobs	39,007	39,007
#hiring	148	147
#retail	17,037	17,035
#realestate	92	92
#hr	400	399

Table 7: Counts of hashtags queried, and counts of their subsets with hashtags coupled with URL.

4 Results and Discussion

Crowdsourced Validation The fundamental difficulty in open-domain classification problems such as this one is there is no gold-standard data to hold out at the beginning of the process. To address this, we adopted a post-hoc evaluation where we took balanced sets of labeled tweets from each classifier (C_0, C_1, C_2 and C_3) and asked AMT workers to label a total of 1,600 samples, taking the majority votes (where at least 3 out of 5 crowdworkers agreed) as reference labels. Our results (Table 8) show that C_3 performs the best, and significantly better than C_0 and C_1.

Estimating Effective Recall The two machine-labeled classes in our test data are roughly balanced, which is not the case in real-world scenarios. We estimated the effective recall under the assumption that the error rates in our test samples are representative of the entire dataset. Let y be the total number of the classifier-labeled "positive" elements in the entire dataset and n be the total of "negative" elements. Let y_t be the number of classifier-labeled "positive" tweets in our $1,600$-samples test set and let $n_t = 1,600 - y_t$. Then the estimated effective recall $\hat{R} = \frac{y \cdot n_t \cdot R}{y \cdot n_t \cdot R + n \cdot y_t \cdot (1-R)}$.

Model	Class	P	R	$\hat{\mathbf{R}}$	F1
C_0	job	0.72	0.33	0.01	0.45
	notjob	0.68	0.92	1.00	0.78
C_1	job	0.79	0.82	0.15	0.80
	notjob	0.88	0.86	0.99	0.87
C_2	job	0.82	0.95	0.41	0.88
	notjob	0.97	0.86	0.99	0.91
C_3	job	0.83	0.96	0.45	0.89
	notjob	0.97	0.87	0.99	0.92

Table 8: Crowdsourced validations of instances identified by 4 distinct models (1,600 total tweets).

Assessing Business Classifier For Table 8's tweets labeled by $C_0 - C_3$ as job-related, we asked AMT workers: *Is this tweet more likely from a personal or business account?* Table 9 shows that this method was quite accurate.

From	Class	P	R	F1
C_0	individual	0.86	1.00	0.92
	business	0.00	0.00	0.00
	avg/total	*0.74*	*0.86*	*0.79*
C_1	individual	1.00	0.97	0.98
	business	0.98	1.00	0.99
	avg/total	*0.99*	*0.99*	*0.99*
C_2	individual	1.00	0.98	0.99
	business	0.98	1.00	0.99
	avg/total	*0.99*	*0.99*	*0.99*
C_3	individual	1.00	0.99	0.99
	business	0.99	1.00	0.99
	avg/total	*0.99*	*0.99*	*0.99*

Table 9: Crowdsourced validations of individuals vs. businesses job-related tweets.

Our explanation for the strong performance of the business classifier is that the class of job-related tweets is relatively rare, and so by applying the classifier only to job-related tweets we sim-

plify the individual-or-business problem dramatically. Another, perhaps equally effective, simplification is that our tweets are geo-specific and so we automatically filter out business tweets from, e.g., national media.

Generalizability Tests Can our best model C_3 discover job-related tweets from other geographical regions, even though it was trained on data from one specific region? We repeated the tests above on 400 geo-tagged tweets from Detroit (balanced between job-related and not). Table 10 shows that C_3 and the business classifier generalize well to another region. This suggests the transferability of our humans-in-the-loop classification framework and of heuristic to separate individual from business accounts for tweets classified as job-related.

Model	Class	P	R	F1
C_3	job	0.85	0.99	0.92
	notjob	0.99	0.87	0.93
Heuristic	individual	1.00	0.96	0.98
	business	0.96	1.00	0.98
	avg/total	*0.98*	*0.98*	*0.98*

Table 10: Validations of C_3 and business classifier on Detroit data.

5 Understanding Job-Related Discourse

Using the job-related tweets – from both individual and business accounts – extracted by C_3 from the July 2013-June 2014 dataset (see Table 12), we conducted the following analyses.

C_3 Versus C_0 The fact that C_3 outperforms C_0 demonstrates our humans-in-the-loop framework is necessary and effective compared to an intuitive term-matching filter. We further examined the messages labeled as job-related by C_3, but not captured by C_0. More than 160,000 tweets fell into this *Difference* set, in which approximately 85,000 tweets are from individual accounts while the rest are from business accounts. Table 11 shows the top 3 most frequent uni-, bi-, and trigrams in the *Difference* dataset. These n-grams from the individual group suggest that people often talk about job-related topics while mentioning temporal information or announcing their working schedules. We neglected such time-related phrases when defining C_0. In contrast, the frequencies of the listed n-grams in the business group are much higher than those in the individual group. This indicates that our definitions of inclusion terms in C_0 did not capture a considerable amount of posts involving broad job-related topics, which is also reflected in Table 9: our business classifier did not find business accounts from the job-related tweets extracted by C_0.

Individual	Business
Unigrams	
day, 6989	ny, 83907
today, 5370	#job, 75178
good, 4245	#jobs, 55105
Bigrams	
last night, 359	#jobs #tweetmyjobs, 32165
getting ready, 354	#rochester ny, 22101
first day, 296	#job #jobs, 16923
Trigrams	
working hour shift, 51	#job #jobs #tweetmyjobs, 12004
first day back, 48	ny #jobs #tweetmyjobs, 4955
separate leader follower, 44	ny #retail #job, 4704

Table 11: Top 3 most frequent uni-, bi-, and trigrams with frequencies in the *Difference* set.

Hashtags Individuals posted 11,935 unique hashtags and businesses only 414. The top 250 hashtags from each group are shown in Figure 2.

Figure 2: Hashtags in job-related tweets: above – individual accounts; below – business accounts.

Individual users used an abbreviation for the name of the midsized city to mark their location,

and *fml*[7] to express personal embarrassing stories. *Work* and *job* are self-explanatory. *Money*, *motivation* relates to jobs. *Tired*, *exhausted*, *fuck*, *insomnia*, *bored*, *struggle* express negative conditions. Likewise, *lovemyjob*, *happy*, *awesome*, *excited*, *yay*, *tgif*[8] convey positive affects experienced from jobs. Business accounts exhibit distinct patterns. Besides the hashtags queried (Table 7), we saw local place names, like *corning*, *rochester*, *batavia*, *pittsford*, and regional ones like *syracuse*, *ithaca*. *Customerservice*, *nursing*, *accounting*, *engineering*, *hospitality*, *construction* record occupations, while *kellyjobs*, *familydollar*, *cintasjobs*, *cfgjobs*, *searsjobs* point to business agents. Unlike individual users, businesses do not use hashtags reflecting affective expressions.

Linguistic Differences We used the TweetNLP POS tagger (Gimpel et al., 2011). Figure 3 shows nine part-of-speech tag[9] frequencies for three subsets of tweets.

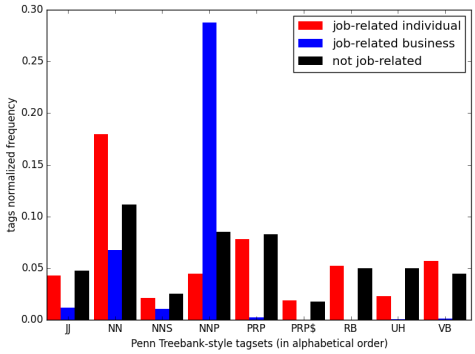


Figure 3: POS tag comparisons (normalized, averaged) among three subsets of tweets: job-related tweets from individual accounts (red), job-related tweets from business accounts (blue) and not job-related tweets (black).

Business accounts use NNPs more than individuals, perhaps because they often advertise job openings at specific locations, like *New York*, *Sears*. Individuals use NNPs less frequently and in a more casual way, e.g., *Jojo*, *galactica*, *Valli*. Also, individuals use JJ, NN, NNS, PRP, PRP$, RB, UH, and VB more regularly than business accounts do. Not job-related tweets have similar patterns to job-related ones from individual accounts, suggesting that individual users exhibit analogous language habits regardless of topic.

Temporal Patterns Our findings that individual users frequently used time-related n-grams (Table 11) prompted us to examine the temporal patterns of job discourse.

Figure 4a suggests that individuals talk about jobs the most in December and January (which also have the most tweets over other topics), and the least in the warmer months. July witnesses the busiest job-related tweeting from business and January the least. The user community is slightly less active in the warmer months, with fewer tweets then.

Figure 4b shows that job-related tweet volumes are higher on weekdays and lower on weekends, following the standard work week. Weekends see fewer business tweets than weekdays do. Sunday is the most – while Friday and Saturday are the least – active days from the not job-related perspective.

Figure 4c shows hourly trends. Job-related tweets from business accounts are most frequent during business hours, peaking at 11, and then taper off. Perhaps professionals are either getting their commercial tasks completed before lunch, or expecting others to check updates during lunch. Individuals post about jobs almost anytime awake and have a similar distribution to non-job-related tweets.

Measuring Affective Changes We examined positive affect (PA) and negative affect (NA) to measure diurnal changes in public mood (Figures 5 and 6), using two recognized lexicons, in job-related tweets from individual accounts (left), job-related tweets from business accounts (middle), and not job-related tweets (right).

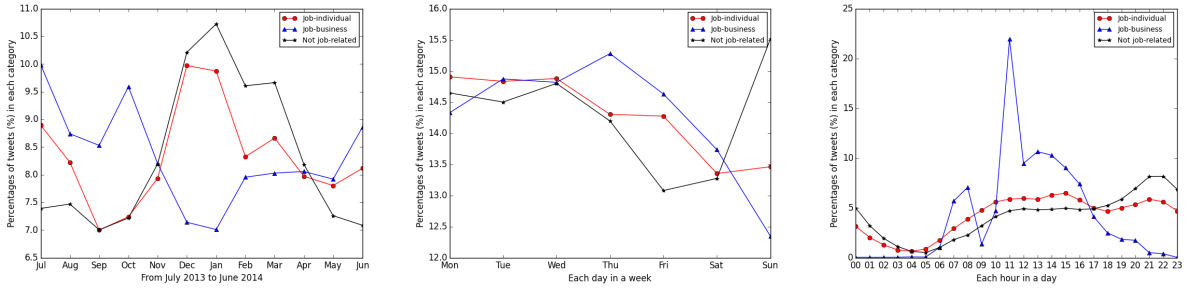
(1) Linguistic Inquiry and Word Count We used LIWC's *positive emotion* and *negative emotion* to represent PA and NA respectively (Pennebaker et al., 2001) because it is common in behavioral health studies, and used as a standard comparison in referenced work. Figure 5 shows the mean daily trends of PA and NA.[10] Panels 5a and 5b reveal contrasting job-related affective patterns, compared to prior trends from

[7] An acronym for *Fuck My Life*.

[8] An acronym for *Thank God It's Friday* to express the joy one feels in knowing that the work week has officially ended and that one has two days off which to enjoy.

[9] JJ – Adjective; NN – Noun (singular or mass); NNS – Noun (plural); NNP – Proper noun (singular); PRP – Personal pronoun; PRP$ – Possessive pronoun; RB – Adverb; UH – Interjection; VB – Verb (base form) (Santorini, 1990).

[10] Non-equal y-axes help show peak/valley patterns here and in Figure 6, also motivated by lexicon's unequal sizes.



(a) In each month (b) On each day of week (c) In each hour

Figure 4: Distributions of job-related tweets over time by job class. We converted timestamps from the Coordinated Universal Time standard (UTC) to local time zone with daylight saving time taken into account.

enterprise-wide micro-blog usage (De Choudhury and Counts, 2013), i.e., public social media exhibit gradual increase in PA while internal enterprise network decrease after business. This perhaps confirms our suspicion that people talk about work on public social media differently than on work-based media.

(2) Word-Emotion Association Lexicon We focused on the words from EmoLex's *positive* and *negative* categories, which represent sentiment polarities (Mohammad and Turney, 2013; Mohammad and Turney, 2010) and calculated the score for each tweet similarly as LIWC. The average daily positive and negative sentiment scores in Figure 6 display patterns analogous to Figure 5.

Labor Statistics We explored associations between Twitter temporal patterns, affect, and official labor statistics (Figure 8). These monthly statistics[11] include: labor force, employment, unemployment, and unemployment rate. We collected one more year of Twitter data from the same area, and applied C_3 to extract the job-related posts from individual and business accounts (Table 12 summarizes the basic statistics), then defined the following monthwise statistics for our two-year dataset: count of overall/job-individual/job-business/others tweets; percentage of job-individual/job-business/others tweets in overall tweets; average LIWC PA/NA scores of job-individual/job-business/others tweets[12].

Positive affect expressed in job-related discourse from both individual and business accounts correlate negatively with unemployment and unemployment rate. This is intuitive, as unemployment is generally believed to have a negative impact on individuals' lives. The counts of job-related tweets from individual and not job-related tweets are both positively correlated with unemployment and unemployment rate, suggesting that unemployment may lead to more activities in public social media. This correlation result shows that online textual disclosure themes and behaviors can reflect institutional survey data.

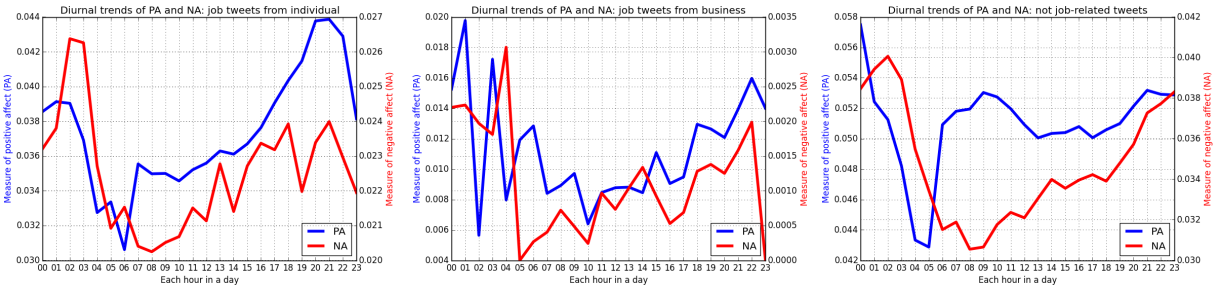
Inside vs. Outside City We compared tweets occurring within the city boundary to those lying outside (Table 13). The percentages of job-related tweets from individual accounts, either in urban or rural areas, remain relatively even. The proportion of job-related tweets from business accounts decreased sharply from urban to rural locations. This may be because business districts are usually centered in urban areas and individual tweets reflect more complex geospatial distributions.

Job-Life Cycle Model Based on hand inspection of a large number of job-related tweets and on models of the relationship between work and wellness found in behavioral studies (Archambault and Grudin, 2012; Schaufeli and Bakker, 2004), we tentatively propose a job-life model for job-related discourse from individual accounts (Figure 7). Each state in the model has three dimensions: the *point of view*, the *affect*, and the job-related *activity*, in terms of basic level of employment, expressed in the tweet.

We concatenated together all job-related tweets posted by each individual into a single document and performed latent Dirichlet allocation (LDA) (Blei et al., 2003) on this user-level corpus, using Gensim (Řehůřek and Sojka, 2010). We used 12

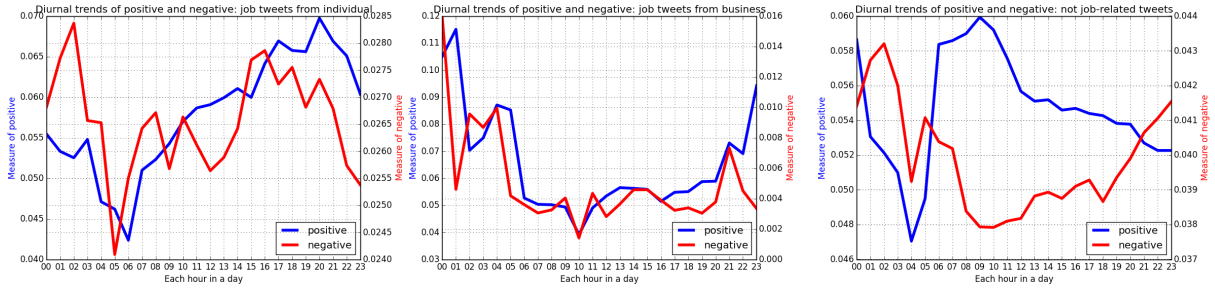
[11] Published by US Department of Labor, including: Local Area Unemployment Statistics; State and Metro Area Employment, Hours, and Earnings.

[12] IND: individual; BIZ: business; pct: %; avg: average.



(a) Job-related tweets from individuals (b) Job-related tweets from businesses (c) Not job-related tweets

Figure 5: Diurnal trends of positive and negative affect based on LIWC.



(a) Job-related tweets from individuals (b) Job-related tweets from businesses (c) Not job-related tweets

Figure 6: Diurnal trends of positive and negative affect based on EmoLex.

Unique counts	job-related tweets from individual accounts		job-related tweets from business accounts		not job-related tweets	
	tweets	accounts	tweets	accounts	tweets	accounts
July 2013 - June 2014	114,302	17,183	79,721	292	6,912,306	84,718
July 2014 - June 2015	85,851	16,350	115,302	333	5,486,943	98,716
Total (unique counts)	200,153	28,161	195,023	431	12,399,249	136,703

Table 12: Summary statistics of the two-year Twitter data classified by C_3.

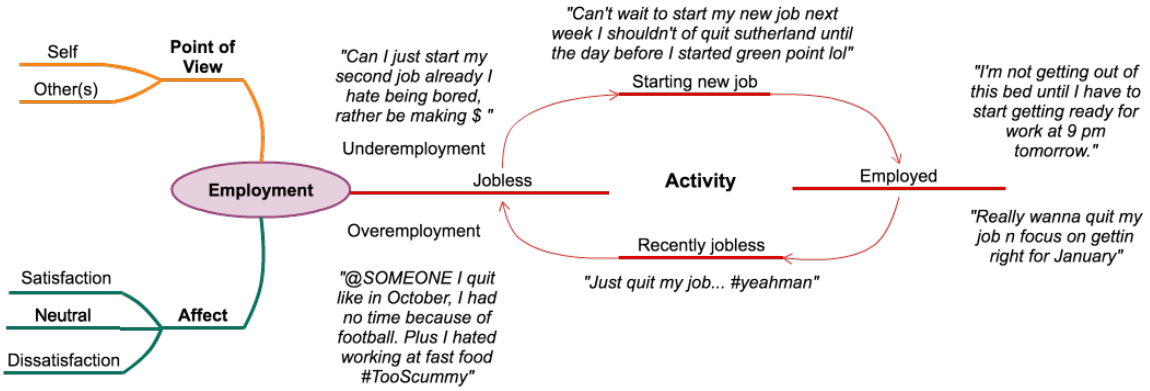


Figure 7: The job-life model captures the point of view, affect, and job-related activity in tweets.

%	job-related individual	job-related business	others
Inside	1.59	3.73	94.68
Outside	1.85	1.51	96.65
Combined	1.82	1.77	96.41

Table 13: Percent inside and outside city tweets.

topics for the LDA based on the number of affect classes (three) times the number of job-related activities (four). See Table 14.

Topic 0 appears to be about getting ready to start a job, and topic 1 about leaving work permanently or temporarily. Topics 2, 5, 6, 8, and 11 suggest how key affect is for understanding job-

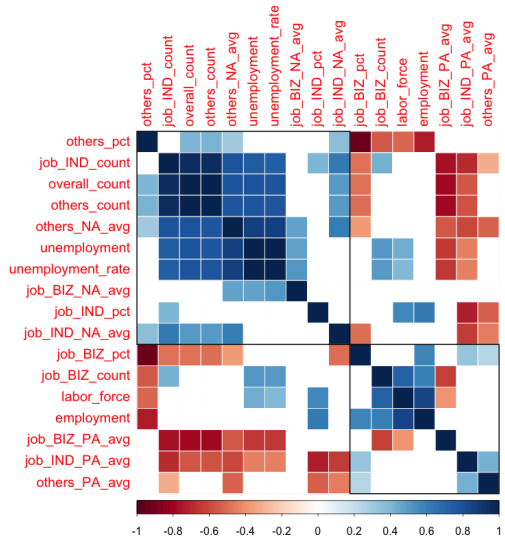


Figure 8: Correlation matrix with Spearman used for test at level .05, with insignificant coefficients left blank. The matrix is ordered by a hierarchical clustering algorithm. Blue – positive correlation, red – negative correlation.

Topic index	Representative words
0	getting, ready, day, first, hopefully
1	last, finally, week, break, last_day
2	fucking, hate, seriously, lol, really
3	come_visit, some, talking, pissed
4	weekend, today, home, thank_god
5	wish, love, better, money, working
6	shift, morning, leave, shit, bored
7	manager, guy, girl, watch, keep
8	feel, sure, supposed, help, miss
9	much, early, long, coffee, care
10	time, still, hour, interview, since
11	best, pay, bored, suck, proud

Table 14: The top five words in each of the twelve topics discovered by LDA.

related discourse: 2 and 6 lean towards dissatisfaction and 5 toward satisfaction. 11 looks like a mixture. Topic 7 connects to coworkers. Many topics point to the importance of time (including leisure time in topic 4).

6 Conclusion

We used crowdsourcing and local expertise to power a humans-in-the-loop classification framework that iteratively improves identification of public job-related tweets. We separated business accounts from individual in job-related discourse. We also analyzed identified tweets integrating temporal, affective, geospatial, and statistical information. While jobs take up enormous amounts of most adults' time, job-related tweets are still rather infrequent. Examining affective changes reveals that PA and NA change independently; low NA appears to indicate the absence of negative feelings, not the presence of positive ones.

Our work is of social importance to working-age adults, especially for those who may struggle with job-related issues. Besides providing insights for discourse and its links to social science, our study could lead to practical applications, such as: aiding policy-makers with macro-level insights on job markets, connecting job-support resources to those in need, and facilitating the development of job recommendation systems.

This work has limitations. We did not study whether providing contextual information in our humans-in-the-loop framework would influence the model performance. This is left for future work. Additionally we recognize that the hashtag inventory used to discover business accounts from job-related topics might need to change over time, to achieve robust performance in the future. As another point, due to Twitter demographics, we are less likely to observe working seniors.

Acknowledgments

We thank the anonymous reviewers for their helpful comments and suggestions. This work was supported in part by a GCCIS Kodak Endowed Chair Fund Health Information Technology Strategic Initiative Grant and NSF Award #SES-1111016.

References

DG Altman. 1991. Inter-Rater Agreement. *Practical Statistics for Medical Research*, 5:403–409.

Anne Archambault and Jonathan Grudin. 2012. A Longitudinal Study of Facebook, LinkedIn, & Twitter Use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2741–2750. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *the Journal of Machine Learning Research*, 3:993–1022.

Michael J Brzozowski. 2009. Watercooler: Exploring an Organization through Enterprise Social Media. In *Proceedings of the ACM 2009 International Conference on Supporting Group Work*, pages 219–228. ACM.

Chris Callison-Burch. 2009. Fast, Cheap, and Creative: Evaluating Translation Quality Using Amazon's Mechanical Turk. In *Proceedings of the 2009*

Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1, pages 286–295. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27.

Munmun De Choudhury and Scott Counts. 2013. Understanding Affect in the Workplace via Social Media. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, pages 303–316. ACM.

Keelan Evanini, Derrick Higgins, and Klaus Zechner. 2010. Using Amazon Mechanical Turk for Transcription of Non-Native Speech. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 53–56. Association for Computational Linguistics.

Gallup. 2015a. Majority of U.S. Employees not Engaged despite Gains in 2014.

Gallup. 2015b. Only 35% of U.S. Managers are Engaged in their Jobs.

Jeroen Geertzen. 2016. Inter-Rater Agreement with Multiple Raters and Variables. [Online; accessed 17-February-2016].

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Hazards Magazine. 2014. Work Suicide.

Pranam Kolari, Tim Finin, Kelly Lyons, Yelena Yesha, Yaacov Yesha, Stephen Perelgut, and Jen Hawkins. 2007. On the Structure, Properties and Utility of Internal Corporate Blogs. *Growth*, 45000:50000.

Jiwei Li, Alan Ritter, Claire Cardie, and Eduard H Hovy. 2014. Major Life Event Extraction from Twitter based on Congratulations/Condolences Speech Acts. In *EMNLP*, pages 1997–2007.

Saif M Mohammad and Peter D Turney. 2010. Emotions Evoked by Common Words and Phrases: Using Mechanical Turk to Create An Emotion Lexicon. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 26–34. Association for Computational Linguistics.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. 29(3):436–465.

James W Pennebaker, Martha E Francis, and Roger J Booth. 2001. Linguistic Inquiry and Word Count: LIWC 2001. *Mahway: Lawrence Erlbaum Associates*, 71:2001.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA.

Beatrice Santorini. 1990. Part-of-Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision).

Wilmar B Schaufeli and Arnold B Bakker. 2004. Job Demands, Job Resources, and Their Relationship with Burnout and Engagement: A Multi-Sample Study. *Journal of Organizational Behavior*, 25(3):293–315.

Sarita Yardi, Scott Golder, and Mike Brzozowski. 2008. The Pulse of the Corporate Blogosphere. In *Conf. Supplement of CSCW 2008*, pages 8–12. Citeseer.

Achieving Open Vocabulary Neural Machine Translation with Hybrid Word-Character Models

Minh-Thang Luong and **Christopher D. Manning**
Computer Science Department, Stanford University, Stanford, CA 94305
{lmthang,manning}@stanford.edu

Abstract

Nearly all previous work on neural machine translation (NMT) has used quite restricted vocabularies, perhaps with a subsequent method to patch in unknown words. This paper presents a novel word-character solution to achieving open vocabulary NMT. We build hybrid systems that translate mostly at the *word* level and consult the *character* components for rare words. Our character-level recurrent neural networks compute source word representations and recover unknown target words when needed. The twofold advantage of such a hybrid approach is that it is much faster and easier to train than character-based ones; at the same time, it never produces unknown words as in the case of word-based models. On the WMT'15 English to Czech translation task, this hybrid approach offers an addition boost of +2.1−11.4 BLEU points over models that already handle unknown words. Our best system achieves a new state-of-the-art result with 20.7 BLEU score. We demonstrate that our character models can successfully learn to not only generate well-formed words for Czech, a highly-inflected language with a very complex vocabulary, but also build correct representations for English source words.

1 Introduction

Neural Machine Translation (NMT) is a simple new architecture for getting machines to translate. At its core, NMT is a single deep neural network that is trained end-to-end with several advantages such as simplicity and generalization. Despite being relatively new, NMT has already achieved state-of-the-art translation results for several language pairs such as English-French (Luong et al., 2015b), English-German (Jean et al., 2015a; Luong et al., 2015a; Luong and Manning, 2015), and English-Czech (Jean et al., 2015b).

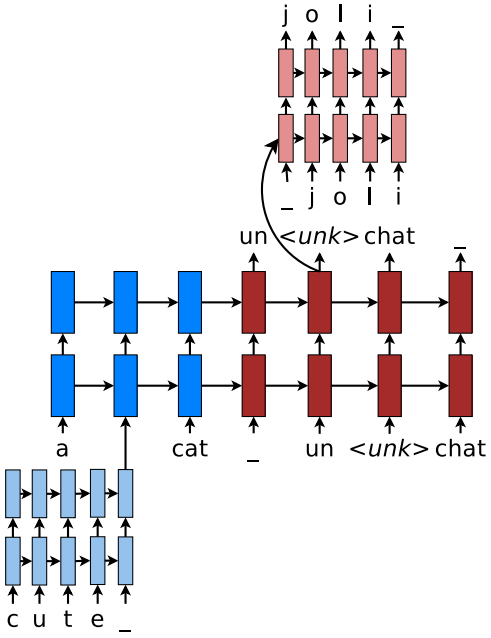


Figure 1: **Hybrid NMT** – example of a word-character model for translating "a cute cat" into "un joli chat". Hybrid NMT translates at the word level. For rare tokens, the character-level components build source representations and recover target <*unk*>. "_" marks sequence boundaries.

While NMT offers many advantages over traditional phrase-based approaches, such as small memory footprint and simple decoder implementation, nearly all previous work in NMT has used quite restricted vocabularies, crudely treating all other words the same with an <*unk*> symbol. Sometimes, a post-processing step that patches in unknown words is introduced to alleviate this problem. Luong et al. (2015b) propose to annotate

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1054–1063,
Berlin, Germany, August 7-12, 2016.

occurrences of target <*unk*> with positional information to track their alignments, after which simple word dictionary lookup or identity copy can be performed to replace <*unk*> in the translation. Jean et al. (2015a) approach the problem similarly but obtain the alignments for unknown words from the attention mechanism. We refer to these as the *unk replacement* technique.

Though simple, these approaches ignore several important properties of languages. First, *monolingually*, words are morphologically related; however, they are currently treated as independent entities. This is problematic as pointed out by Luong et al. (2013): neural networks can learn good representations for frequent words such as "distinct", but fail for rare-but-related words like "distinctiveness". Second, *crosslingually*, languages have different alphabets, so one cannot naïvely memorize all possible surface word translations such as name transliteration between "Christopher" (English) and "Kryštof" (Czech). See more on this problem in (Sennrich et al., 2016).

To overcome these shortcomings, we propose a novel *hybrid* architecture for NMT that translates mostly at the word level and consults the character components for rare words when necessary. As illustrated in Figure 1, our hybrid model consists of a word-based NMT that performs most of the translation job, except for the two (hypothetically) rare words, "cute" and "joli", that are handled separately. On the *source* side, representations for rare words, "cute", are computed on-the-fly using a deep recurrent neural network that operates at the character level. On the *target* side, we have a separate model that recovers the surface forms, "joli", of <*unk*> tokens character-by-character. These components are learned jointly end-to-end, removing the need for a separate unk replacement step as in current NMT practice.

Our hybrid NMT offers a twofold advantage: it is much faster and easier to train than character-based models; at the same time, it never produces unknown words as in the case of word-based ones. We demonstrate at scale that on the WMT'15 English to Czech translation task, such a hybrid approach provides an additional boost of +2.1−11.4 BLEU points over models that already handle unknown words. We achieve a new state-of-the-art result with 20.7 BLEU score. Our analysis demonstrates that our character models can successfully learn to not only generate well-formed words for Czech, a highly-inflected language with a very complex vocabulary, but also build correct representations for English source words.

We provide code, data, and models at `http://nlp.stanford.edu/projects/nmt`.

2 Related Work

There has been a recent line of work on end-to-end character-based neural models which achieve good results for part-of-speech tagging (dos Santos and Zadrozny, 2014; Ling et al., 2015a), dependency parsing (Ballesteros et al., 2015), text classification (Zhang et al., 2015), speech recognition (Chan et al., 2016; Bahdanau et al., 2016), and language modeling (Kim et al., 2016; Jozefowicz et al., 2016). However, success has not been shown for cross-lingual tasks such as machine translation.[1] Sennrich et al. (2016) propose to segment words into smaller units and translate just like at the word level, which does not learn to understand relationships among words.

Our work takes inspiration from (Luong et al., 2013) and (Li et al., 2015). Similar to the former, we build representations for rare words on-the-fly from subword units. However, we utilize recurrent neural networks with characters as the basic units; whereas Luong et al. (2013) use recursive neural networks with morphemes as units, which requires existence of a morphological analyzer. In comparison with (Li et al., 2015), our hybrid architecture is also a hierarchical sequence-to-sequence model, but operates at a different granularity level, word-character. In contrast, Li et al. (2015) build hierarchical models at the sentence-word level for paragraphs and documents.

3 Background & Our Models

Neural machine translation aims to directly model the conditional probability $p(y|x)$ of translating a source sentence, $x_1, \ldots, x_n$, to a target sentence, $y_1, \ldots, y_m$. It accomplishes this goal through an *encoder-decoder* framework (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Cho et al., 2014). The *encoder* computes a representation $\boldsymbol{s}$ for each source sentence. Based on that source

[1] Recently, Ling et al. (2015b) attempt character-level NMT; however, the experimental evidence is weak. The authors demonstrate only small improvements over word-level baselines and acknowledge that there are no differences of significance. Furthermore, only small datasets were used without comparable results from past NMT work.

representation, the *decoder* generates a translation, one target word at a time, and hence, decomposes the log conditional probability as:

$$\log p(y|x) = \sum_{t=1}^{m} \log p\left(y_t|y_{<t}, \boldsymbol{s}\right) \quad (1)$$

A natural model for sequential data is the recurrent neural network (RNN), used by most of the recent NMT work. Papers, however, differ in terms of: (a) architecture – from unidirectional, to bidirectional, and deep multi-layer RNNs; and (b) RNN type – which are long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) and the gated recurrent unit (Cho et al., 2014). All our models utilize the *deep multi-layer* architecture with *LSTM* as the recurrent unit; detailed formulations are in (Zaremba et al., 2014).

Considering the top recurrent layer in a deep LSTM, with $\boldsymbol{h}_t$ being the current target hidden state as in Figure 2, one can compute the probability of decoding each target word y_t as:

$$p\left(y_t|y_{<t}, \boldsymbol{s}\right) = \text{softmax}\left(\boldsymbol{h}_t\right) \quad (2)$$

For a parallel corpus $\mathbb{D}$, we train our model by minimizing the below cross-entropy loss:

$$J = \sum_{(x,y)\in\mathbb{D}} -\log p(y|x) \quad (3)$$

Attention Mechanism – The early NMT approaches (Sutskever et al., 2014; Cho et al., 2014), which we have described above, use only the last encoder state to initialize the decoder, i.e., setting the input representation $\boldsymbol{s}$ in Eq. (1) to $[\bar{\boldsymbol{h}}_n]$. Recently, Bahdanau et al. (2015) propose an *attention mechanism*, a form of random access memory for NMT to cope with long input sequences. Luong et al. (2015a) further extend the attention mechanism to different scoring functions, used to compare source and target hidden states, as well as different strategies to place the attention. In all our models, we utilize the *global* attention mechanism and the *bilinear form* for the attention scoring function similar to (Luong et al., 2015a).

Specifically, we set $\boldsymbol{s}$ in Eq. (1) to the set of source hidden states at the top layer, $[\bar{\boldsymbol{h}}_1, \ldots, \bar{\boldsymbol{h}}_n]$. As illustrated in Figure 2, the attention mechanism consists of two stages: (a) *context vector* – the current hidden state $\boldsymbol{h}_t$ is compared with individual source hidden states in $\boldsymbol{s}$ to learn an alignment vector, which is then used to compute the context vector $\boldsymbol{c}_t$ as a weighted average of $\boldsymbol{s}$; and (b) *attentional hidden state* – the context vector $\boldsymbol{c}_t$ is then

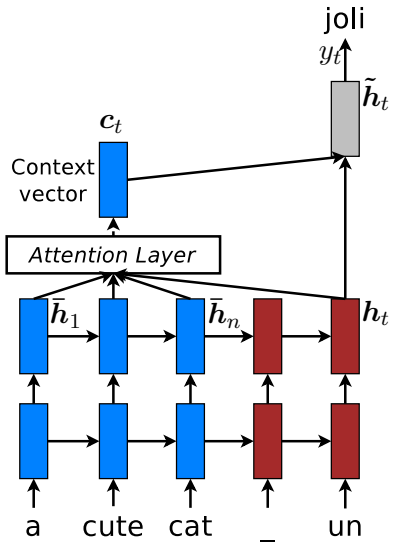


Figure 2: **Attention mechanism**.

used to derive a new attentional hidden state:

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad (4)$$

The attentional vector $\tilde{\boldsymbol{h}}_t$ then replaces $\boldsymbol{h}_t$ in Eq. (2) in predicting the next word.

4 Hybrid Neural Machine Translation

Our hybrid architecture, illustrated in Figure 1, leverages the power of both words and characters to achieve the goal of open vocabulary NMT. The core of the design is a *word*-level NMT with the advantage of being fast and easy to train. The *character* components empower the word-level system with the abilities to compute any source word representation on the fly from characters and to recover character-by-character unknown target words originally produced as <*unk*>.

4.1 Word-based Translation as a Backbone

The core of our hybrid NMT is a deep LSTM encoder-decoder that translates at the *word* level as described in Section 3. We maintain a vocabulary of $|V|$ frequent words for each language. Other words not inside these lists are represented by a universal symbol <*unk*>, one per language. We translate just like a word-based NMT system with respect to these source and target vocabularies, except for cases that involve <*unk*> in the source input or the target output. These correspond to the character-level components illustrated in Figure 1.

A nice property of our hybrid approach is that by varying the vocabulary size, one can control

how much to blend the word- and character-based models; hence, taking the best of both worlds.

4.2 Source Character-based Representation

In regular word-based NMT, for all rare words outside the source vocabulary, one feeds the universal embedding representing <*unk*> as input to the encoder. This is problematic because it discards valuable information about the source word. To fix that, we learn a deep LSTM model over characters of source words. For example, in Figure 1, we run our deep character-based LSTM over 'c', 'u', 't', 'e', and '_' (the boundary symbol). The final hidden state at the top layer will be used as the on-the-fly representation for the current rare word.

The layers of the deep character-based LSTM are always initialized with *zero* states. One might propose to connect hidden states of the word-based LSTM to the character-based model; however, we chose this design for various reasons. First, it simplifies the architecture. Second, it allows for efficiency through *precomputation*: before each mini-batch, we can compute representations for rare source words all at once. All instances of the same word share the same embedding, so the computation is per *type*.[2]

4.3 Target Character-level Generation

General word-based NMT allows generation of <*unk*> in the target output. Afterwards, there is usually a post-processing step that handles these unknown tokens by utilizing the alignment information derived from the attention mechanism and then performing simple word dictionary lookup or identity copy (Luong et al., 2015a; Jean et al., 2015a). While this approach works, it suffers from various problems such as alphabet mismatches between the source and target vocabularies and multi-word alignments. Our goal is to address all these issues and create a coherent framework that handles an unlimited output vocabulary.

Our solution is to have a separate deep LSTM that "translates" at the character level given the current word-level state. We train our system such that whenever the word-level NMT produces an <*unk*>, we can consult this character-level decoder to recover the correct surface form of the unknown target word. This is illustrated in Figure 1.

[2]While Ling et al. (2015b) found that it is slow and difficult to train source character-level models and had to resort to pretraining, we demonstrate later that we can train our deep character-level LSTM perfectly fine in an end-to-end fashion.

The training objective in Eq. (3) now becomes:

$$J = J_w + \alpha J_c \tag{5}$$

Here, J_w refers to the usual loss of the word-level NMT; in our example, it is the sum of the negative log likelihood of generating {"un", "<*unk*>", "chat", "_"}. The remaining component J_c corresponds to the loss incurred by the character-level decoder when predicting characters, e.g., {'j', 'o', 'l', 'i', '_'}, of those rare words not in the target vocabulary.

Hidden-state Initialization Unlike the source character-based representations, which are context-independent, the target character-level generation requires the current word-level context to produce meaningful translation. This brings up an important question about what can best represent the current context so as to initialize the character-level decoder. We answer this question in the context of the attention mechanism (§3).

The final vector $\tilde{\boldsymbol{h}}_t$, just before the softmax as shown in Figure 2, seems to be a good candidate to initialize the character-level decoder. The reason is that $\tilde{\boldsymbol{h}}_t$ combines information from both the context vector $\boldsymbol{c}_t$ and the top-level recurrent state $\boldsymbol{h}_t$. We refer to it later in our experiments as the *same-path* target generation approach.

On the other hand, the same-path approach worries us because all vectors $\tilde{\boldsymbol{h}}_t$ used to seed the character-level decoder might have similar values, leading to the same character sequence being produced. The reason is because $\tilde{\boldsymbol{h}}_t$ is directly used in the softmax, Eq. (2), to predict the same <*unk*>. That might pose some challenges for the model to learn useful representations that can be used to accomplish two tasks at the same time, that is to predict <*unk*> and to generate character sequences. To address that concern, we propose another approach called the *separate-path* target generation.

Our separate-path target generation approach works as follows. We mimic the process described in Eq. (4) to create a counterpart vector $\breve{\boldsymbol{h}}_t$ that will be used to seed the character-level decoder:

$$\breve{\boldsymbol{h}}_t = \tanh(\breve{\boldsymbol{W}}[\boldsymbol{c}_t; \boldsymbol{h}_t]) \tag{6}$$

Here, $\breve{\boldsymbol{W}}$ is a new learnable parameter matrix, with which we hope to release $\boldsymbol{W}$ from the pressure of having to extract information relevant to both the word- and character-generation processes. Only the hidden state of the first layer

is initialized as discussed above. The other components in the character-level decoder such as the LSTM cells of all layers and the hidden states of higher layers, all start with zero values.

Implementation-wise, the computation in the character-level decoder is done per word *token* instead of per *type* as in the source character component (§4.2). This is because of the context-dependent nature of the decoder.

Word-Character Generation Strategy With the character-level decoder, we can view the final hidden states as representations for the surface forms of unknown tokens and could have fed these to the next time step. However, we chose not to do so for the efficiency reason explained next; instead, <*unk*> is fed to the word-level decoder "as is" using its corresponding word embedding.

During *training*, this design choice decouples all executions over <*unk*> instances of the character-level decoder as soon the word-level NMT completes. As such, the forward and backward passes of the character-level decoder over rare words can be invoked in batch mode. At *test* time, our strategy is to first run a beam search decoder at the word level to find the best translations given by the word-level NMT. Such translations contains <*unk*> tokens, so we utilize our character-level decoder with beam search to generate actual words for these <*unk*>.

5 Experiments

We evaluate the effectiveness of our models on the publicly available WMT'15 translation task from English into Czech with *newstest2013* (3000 sentences) as a development set and *newstest2015* (2656 sentences) as a test set. Two metrics are used: case-sensitive NIST BLEU (Papineni et al., 2002) and chrF$_3$ (Popović, 2015).[3] The latter measures the amounts of overlapping character n-grams and has been argued to be a better metric for translation tasks out of English.

5.1 Data

Among the available language pairs in WMT'15, all involving English, we choose *Czech* as a target language for several reasons. First and foremost, Czech is a Slavic language with not only rich and complex inflection, but also fusional morphology in which a single morpheme can encode multiple grammatical, syntactic, or semantic meanings. As a result, Czech possesses an enormously large vocabulary (about 1.5 to 2 times bigger than that of English according to statistics in Table 1) and is a challenging language to translate into. Furthermore, this language pair has a large amount of training data, so we can evaluate at scale. Lastly, though our techniques are language independent, it is easier for us to work with Czech since Czech uses the Latin alphabet with some diacritics.

	English		Czech	
	word	char	word	char
# Sents	15.8M			
# Tokens	254M	1,269M	224M	1,347M
# Types	1,172K	2003	1,760K	2053
200-char	98.1%		98.8%	

Table 1: **WMT'15 English-Czech data** – shown are various statistics of our training data such as *sentence*, *token* (word and character counts), as well as *type* (sizes of the word and character vocabularies). We show in addition the amount of words in a vocabulary expressed by a list of 200 characters found in frequent words.

In terms of preprocessing, we apply only the standard tokenization practice.[4] We choose for each language a list of 200 characters found in frequent words, which, as shown in Table 1, can represent more than 98% of the vocabulary.

5.2 Training Details

We train three types of systems, purely *word-based*, purely *character-based*, and *hybrid*. Common to these architectures is a word-based NMT since the character-based systems are essentially word-based ones with longer sequences and the core of hybrid models is also a word-based NMT.

In training word-based NMT, we follow Luong et al. (2015a) to use the global attention mechanism together with similar hyperparameters: (a) deep LSTM models, 4 layers, 1024 cells, and 1024-dimensional embeddings, (b) uniform initialization of parameters in $[-0.1, 0.1]$, (c) 6-epoch training with plain SGD and a simple learning rate schedule – start with a learning rate of 1.0; after 4 epochs, halve the learning rate every 0.5 epoch, (d) mini-batches are of size 128 and shuf-

[3]For NIST BLEU, we first run `detokenizer.pl` and then use `mteval-v13a` to compute the scores as per WMT guideline. For chrF$_3$, we utilize the implementation here `https://github.com/rsennrich/subword-nmt`.

[4]Use `tokenizer.perl` in Moses with default settings.

	System	Vocab	Perplexity		BLEU	chrF$_3$
			w	c		
(a)	Best WMT'15, big data (Bojar and Tamchyna, 2015)	-	-	-	***18.8***	-
	Existing NMT					
(b)	RNNsearch + unk replace (Jean et al., 2015b)	200K	-	-	15.7	-
(c)	***Ensemble*** 4 models + unk replace (Jean et al., 2015b)	200K	-	-	18.3	-
	Our *word-based* NMT					
(d)	Base + attention + unk replace	50K	5.9	-	17.5	42.4
(e)	***Ensemble*** 4 models + unk replace	50K	-	-	18.4	43.9
	Our *character-based* NMT					
(f)	Base-512 (600-step backprop)	200	-	2.4	3.8	25.9
(g)	Base-512 + attention (600-step backprop)	200	-	1.6	17.5	***46.6***
(h)	Base-1024 + attention (300-step backprop)	200	-	1.9	15.7	41.1
	Our *hybrid* NMT					
(i)	Base + attention + same-path	10K	4.9	1.7	14.1	37.2
(j)	Base + attention + separate-path	10K	4.9	1.7	15.6	39.6
(k)	Base + attention + separate-path + 2-layer char	10K	4.7	1.6	***17.7***	44.1
(l)	Base + attention + separate-path + 2-layer char	50K	5.7	1.6	19.6	46.5
(m)	***Ensemble*** 4 models	50K	-	-	**20.7**	**47.5**

Table 2: **WMT'15 English-Czech results** – shown are the vocabulary sizes, perplexities, BLEU, and chrF$_3$ scores of various systems on *newstest2015*. Perplexities are listed under two categories, word (w) and character (c). **Best** and ***important*** results per metric are highlighed.

fled, (e) the gradient is rescaled whenever its norm exceeds 5, and (f) dropout is used with probability 0.2 according to (Pham et al., 2014). We now detail differences across the three architectures.

Word-based NMT – We constrain our source and target sequences to have a maximum length of 50 each; words that go past the boundary are ignored. The vocabularies are limited to the top $|V|$ most frequent words in both languages. Words not in these vocabularies are converted into <*unk*>. After translating, we will perform dictionary[5] lookup or identity copy for <*unk*> using the alignment information from the attention models. Such procedure is referred as the *unk replace* technique (Luong et al., 2015b; Jean et al., 2015a).

Character-based NMT – The source and target sequences at the character level are often about 5 times longer than their counterparts in the word-based models as we can infer from the statistics in Table 1. Due to memory constraint in GPUs, we limit our source and target sequences to a maximum length of 150 each, i.e., we backpropagate through at most 300 timesteps from the decoder to the encoder. With smaller 512-dimensional models, we can afford to have longer sequences with up to 600-step backpropagation.

Hybrid NMT – The *word*-level component uses the same settings as the purely word-based NMT. For the *character*-level source and target components, we experiment with both shallow and deep 1024-dimensional models of 1 and 2 LSTM layers. We set the weight α in Eq. (5) for our character-level loss to 1.0.

Training Time – It takes about 3 weeks to train a word-based model with $|V| = 50K$ and about 3 months to train a character-based model. Training and testing for the hybrid models are about 10-20% slower than those of the word-based models with the same vocabulary size.

5.3 Results

We compare our models with several strong systems. These include the winning entry in WMT'15, which was trained on a much larger amount of data, 52.6M parallel and 393.0M monolingual sentences (Bojar and Tamchyna, 2015).[6] In contrast, we merely use the provided parallel corpus of 15.8M sentences. For NMT, to the best

[5]Obtained from the alignment links produced by the Berkeley aligner (Liang et al., 2006) over the training corpus.

[6]This entry combines two independent systems, a phrase-based Moses model and a deep-syntactic transfer-based model. Additionally, there is an automatic post-editing system with hand-crafted rules to correct errors in morphological agreement and semantic meanings, e.g., loss of negation.

of our knowledge, (Jean et al., 2015b) has the best published performance on English-Czech.

As shown in Table 2, for a purely *word-based* approach, our single NMT model outperforms the best single model in (Jean et al., 2015b) by +1.8 points despite using a smaller vocabulary of only 50K words versus 200K words. Our ensemble system *(e)* slightly outperforms the best previous NMT system with 18.4 BLEU.

To our surprise, purely *character-based* models, though extremely slow to train and test, perform quite well. The 512-dimensional attention-based model *(g)* is best, surpassing the single word-based model in (Jean et al., 2015b) despite having much fewer parameters. It even outperforms most NMT systems on chrF$_3$ with 46.6 points. This indicates that this model translate words that closely but not exactly match the reference ones as evidenced in Section 6.3. We notice two interesting observations. First, attention is critical for character-based models to work as is obvious from the poor performance of the non-attentional model; this has also been shown in speech recognition (Chan et al., 2016). Second, long time-step backpropagation is more important as reflected by the fact that the larger 1024-dimensional model *(h)* with shorter backprogration is inferior to *(g)*.

Our *hybrid* models achieve the best results. At 10K words, we demonstrate that our *separate-path* strategy for the character-level target generation (§4.3) is effective, yielding an improvement of +1.5 BLEU points when comparing systems *(j)* vs. *(i)*. A *deeper* character-level architecture of 2 LSTM layers provides another significant boost of +2.1 BLEU. With 17.7 BLEU points, our hybrid system *(k)* has surpassed word-level NMT models.

When extending to 50K words, we further improve the translation quality. Our best single model, system *(l)* with 19.6 BLEU, is already better than all existing systems. Our ensemble model *(m)* further advances the SOTA result to ***20.7*** BLEU, outperforming the winning entry in the WMT'15 English-Czech translation task by a large margin of +1.9 points. Our ensemble model is also best in terms of chrF$_3$ with ***47.5*** points.

6 Analysis

This section first studies the effects of vocabulary sizes towards translation quality. We then analyze more carefully our character-level components by visualizing and evaluating rare word embeddings as well as examining sample translations.

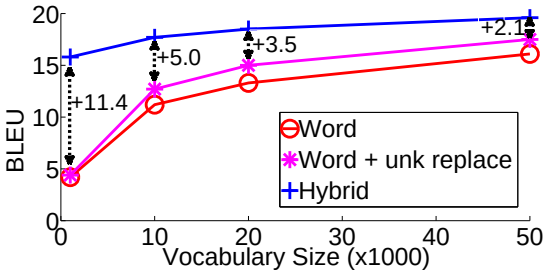


Figure 3: **Vocabulary size effect** – shown are the performances of different systems as we vary their vocabulary sizes. We highlight the improvements obtained by our hybrid models over word-based systems which already handle unknown words.

6.1 Effects of Vocabulary Sizes

As shown in Figure 3, our hybrid models offer large gains of +2.1-11.4 BLEU points over strong word-based systems which already handle unknown words. With only a small vocabulary, e.g., 1000 words, our hybrid approach can produce systems that are better than word-based models that possess much larger vocabularies. While it appears from the plot that gains diminish as we increase the vocabulary size, we argue that our hybrid models are still preferable since they understand word structures and can handle new complex words at test time as illustrated in Section 6.3.

6.2 Rare Word Embeddings

We evaluate the *source* character-level model by building representations for rare words and measuring how good these embeddings are.

Quantitatively, we follow Luong et al. (2013) in using the word similarity task, specifically on the *Rare Word* dataset, to judge the learned representations for complex words. The evaluation metric is the Spearman's correlation ρ between similarity scores assigned by a model and by human annotators. From the results in Table 3, we can see that source representations produced by our hybrid[7] models are significantly better than those of the word-based one. It is noteworthy that our deep recurrent character-level models can outperform the model of (Luong et al., 2013), which uses recursive neural networks and requires a complex morphological analyzer, by a large margin. Our performance is also competitive to the best Glove

[7]We look up the encoder embeddings for frequent words and build representations for rare word from characters.

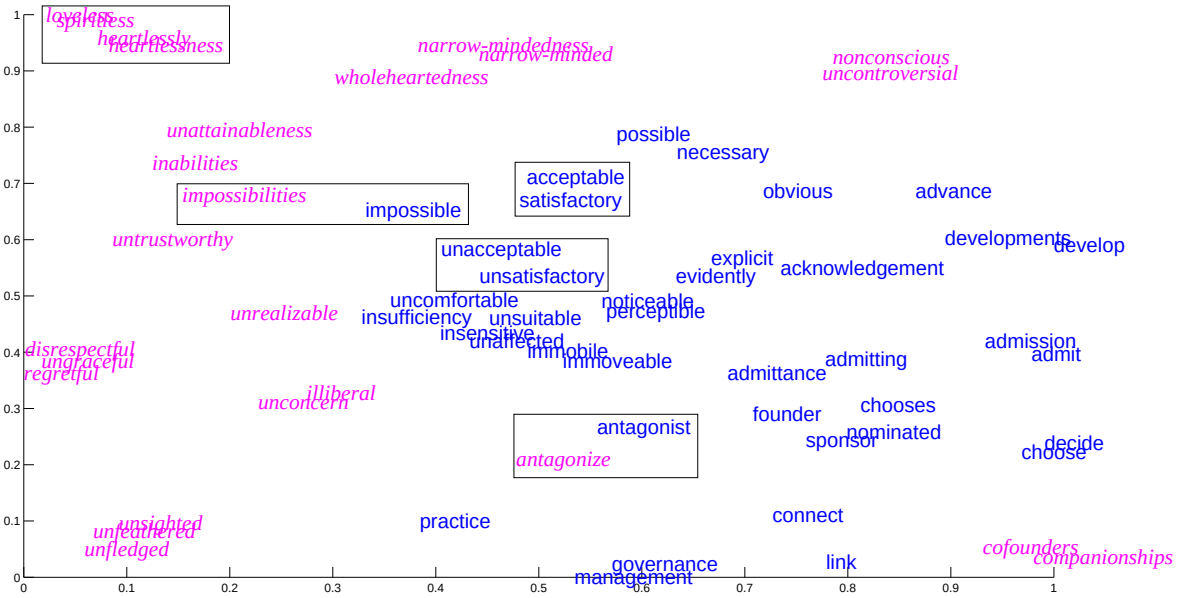


Figure 4: **Barnes-Hut-SNE visualization of source word representations** – shown are sample words from the *Rare Word* dataset. We differentiate two types of embeddings: frequent words in which encoder embeddings are looked up directly and *rare* words where we build representations from characters. Boxes highlight examples that we will discuss in the text. We use the hybrid model *(l)* in this visualization.

embeddings (Pennington et al., 2014) which were trained on a much larger dataset.

System		Size	$\|V\|$	ρ
(Luong et al., 2013)		1B	138K	34.4
Glove (Pennington et al., 2014)		6B	400K	38.1
		42B	400K	**47.8**
Our NMT models				
(d)	Word-based	0.3B	50K	20.4
(k)	Hybrid	0.3B	10K	42.4
(l)	Hybrid	0.3B	50K	***47.1***

Table 3: **Word similarity task** – shown are Spearman's correlation ρ on the *Rare Word* dataset of various models (with different vocab sizes $|V|$).

Qualitatively, we visualize embeddings produced by the hybrid model *(l)* for selected words in the Rare Word dataset. Figure 4 shows the two-dimensional representations of words computed by the Barnes-Hut-SNE algorithm (van der Maaten, 2013).[8] It is extremely interesting to observe that words are clustered together not only by the word structures but also by the meanings. For example, in the top-left box, the *character*-based representations for "loveless", "spiritless", "heartlessly", and "heartlessness" are nearby, but clearly separated into two groups. Similarly, in the center boxes, *word*-based embeddings of "acceptable", "satisfactory", "unacceptable", and "unsatisfactory", are close by but separated by meanings. Lastly, the remaining boxes demonstrate that our character-level models are able to build representations comparable to the word-based ones, e.g., "impossibilities" vs. "impossible" and "antagonize" vs. "antagonist". All of this evidence strongly supports that the source character-level models are useful and effective.

[8] We run Barnes-Hut-SNE algorithm over a set of 91 words, but filter out 27 words for displaying clarity.

6.3 Sample Translations

We show in Table 4 sample translations between various systems. In the first example, our hybrid model translates perfectly. The word-based model fails to translate "diagnosis" because the second <*unk*> was incorrectly aligned to the word "after". The character-based model, on the other hand, makes a mistake in translating names.

For the second example, the hybrid model surprises us when it can capture the long-distance reordering of "fifty years ago" and "před padesáti lety" while the other two models do not. The word-based model translates "Jr." inaccurately due to the incorrect alignment between the second <*unk*> and the word "said". The character-based model literally translates the name "King" into "král" which means "king".

Lastly, both the character-based and hybrid

1	source	The author ***Stephen Jay Gould*** died 20 years after ***diagnosis*** .
	human	Autor ***Stephen Jay Gould*** zemřel 20 let po ***diagnóze*** .
	word	Autor Stephen Jay <*unk*> zemřel 20 let po <*unk*> .
		Autor ***Stephen Jay Gould*** zemřel 20 let po **po** .
	char	Autor **Stepher Stepher** zemřel 20 let po ***diagnóze*** .
	hybrid	Autor <*unk*> <*unk*> <*unk*> zemřel 20 let po <*unk*>.
		Autor ***Stephen Jay Gould*** zemřel 20 let po ***diagnóze*** .
2	source	As the Reverend ***Martin Luther King Jr.*** said ***fifty years ago*** :
	human	Jak ***před padesáti lety*** řekl reverend ***Martin Luther King Jr*** . :
	word	Jak řekl reverend Martin <*unk*> King <*unk*> před padesáti lety :
		Jak řekl reverend ***Martin Luther King*** **řekl** před padesáti lety :
	char	Jako reverend ***Martin Luther*** **král říkal** před padesáti lety :
	hybrid	Jak před <*unk*> lety řekl <*unk*> Martin <*unk*> <*unk*> <*unk*> :
		Jak ***před padesáti lety*** řekl reverend ***Martin Luther King*** Jr. :
3	source	Her ***11-year-old*** daughter , ***Shani Bart*** , said it felt a " little bit ***weird*** " [..] back to school .
	human	Její ***jedenáctiletá*** dcera ***Shani Bartová*** prozradila , že " je to trochu ***zvláštní*** " [..] znova do školy .
	word	Její <*unk*> dcera <*unk*> <*unk*> řekla , že je to " trochu divné " , [..] vrací do školy .
		Její **11-year-old** dcera ***Shani*** **,** řekla , že je to " trochu divné " , [..] vrací do školy .
	char	Její ***jedenáctiletá*** dcera , ***Shani Bartová*** , říkala , že cítí trochu divně , [..] vrátila do školy .
	hybrid	Její <*unk*> dcera , <*unk*> <*unk*> , řekla , že cítí " trochu <*unk*> " , [..] vrátila do školy .
		Její ***jedenáctiletá*** dcera , **Graham** Bart , řekla , že cítí " trochu divný " , [..] vrátila do školy .

Table 4: **Sample translations on newstest2015** – for each example, we show the *source*, *human* translation, and translations of the following NMT systems: *word* model *(d)*, *char* model *(g)*, and *hybrid* model *(k)*. We show the translations before replacing <*unk*> tokens (if any) for the word-based and hybrid models. The following formats are used to highlight ***correct***, **wrong**, and close translation segments.

models impress us by their ability to translate compound words exactly, e.g., "11-year-old" and "jedenáctiletá"; whereas the identity copy strategy of the word-based model fails. Of course, our hybrid model does make mistakes, e.g., it fails to translate the name "Shani Bart". Overall, these examples highlight how challenging translating into Czech is and that being able to translate at the character level helps improve the quality.

7 Conclusion

We have proposed a novel *hybrid* architecture that combines the strength of both word- and character-based models. Word-level models are fast to train and offer high-quality translation; whereas, character-level models help achieve the goal of open vocabulary NMT. We have demonstrated these two aspects through our experimental results and translation examples.

Our best hybrid model has surpassed the performance of both the best word-based NMT system and the best non-neural model to establish a new state-of-the-art result for English-Czech translation in WMT'15 with 20.7 BLEU. Moreover, we have succeeded in replacing the standard unk replacement technique in NMT with our character-level components, yielding an improvement of $+2.1-11.4$ BLEU points. Our analysis has shown that our model has the ability to not only generate well-formed words for Czech, a highly inflected language with an enormous and complex vocabulary, but also build accurate representations for English source words.

Additionally, we have demonstrated the potential of purely character-based models in producing good translations; they have outperformed past word-level NMT models. For future work, we hope to be able to improve the memory usage and speed of purely character-based models.

Acknowledgments

This work was partially supported by NSF Award IIS-1514268 and by a gift from Bloomberg L.P. We thank Dan Jurafsky, Andrew Ng, and Quoc Le for earlier feedback on the work, as well as Sam Bowman, Ziang Xie, and Jiwei Li for their valuable comments on the paper draft. Lastly, we thank NVIDIA Corporation for the donation of Tesla K40 GPUs as well as Andrew Ng and his group for letting us use their computing resources.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *ICASSP*.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *EMNLP*.

Ondřej Bojar and Aleš Tamchyna. 2015. CUNI in WMT15: Chimera Strikes Again. In *WMT*.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell. In *ICASSP*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

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

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015a. On using very large target vocabulary for neural machine translation. In *ACL*.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015b. Montreal neural machine translation systems for WMT'15. In *WMT*.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *AAAI*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. In *ACL*.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by agreement. In *NAACL*.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luís Marujo, and Tiago Luís. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. In *EMNLP*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan Black. 2015b. Character-based neural machine translation.

Minh-Thang Luong and Christopher D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *IWSLT*.

Minh-Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *EMNLP*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *ICFHR*.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *WMT*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Laurens van der Maaten. 2013. Barnes-Hut-SNE. In *ICLR*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. abs/1409.2329.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*.

End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF

Xuezhe Ma and **Eduard Hovy**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
xuezhem@cs.cmu.edu, hovy@cmu.edu

Abstract

State-of-the-art sequence labeling systems traditionally require large amounts of task-specific knowledge in the form of hand-crafted features and data pre-processing. In this paper, we introduce a novel neutral network architecture that benefits from both word- and character-level representations automatically, by using combination of bidirectional LSTM, CNN and CRF. Our system is truly end-to-end, requiring no feature engineering or data pre-processing, thus making it applicable to a wide range of sequence labeling tasks. We evaluate our system on two data sets for two sequence labeling tasks — Penn Treebank WSJ corpus for part-of-speech (POS) tagging and CoNLL 2003 corpus for named entity recognition (NER). We obtain state-of-the-art performance on both datasets — 97.55% accuracy for POS tagging and 91.21% F1 for NER.

1 Introduction

Linguistic sequence labeling, such as part-of-speech (POS) tagging and named entity recognition (NER), is one of the first stages in deep language understanding and its importance has been well recognized in the natural language processing community. Natural language processing (NLP) systems, like syntactic parsing (Nivre and Scholz, 2004; McDonald et al., 2005; Koo and Collins, 2010; Ma and Zhao, 2012a; Ma and Zhao, 2012b; Chen and Manning, 2014; Ma and Hovy, 2015) and entity coreference resolution (Ng, 2010; Ma et al., 2016), are becoming more sophisticated, in part because of utilizing output information of POS tagging or NER systems.

Most traditional high performance sequence labeling models are linear statistical models, including Hidden Markov Models (HMM) and Conditional Random Fields (CRF) (Ratinov and Roth, 2009; Passos et al., 2014; Luo et al., 2015), which rely heavily on hand-crafted features and task-specific resources. For example, English POS taggers benefit from carefully designed word spelling features; orthographic features and external resources such as gazetteers are widely used in NER. However, such task-specific knowledge is costly to develop (Ma and Xia, 2014), making sequence labeling models difficult to adapt to new tasks or new domains.

In the past few years, non-linear neural networks with as input distributed word representations, also known as word embeddings, have been broadly applied to NLP problems with great success. Collobert et al. (2011) proposed a simple but effective feed-forward neutral network that independently classifies labels for each word by using contexts within a window with fixed size. Recently, recurrent neural networks (RNN) (Goller and Kuchler, 1996), together with its variants such as long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) and gated recurrent unit (GRU) (Cho et al., 2014), have shown great success in modeling sequential data. Several RNN-based neural network models have been proposed to solve sequence labeling tasks like speech recognition (Graves et al., 2013), POS tagging (Huang et al., 2015) and NER (Chiu and Nichols, 2015; Hu et al., 2016), achieving competitive performance against traditional models. However, even systems that have utilized distributed representations as inputs have used these to augment, rather than replace, hand-crafted features (e.g. word spelling and capitalization patterns). Their performance drops rapidly when the models solely depend on neural embeddings.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1064–1074,
Berlin, Germany, August 7-12, 2016.

In this paper, we propose a neural network architecture for sequence labeling. It is a truly end-to-end model requiring no task-specific resources, feature engineering, or data pre-processing beyond pre-trained word embeddings on unlabeled corpora. Thus, our model can be easily applied to a wide range of sequence labeling tasks on different languages and domains. We first use convolutional neural networks (CNNs) (LeCun et al., 1989) to encode character-level information of a word into its character-level representation. Then we combine character- and word-level representations and feed them into bi-directional LSTM (BLSTM) to model context information of each word. On top of BLSTM, we use a sequential CRF to jointly decode labels for the whole sentence. We evaluate our model on two linguistic sequence labeling tasks — POS tagging on Penn Treebank WSJ (Marcus et al., 1993), and NER on English data from the CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003). Our end-to-end model outperforms previous state-of-the-art systems, obtaining 97.55% accuracy for POS tagging and 91.21% F1 for NER. The contributions of this work are (i) proposing a novel neural network architecture for linguistic sequence labeling. (ii) giving empirical evaluations of this model on benchmark data sets for two classic NLP tasks. (iii) achieving state-of-the-art performance with this truly end-to-end system.

2 Neural Network Architecture

In this section, we describe the components (layers) of our neural network architecture. We introduce the neural layers in our neural network one-by-one from bottom to top.

2.1 CNN for Character-level Representation

Previous studies (Santos and Zadrozny, 2014; Chiu and Nichols, 2015) have shown that CNN is an effective approach to extract morphological information (like the prefix or suffix of a word) from characters of words and encode it into neural representations. Figure 1 shows the CNN we use to extract character-level representation of a given word. The CNN is similar to the one in Chiu and Nichols (2015), except that we use only character embeddings as the inputs to CNN, without character type features. A dropout layer (Srivastava et al., 2014) is applied before character embeddings are input to CNN.

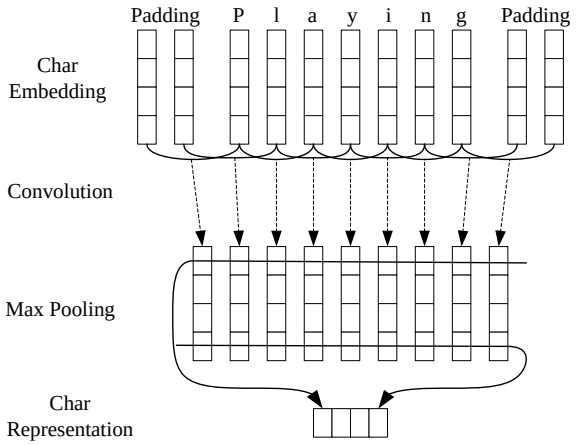


Figure 1: The convolution neural network for extracting character-level representations of words. Dashed arrows indicate a dropout layer applied before character embeddings are input to CNN.

2.2 Bi-directional LSTM

2.2.1 LSTM Unit

Recurrent neural networks (RNNs) are a powerful family of connectionist models that capture time dynamics via cycles in the graph. Though, in theory, RNNs are capable to capturing long-distance dependencies, in practice, they fail due to the gradient vanishing/exploding problems (Bengio et al., 1994; Pascanu et al., 2012).

LSTMs (Hochreiter and Schmidhuber, 1997) are variants of RNNs designed to cope with these gradient vanishing problems. Basically, a LSTM unit is composed of three multiplicative gates which control the proportions of information to forget and to pass on to the next time step. Figure 2 gives the basic structure of an LSTM unit.

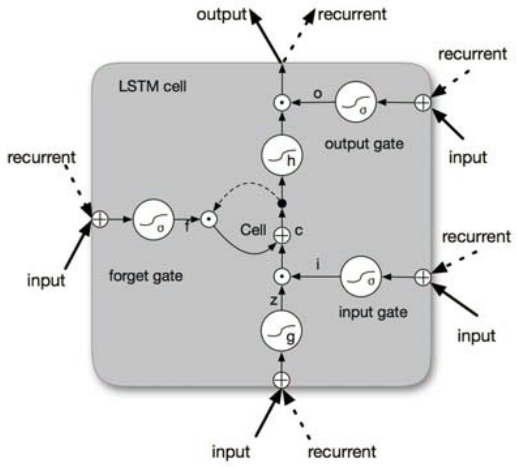


Figure 2: Schematic of LSTM unit.

Formally, the formulas to update an LSTM unit at time t are:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\boldsymbol{W}_i\mathbf{h}_{t-1} + \boldsymbol{U}_i\mathbf{x}_t + \boldsymbol{b}_i) \\
\mathbf{f}_t &= \sigma(\boldsymbol{W}_f\mathbf{h}_{t-1} + \boldsymbol{U}_f\mathbf{x}_t + \boldsymbol{b}_f) \\
\tilde{\mathbf{c}}_t &= \tanh(\boldsymbol{W}_c\mathbf{h}_{t-1} + \boldsymbol{U}_c\mathbf{x}_t + \boldsymbol{b}_c) \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\
\mathbf{o}_t &= \sigma(\boldsymbol{W}_o\mathbf{h}_{t-1} + \boldsymbol{U}_o\mathbf{x}_t + \boldsymbol{b}_o) \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product. $\mathbf{x}_t$ is the input vector (e.g. word embedding) at time t, and $\mathbf{h}_t$ is the hidden state (also called output) vector storing all the useful information at (and before) time t. $\boldsymbol{U}_i, \boldsymbol{U}_f, \boldsymbol{U}_c, \boldsymbol{U}_o$ denote the weight matrices of different gates for input $\mathbf{x}_t$, and $\boldsymbol{W}_i, \boldsymbol{W}_f, \boldsymbol{W}_c, \boldsymbol{W}_o$ are the weight matrices for hidden state $\mathbf{h}_t$. $\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_c, \boldsymbol{b}_o$ denote the bias vectors. It should be noted that we do not include peephole connections (Gers et al., 2003) in the our LSTM formulation.

2.2.2 BLSTM

For many sequence labeling tasks it is beneficial to have access to both past (left) and future (right) contexts. However, the LSTM's hidden state $\mathbf{h}_t$ takes information only from past, knowing nothing about the future. An elegant solution whose effectiveness has been proven by previous work (Dyer et al., 2015) is bi-directional LSTM (BLSTM). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are concatenated to form the final output.

2.3 CRF

For sequence labeling (or general structured prediction) tasks, it is beneficial to consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence. For example, in POS tagging an adjective is more likely to be followed by a noun than a verb, and in NER with standard BIO2 annotation (Tjong Kim Sang and Veenstra, 1999) I-ORG cannot follow I-PER. Therefore, we model label sequence jointly using a conditional random field (CRF) (Lafferty et al., 2001), instead of decoding each label independently.

Formally, we use $\mathbf{z} = \{\mathbf{z}_1, \cdots, \mathbf{z}_n\}$ to represent a generic input sequence where $\mathbf{z}_i$ is the input vector of the ith word. $\boldsymbol{y} = \{y_1, \cdots, y_n\}$ represents a generic sequence of labels for $\mathbf{z}$. $\mathcal{Y}(\mathbf{z})$ denotes the set of possible label sequences for $\mathbf{z}$. The probabilistic model for sequence CRF defines a family of conditional probability $p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b})$ over all possible label sequences $\boldsymbol{y}$ given $\mathbf{z}$ with the following form:

$$p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b}) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, \mathbf{z})}{\sum_{y' \in \mathcal{Y}(\mathbf{z})} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, \mathbf{z})}$$

where $\psi_i(y', y, \mathbf{z}) = \exp(\mathbf{W}_{y',y}^T \mathbf{z}_i + \mathbf{b}_{y',y})$ are potential functions, and $\mathbf{W}_{y',y}^T$ and $\mathbf{b}_{y',y}$ are the weight vector and bias corresponding to label pair (y', y), respectively.

For CRF training, we use the maximum conditional likelihood estimation. For a training set $\{(\mathbf{z}_i, \boldsymbol{y}_i)\}$, the logarithm of the likelihood (a.k.a. the log-likelihood) is given by:

$$L(\mathbf{W}, \mathbf{b}) = \sum_i \log p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b})$$

Maximum likelihood training chooses parameters such that the log-likelihood $L(\mathbf{W}, \mathbf{b})$ is maximized.

Decoding is to search for the label sequence $\boldsymbol{y}^*$ with the highest conditional probability:

$$\boldsymbol{y}^* = \operatorname*{argmax}_{y \in \mathcal{Y}(\mathbf{z})} p(\boldsymbol{y}|\mathbf{z}; \mathbf{W}, \mathbf{b})$$

For a sequence CRF model (only interactions between two successive labels are considered), training and decoding can be solved efficiently by adopting the Viterbi algorithm.

2.4 BLSTM-CNNs-CRF

Finally, we construct our neural network model by feeding the output vectors of BLSTM into a CRF layer. Figure 3 illustrates the architecture of our network in detail.

For each word, the character-level representation is computed by the CNN in Figure 1 with character embeddings as inputs. Then the character-level representation vector is concatenated with the word embedding vector to feed into the BLSTM network. Finally, the output vectors of BLSTM are fed to the CRF layer to jointly decode the best label sequence. As shown in Figure 3, dropout layers are applied on both the input and output vectors of BLSTM. Experimental results show that using dropout significantly

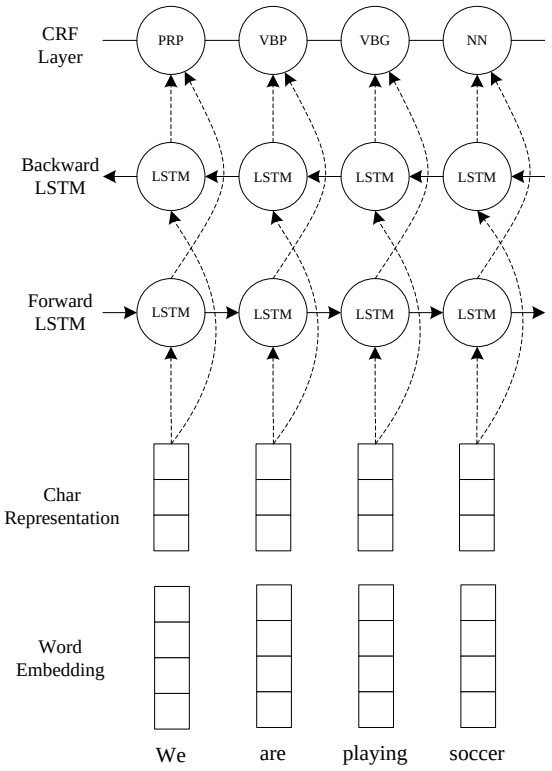


Figure 3: The main architecture of our neural network. The character representation for each word is computed by the CNN in Figure 1. Then the character representation vector is concatenated with the word embedding before feeding into the BLSTM network. Dashed arrows indicate dropout layers applied on both the input and output vectors of BLSTM.

improve the performance of our model (see Section 4.5 for details).

3 Network Training

In this section, we provide details about training the neural network. We implement the neural network using the Theano library (Bergstra et al., 2010). The computations for a single model are run on a GeForce GTX TITAN X GPU. Using the settings discussed in this section, the model training requires about 12 hours for POS tagging and 8 hours for NER.

3.1 Parameter Initialization

Word Embeddings. We use Stanford's publicly available GloVe 100-dimensional embeddings[1] trained on 6 billion words from Wikipedia and web text (Pennington et al., 2014)

We also run experiments on two other sets of published embeddings, namely Senna 50-dimensional embeddings[2] trained on Wikipedia and Reuters RCV-1 corpus (Collobert et al., 2011), and Google's Word2Vec 300-dimensional embeddings[3] trained on 100 billion words from Google News (Mikolov et al., 2013). To test the effectiveness of pretrained word embeddings, we experimented with randomly initialized embeddings with 100 dimensions, where embeddings are uniformly sampled from range $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$ where dim is the dimension of embeddings (He et al., 2015). The performance of different word embeddings is discussed in Section 4.4.

Character Embeddings. Character embeddings are initialized with uniform samples from $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$, where we set $dim = 30$.

Weight Matrices and Bias Vectors. Matrix parameters are randomly initialized with uniform samples from $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}]$, where r and c are the number of of rows and columns in the structure (Glorot and Bengio, 2010). Bias vectors are initialized to zero, except the bias $\mathbf{b}_f$ for the forget gate in LSTM , which is initialized to 1.0 (Jozefowicz et al., 2015).

3.2 Optimization Algorithm

Parameter optimization is performed with mini-batch stochastic gradient descent (SGD) with batch size 10 and momentum 0.9. We choose an initial learning rate of η_0 ($\eta_0 = 0.01$ for POS tagging, and 0.015 for NER, see Section 3.3.), and the learning rate is updated on each epoch of training as $\eta_t = \eta_0/(1 + \rho t)$, with decay rate $\rho = 0.05$ and t is the number of epoch completed. To reduce the effects of "gradient exploding", we use a gradient clipping of 5.0 (Pascanu et al., 2012). We explored other more sophisticated optimization algorithms such as AdaDelta (Zeiler, 2012), Adam (Kingma and Ba, 2014) or RMSProp (Dauphin et al., 2015), but none of them meaningfully improve upon SGD with momentum and gradient clipping in our preliminary experiments.

Early Stopping. We use early stopping (Giles, 2001; Graves et al., 2013) based on performance on validation sets. The "best" parameters appear at around 50 epochs, according to our experiments.

[1] http://nlp.stanford.edu/projects/glove/

[2] http://ronan.collobert.com/senna/

[3] https://code.google.com/archive/p/word2vec/

Layer	Hyper-parameter	POS	NER
CNN	window size	3	3
	number of filters	30	30
LSTM	state size	200	200
	initial state	0.0	0.0
	peepholes	no	no
Dropout	dropout rate	0.5	0.5
	batch size	10	10
	initial learning rate	0.01	0.015
	decay rate	0.05	0.05
	gradient clipping	5.0	5.0

Table 1: Hyper-parameters for all experiments.

Fine Tuning. For each of the embeddings, we fine-tune initial embeddings, modifying them during gradient updates of the neural network model by back-propagating gradients. The effectiveness of this method has been previously explored in sequential and structured prediction problems (Collobert et al., 2011; Peng and Dredze, 2015).

Dropout Training. To mitigate overfitting, we apply the dropout method (Srivastava et al., 2014) to regularize our model. As shown in Figure 1 and 3, we apply dropout on character embeddings before inputting to CNN, and on both the input and output vectors of BLSTM. We fix dropout rate at 0.5 for all dropout layers through all the experiments. We obtain significant improvements on model performance after using dropout (see Section 4.5).

3.3 Tuning Hyper-Parameters

Table 1 summarizes the chosen hyper-parameters for all experiments. We tune the hyper-parameters on the development sets by random search. Due to time constrains it is infeasible to do a random search across the full hyper-parameter space. Thus, for the tasks of POS tagging and NER we try to share as many hyper-parameters as possible. Note that the final hyper-parameters for these two tasks are almost the same, except the initial learning rate. We set the state size of LSTM to 200. Tuning this parameter did not significantly impact the performance of our model. For CNN, we use 30 filters with window length 3.

4 Experiments

4.1 Data Sets

As mentioned before, we evaluate our neural network model on two sequence labeling tasks: POS tagging and NER.

Dataset		WSJ	CoNLL2003
Train	SENT	38,219	14,987
	TOKEN	912,344	204,567
Dev	SENT	5,527	3,466
	TOKEN	131,768	51,578
Test	SENT	5,462	3,684
	TOKEN	129,654	46,666

Table 2: Corpora statistics. SENT and TOKEN refer to the number of sentences and tokens in each data set.

POS Tagging. For English POS tagging, we use the Wall Street Journal (WSJ) portion of Penn Treebank (PTB) (Marcus et al., 1993), which contains 45 different POS tags. In order to compare with previous work, we adopt the standard splits — section 0–18 as training data, section 19–21 as development data and section 22–24 as test data (Manning, 2011; Søgaard, 2011).

NER. For NER, We perform experiments on the English data from CoNLL 2003 shared task (Tjong Kim Sang and De Meulder, 2003). This data set contains four different types of named entities: *PERSON, LOCATION, ORGANIZATION*, and *MISC*. We use the BIOES tagging scheme instead of standard BIO2, as previous studies have reported meaningful improvement with this scheme (Ratinov and Roth, 2009; Dai et al., 2015; Lample et al., 2016).

The corpora statistics are shown in Table 2. We did not perform any pre-processing for data sets, leaving our system truly end-to-end.

4.2 Main Results

We first run experiments to dissect the effectiveness of each component (layer) of our neural network architecture by ablation studies. We compare the performance with three baseline systems — BRNN, the bi-direction RNN; BLSTM, the bi-direction LSTM, and BLSTM-CNNs, the combination of BLSTM with CNN to model character-level information. All these models are run using Stanford's GloVe 100 dimensional word embeddings and the same hyper-parameters as shown in Table 1. According to the results shown in Table 3, BLSTM obtains better performance than BRNN on all evaluation metrics of both the two tasks. BLSTM-CNN models significantly outperform the BLSTM model, showing that character-level representations are important for linguistic sequence labeling tasks. This is consistent with

	POS		NER					
	Dev	Test	Dev			Test		
Model	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	F1
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BRNN-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21

Table 3: Performance of our model on both the development and test sets of the two tasks, together with three baseline systems.

Model	**Acc.**
Giménez and Màrquez (2004)	97.16
Toutanova et al. (2003)	97.27
Manning (2011)	97.28
Collobert et al. (2011)‡	97.29
Santos and Zadrozny (2014)‡	97.32
Shen et al. (2007)	97.33
Sun (2014)	97.36
Søgaard (2011)	97.50
This paper	**97.55**

Table 4: POS tagging accuracy of our model on test data from WSJ proportion of PTB, together with top-performance systems. The neural network based models are marked with ‡.

Model	**F1**
Chieu and Ng (2002)	88.31
Florian et al. (2003)	88.76
Ando and Zhang (2005)	89.31
Collobert et al. (2011)‡	89.59
Huang et al. (2015)‡	90.10
Chiu and Nichols (2015)‡	90.77
Ratinov and Roth (2009)	90.80
Lin and Wu (2009)	90.90
Passos et al. (2014)	90.90
Lample et al. (2016)‡	90.94
Luo et al. (2015)	91.20
This paper	**91.21**

Table 5: NER F1 score of our model on test data set from CoNLL-2003. For the purpose of comparison, we also list F1 scores of previous top-performance systems. ‡ marks the neural models.

results reported by previous work (Santos and Zadrozny, 2014; Chiu and Nichols, 2015). Finally, by adding CRF layer for joint decoding we achieve significant improvements over BLSTM-CNN models for both POS tagging and NER on all metrics. This demonstrates that jointly decoding label sequences can significantly benefit the final performance of neural network models.

4.3 Comparison with Previous Work

4.3.1 POS Tagging

Table 4 illustrates the results of our model for POS tagging, together with seven previous top-performance systems for comparison. Our model significantly outperform Senna (Collobert et al., 2011), which is a feed-forward neural network model using capitalization and discrete suffix features, and data pre-processing. Moreover, our model achieves 0.23% improvements on accuracy over the "CharWNN" (Santos and Zadrozny, 2014), which is a neural network model based on Senna and also uses CNNs to model character-level representations. This demonstrates the effectiveness of BLSTM for modeling sequential data and the importance of joint decoding with structured prediction model.

Comparing with traditional statistical models, our system achieves state-of-the-art accuracy, obtaining 0.05% improvement over the previously best reported results by Søgaard (2011). It should be noted that Huang et al. (2015) also evaluated their BLSTM-CRF model for POS tagging on WSJ corpus. But they used a different splitting of the training/dev/test data sets. Thus, their results are not directly comparable with ours.

4.3.2 NER

Table 5 shows the F1 scores of previous models for NER on the test data set from CoNLL-2003 shared task. For the purpose of comparison, we list their results together with ours. Similar to the observations of POS tagging, our model achieves significant improvements over Senna and the other three neural models, namely the LSTM-CRF proposed by Huang et al. (2015), LSTM-CNNs pro-

Embedding	Dimension	POS	NER
Random	100	97.13	80.76
Senna	50	97.44	90.28
Word2Vec	300	97.40	84.91
GloVe	100	**97.55**	**91.21**

Table 6: Results with different choices of word embeddings on the two tasks (accuracy for POS tagging and F1 for NER).

posed by Chiu and Nichols (2015), and the LSTM-CRF by Lample et al. (2016). Huang et al. (2015) utilized discrete spelling, POS and context features, Chiu and Nichols (2015) used character-type, capitalization, and lexicon features, and all the three model used some task-specific data pre-processing, while our model does not require any carefully designed features or data pre-processing. We have to point out that the result (90.77%) reported by Chiu and Nichols (2015) is incomparable with ours, because their final model was trained on the combination of the training and development data sets[4].

To our knowledge, the previous best F1 score (91.20)[5] reported on CoNLL 2003 data set is by the joint NER and entity linking model (Luo et al., 2015). This model used many hand-crafted features including stemming and spelling features, POS and chunks tags, WordNet clusters, Brown Clusters, as well as external knowledge bases such as Freebase and Wikipedia. Our end-to-end model slightly improves this model by 0.01%, yielding a state-of-the-art performance.

4.4 Word Embeddings

As mentioned in Section 3.1, in order to test the importance of pretrained word embeddings, we performed experiments with different sets of publicly published word embeddings, as well as a random sampling method, to initialize our model. Table 6 gives the performance of three different word embeddings, as well as the randomly sampled one. According to the results in Table 6, models using pretrained word embeddings obtain a significant improvement as opposed to the ones using random embeddings. Comparing the two tasks, NER relies more heavily on pretrained embeddings than POS tagging. This is consistent with results reported by previous work (Collobert et al., 2011; Huang et al., 2015; Chiu and Nichols, 2015).

For different pretrained embeddings, Stanford's GloVe 100 dimensional embeddings achieve best results on both tasks, about 0.1% better on POS accuracy and 0.9% better on NER F1 score than the Senna 50 dimensional one. This is different from the results reported by Chiu and Nichols (2015), where Senna achieved slightly better performance on NER than other embeddings. Google's Word2Vec 300 dimensional embeddings obtain similar performance with Senna on POS tagging, still slightly behind GloVe. But for NER, the performance on Word2Vec is far behind GloVe and Senna. One possible reason that Word2Vec is not as good as the other two embeddings on NER is because of vocabulary mismatch — Word2Vec embeddings were trained in case-sensitive manner, excluding many common symbols such as punctuations and digits. Since we do not use any data pre-processing to deal with such common symbols or rare words, it might be an issue for using Word2Vec.

	POS			NER		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**
No	98.46	97.06	97.11	99.97	93.51	89.25
Yes	97.86	97.46	97.55	99.63	94.74	91.21

Table 7: Results with and without dropout on two tasks (accuracy for POS tagging and F1 for NER).

	POS		NER	
	Dev	Test	Dev	Test
IV	127,247	125,826	4,616	3,773
OOTV	2,960	2,412	1,087	1,597
OOEV	659	588	44	8
OOBV	902	828	195	270

Table 8: Statistics of the partition on each corpus. It lists the number of tokens of each subset for POS tagging and the number of entities for NER.

4.5 Effect of Dropout

Table 7 compares the results with and without dropout layers for each data set. All other hyper-parameters remain the same as in Table 1. We observe a essential improvement for both the two tasks. It demonstrates the effectiveness of dropout in reducing overfitting.

[4]We run experiments using the same setting and get 91.37% F1 score.

[5]Numbers are taken from the Table 3 of the original paper (Luo et al., 2015). While there is clearly inconsistency among the precision (91.5%), recall (91.4%) and F1 scores (91.2%), it is unclear in which way they are incorrect.

	POS							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	97.57	**93.75**	90.29	80.27	97.55	**93.45**	90.14	80.07
LSTM-CNN-CRF	**97.68**	93.65	**91.05**	**82.71**	**97.77**	93.16	**90.65**	**82.49**
	NER							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	94.83	87.28	96.55	82.90	90.07	89.45	100.00	78.44
LSTM-CNN-CRF	**96.49**	**88.63**	**97.67**	**86.91**	**92.14**	**90.73**	100.00	**80.60**

Table 9: Comparison of performance on different subsets of words (accuracy for POS and F1 for NER).

4.6 OOV Error Analysis

To better understand the behavior of our model, we perform error analysis on Out-of-Vocabulary words (OOV). Specifically, we partition each data set into four subsets — in-vocabulary words (IV), out-of-training-vocabulary words (OOTV), out-of-embedding-vocabulary words (OOEV) and out-of-both-vocabulary words (OOBV). A word is considered IV if it appears in both the training and embedding vocabulary, while OOBV if neither. OOTV words are the ones do not appear in training set but in embedding vocabulary, while OOEV are the ones do not appear in embedding vocabulary but in training set. For NER, an entity is considered as OOBV if there exists at lease one word not in training set and at least one word not in embedding vocabulary, and the other three subsets can be done in similar manner. Table 8 informs the statistics of the partition on each corpus. The embedding we used is Stanford's GloVe with dimension 100, the same as Section 4.2.

Table 9 illustrates the performance of our model on different subsets of words, together with the baseline LSTM-CNN model for comparison. The largest improvements appear on the OOBV subsets of both the two corpora. This demonstrates that by adding CRF for joint decoding, our model is more powerful on words that are out of both the training and embedding sets.

5 Related Work

In recent years, several different neural network architectures have been proposed and successfully applied to linguistic sequence labeling such as POS tagging, chunking and NER. Among these neural architectures, the three approaches most similar to our model are the BLSTM-CRF model proposed by Huang et al. (2015), the LSTM-CNNs model by Chiu and Nichols (2015) and the BLSTM-CRF by Lample et al. (2016).

Huang et al. (2015) used BLSTM for word-level representations and CRF for jointly label decoding, which is similar to our model. But there are two main differences between their model and ours. First, they did not employ CNNs to model character-level information. Second, they combined their neural network model with handcrafted features to improve their performance, making their model not an end-to-end system. Chiu and Nichols (2015) proposed a hybrid of BLSTM and CNNs to model both character- and word-level representations, which is similar to the first two layers in our model. They evaluated their model on NER and achieved competitive performance. Our model mainly differ from this model by using CRF for joint decoding. Moreover, their model is not truly end-to-end, either, as it utilizes external knowledge such as character-type, capitalization and lexicon features, and some data preprocessing specifically for NER (e.g. replacing all sequences of digits 0-9 with a single "0"). Recently, Lample et al. (2016) proposed a BLSTM-CRF model for NER, which utilized BLSTM to model both the character- and word-level information, and use data pre-processing the same as Chiu and Nichols (2015). Instead, we use CNN to model character-level information, achieving better NER performance without using any data preprocessing.

There are several other neural networks previously proposed for sequence labeling. Labeau et al. (2015) proposed a RNN-CNNs model for German POS tagging. This model is similar to the LSTM-CNNs model in Chiu and Nichols (2015), with the difference of using vanila RNN instead of LSTM. Another neural architecture employing

CNN to model character-level information is the "CharWNN" architecture (Santos and Zadrozny, 2014) which is inspired by the feed-forward network (Collobert et al., 2011). CharWNN obtained near state-of-the-art accuracy on English POS tagging (see Section 4.3 for details). A similar model has also been applied to Spanish and Portuguese NER (dos Santos et al., 2015) Ling et al. (2015) and Yang et al. (2016) also used BSLTM to compose character embeddings to word's representation, which is similar to Lample et al. (2016). Peng and Dredze (2016) Improved NER for Chinese Social Media with Word Segmentation.

6 Conclusion

In this paper, we proposed a neural network architecture for sequence labeling. It is a truly end-to-end model relying on no task-specific resources, feature engineering or data pre-processing. We achieved state-of-the-art performance on two linguistic sequence labeling tasks, comparing with previously state-of-the-art systems.

There are several potential directions for future work. First, our model can be further improved by exploring multi-task learning approaches to combine more useful and correlated information. For example, we can jointly train a neural network model with both the POS and NER tags to improve the intermediate representations learned in our network. Another interesting direction is to apply our model to data from other domains such as social media (Twitter and Weibo). Since our model does not require any domain- or task-specific knowledge, it might be effortless to apply it to these domains.

Acknowledgements

This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DARPA.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a cpu and gpu math expression compiler. In *Proceedings of the Python for scientific computing conference (SciPy)*, volume 4, page 3. Austin, TX.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP-2014*, pages 740–750, Doha, Qatar, October.

Hai Leong Chieu and Hwee Tou Ng. 2002. Named entity recognition: a maximum entropy approach using global information. In *Proceedings of CoNLL-2003*, pages 1–7.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Syntax, Semantics and Structure in Statistical Translation*, page 103.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Hong-Jie Dai, Po-Ting Lai, Yung-Chun Chang, and Richard Tzong-Han Tsai. 2015. Enhancing of chemical compound and drug name recognition using representative tag scheme and fine-grained tokenization. *Journal of cheminformatics*, 7(S1):1–10.

Yann N Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. 2015. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv preprint arXiv:1502.04390*.

Cıcero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. In *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 25.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL-2015 (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July.

Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. 2003. Named entity recognition through classifier combination. In *Proceedings of HLT-NAACL-2003*, pages 168–171.

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

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

Rich Caruana Steve Lawrence Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, volume 13, page 402. MIT Press.

Jesús Giménez and Lluís Màrquez. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *In Proceedings of LREC-2004*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Neural Networks, 1996., IEEE International Conference on*, volume 1, pages 347–352. IEEE.

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP-2013*, pages 6645–6649. IEEE.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1026–1034.

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

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard H. Hovy, and Eric P. Xing. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of ACL-2016*, Berlin, Germany, August.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL-2010*, pages 1–11, Uppsala, Sweden, July.

Matthieu Labeau, Kevin Löser, Alexandre Allauzen, and Rue John von Neumann. 2015. Non-lexical neural architecture for fine-grained pos tagging. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 232–237.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-2001*, volume 951, pages 282–289.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-2016*, San Diego, California, USA, June.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of ACL-2009*, pages 1030–1038.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP-2015*, pages 1520–1530, Lisbon, Portugal, September.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint entity recognition and disambiguation. In *Proceedings of EMNLP-2015*, pages 879–888, Lisbon, Portugal, September.

Xuezhe Ma and Eduard Hovy. 2015. Efficient inner-to-outer greedy algorithm for higher-order labeled dependency parsing. In *Proceedings of the EMNLP-2015*, pages 1322–1328, Lisbon, Portugal, September.

Xuezhe Ma and Fei Xia. 2014. Unsupervised dependency parsing with transferring distribution via parallel guidance and entropy regularization. In *Proceedings of ACL-2014*, pages 1337–1348, Baltimore, Maryland, June.

Xuezhe Ma and Hai Zhao. 2012a. Fourth-order dependency parsing. In *Proceedings of COLING 2012: Posters*, pages 785–796, Mumbai, India, December.

Xuezhe Ma and Hai Zhao. 2012b. Probabilistic models for high-order projective dependency parsing. *Technical Report, arXiv:1502.04174*.

Xuezhe Ma, Zhengzhong Liu, and Eduard Hovy. 2016. Unsupervised ranking model for entity coreference resolution. In *Proceedings of NAACL-2016*, San Diego, California, USA, June.

Christopher D Manning. 2011. Part-of-speech tagging from 97% to 100%: is it time for some linguistics? In *Computational Linguistics and Intelligent Text Processing*, pages 171–189. Springer.

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

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL-2005*, pages 91–98, Ann Arbor, Michigan, USA, June 25-30.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of ACL-2010*, pages 1396–1411, Uppsala, Sweden, July. Association for Computational Linguistics.

Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *Proceedings of COLING-2004*, pages 64–70, Geneva, Switzerland, August 23-27.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. In *Proceedings of CoNLL-2014*, pages 78–86, Ann Arbor, Michigan, June.

Nanyun Peng and Mark Dredze. 2015. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of EMNLP-2015*, pages 548–554, Lisbon, Portugal, September.

Nanyun Peng and Mark Dredze. 2016. Improving named entity recognition for chinese social media with word segmentation representation learning. In *Proceedings of ACL-2016*, Berlin, Germany, August.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP-2014*, pages 1532–1543, Doha, Qatar, October.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-2009*, pages 147–155.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of ICML-2014*, pages 1818–1826.

Libin Shen, Giorgio Satta, and Aravind Joshi. 2007. Guided learning for bidirectional sequence classification. In *Proceedings of ACL-2007*, volume 7, pages 760–767.

Anders Søgaard. 2011. Semi-supervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 48–52, Portland, Oregon, USA, June.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Xu Sun. 2014. Structure regularization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 2402–2410.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003 - Volume 4*, pages 142–147, Stroudsburg, PA, USA.

Erik F. Tjong Kim Sang and Jorn Veenstra. 1999. Representing text chunks. In *Proceedings of EACL'99*, pages 173–179. Bergen, Norway.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of NAACL-HLT-2003, Volume 1*, pages 173–180.

Zhilin Yang, Ruslan Salakhutdinov, and William Cohen. 2016. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Off-topic Response Detection for Spontaneous Spoken English Assessment

Andrey Malinin, Rogier C. Van Dalen, Yu Wang, Kate M. Knill, Mark J. F. Gales
University of Cambridge, Department of Engineering
Trumpington St, Cambridge CB2 1PZ, UK
{am969, yw396, kate.knill, mjfg}@eng.cam.ac.uk

Abstract

Automatic spoken language assessment systems are becoming increasingly important to meet the demand for English second language learning. This is a challenging task due to the high error rates of, even state-of-the-art, non-native speech recognition. Consequently current systems primarily assess fluency and pronunciation. However, content assessment is essential for full automation. As a first stage it is important to judge whether the speaker responds on topic to test questions designed to elicit spontaneous speech. Standard approaches to off-topic response detection assess similarity between the response and question based on bag-of-words representations. An alternative framework based on Recurrent Neural Network Language Models (RNNLM) is proposed in this paper. The RNNLM is adapted to the topic of each test question. It learns to associate example responses to questions with points in a topic space constructed using these example responses. Classification is done by ranking the topic-conditional posterior probabilities of a response. The RNNLMs associate a broad range of responses with each topic, incorporate sequence information and scale better with additional training data, unlike standard methods. On experiments conducted on data from the Business Language Testing Service (BULATS) this approach outperforms standard approaches.

1 Introduction

As English has become the global *lingua franca*, there is growing demand worldwide for assessment of English as a second language (Seidlhofer, 2005). To assess spoken communication, spontaneous speech is typically elicited through a series of questions such as 'describe the photo' or 'plan a meeting'. Grades are awarded based on a candidate's responses.

Automatic assessment systems are becoming attractive as they allow second language assessment programmes to economically scale their operations while decreasing throughput time and provide testing on demand. Features for automatic graders are derived from the audio and from hypotheses produced by automatic speech recognition (ASR) systems. The latter is highly errorful due to the large variability in the input speech; disfluencies common to spontaneous speech, non-native accents and pronunciations. Current systems, such as ETS' *SpeechRater* (Zechner et al., 2009) and Pearson's *AZELLA* (Metallinou and Cheng, 2014), primarily assess pronunciation and fluency. Although these are clearly indicative of spoken language ability, full assessment of spoken communication requires judgement of high-level content and communication skills, such as response construction and relevance. The first stage of this is to assess whether the responses are off-topic, that is, has the candidate misunderstood the question and/or memorised a response.

While there has been little work done on detecting off-topic responses for spoken language assessment, detection of off-topic responses and content assessment has been studied for essay assessment. One approach for essay content assessment uses features based on semantic similarity metrics between vector space representations of responses. Common vector representations include lexical Vector Space Models and Latent Semantic Analysis (LSA) (Yannakoudakis, 2013). This approach was first applied to spoken assess-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1075–1084,
Berlin, Germany, August 7-12, 2016.

ment in (Xie et al., 2012) and then in (Evanini et al., 2013). Following this, (Yoon and Xie, 2014) investigated the detection of responses for which an automatic assessment system will have difficulty in assigning a valid score, of which off-topic responses are a specific type. A decision tree classifier is used with features based on cosine similarity between a test response and *tf-idf* vectors of both aggregate example responses and questions, as well as pronunciation and fluency. In (Evanini and Wang, 2014) text reuse and plagiarism in spoken responses are detected using a decision tree classifier based on vector similarity and lexical matching features which compare a response to a set of example 'source texts' . This task is similar to off-topic response detection in that it is based on comparing a test response to example responses. Thus, a standard approach to off-topic response detection would be based on measuring the similarity between vector representations of a spoken response and the test question. A major deficiency of this approach is that it is based on bag-of-words vector representations, which loses information about the sequential nature of speech, which is important to evaluating response construction and relevance. Additionally, adapting the approach to model a range of responses for each topic causes classification time to scale poorly with training data size and the number of questions.

To address these issues a general off-topic content detection framework based on topic adapted Recurrent Neural Network language models (RNNLM) has been developed and applied to off-topic response detection for spoken language assessment. This framework uses example responses to test questions in training of the language model and construction of the topic-space. The RNNLM learns to associate the example responses with points in the topic-space. Classification is done by ranking the topic-conditional posterior probabilities of a response. The advantage of this approach is that sequence information can be taken into account and broad ranges of responses can be associated with each topic without affecting classifcation speed. Two topic vector representations are investigated: Latent Dirichlet Allocation (LDA) (Blei et al., 2003; Griffiths and Steyvers, 2004) and Latent Semantic Analysis (LSA) (Landauer et al., 1998). They are compared to standard approaches on data from the Cambridge Business English (BULATS) exam.

The rest of this paper is structured as follows: Section 2 discusses the RNNLM adaptation and topic spaces; Section 3 discusses approaches to topic detection; Section 4 presents data sets and experimental infrastructure; Section 5 analyzes experimental results; Section 6 concludes the paper.

2 Topic Adapted RNNLMs

2.1 RNNLM Architecture

A statistical language model is used to model the semantic and syntactic information in text in the form of a probability distribution over word sequences. It assigns a probability to a word sequence $\mathbf{w} = \{w_0, w_1, \cdots, w_L\}$ as follows:

$$\mathtt{P}(w_i|w_{i-1}, \cdots, w_0) = \mathtt{P}(w_i|\mathbf{h}_0^{i-1}) \quad (1)$$

$$\mathtt{P}(\mathbf{w}) = \prod_{i=1}^{L} \mathtt{P}(w_i|\mathbf{h}_0^{i-1}) \quad (2)$$

where w_0 is the start of sentence symbol <s>. In this work a language model is trained to model example responses to questions on a spoken language assessment test. $\mathtt{P}(w_i|\mathbf{h}_0^{i-1})$ can be estimated by a number of approaches, most notably N-grams and Recurrent Neural Networks (Mikolov et al., 2010).

Recurrent Neural Network language models (RNNLMs) (Figure 1) (Mikolov, 2012) are a variable context length language model, capable of representing the entire context efficiently, unlike N-grams. RNNLMs represent the full untruncated history $\mathbf{h}_0^{i-1} = \{w_{i-1}, \cdots, w_0\}$ for word w_i as the hidden layer $\mathbf{s}_{i-1}$, a form of short-term memory, whose representation is learned from the data. Words and phrases are represented in a continuous space, which gives RNNLMs greater generalization abilities than other language models, such as N-grams.

RNNLMs can be adapted by adding a feature vector $\mathbf{f}$ which represents information absent from the RNN (Mikolov and Zweig, 2012). In this work, the vector representation of a spoken language test question topic $\mathbf{f}_q$ is used for the context vector $\mathbf{f}$. Architecturally, a context adapted RNNLM is described by equations 3-5. $\mathtt{e}(x)$ and $\mathtt{g}(x)$ are element-wise sigmoid and softmax acti-

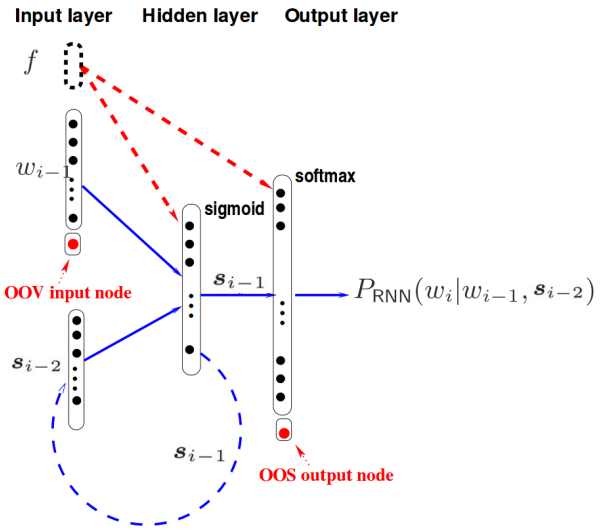


Figure 1: Context adapted RNN language model

vation functions.

$$\mathrm{P}(w_i|\mathbf{h}_0^{i-1}, \mathbf{f}) = \mathrm{P}_{\text{RNN}}(w_i|w_{i-1}, \mathbf{s}_{i-2}, \mathbf{f}) \quad (3)$$

$$\mathrm{P}_{\text{RNN}}(w_i|w_{i-1}, \mathbf{s}_{i-2}, \mathbf{f}) = \mathrm{g}(\mathbf{V}\mathbf{s}_{i-1} + \mathbf{H}\mathbf{f}) \quad (4)$$

$$\mathbf{s}_{i-1} = \mathrm{e}(\mathbf{U}w_{i-1} + \mathbf{W}\mathbf{s}_{i-2} + \mathbf{G}\mathbf{f}) \quad (5)$$

Through the process of adaptation the RNNLM learns to associate particular types of responses with particular topics, thereby becoming more discriminative. Thus, a sentence's topic-conditional probability $\mathrm{P}_{\text{RNN}}(\mathbf{w}|\mathbf{f}_q)$ will be higher if it corresponds to the topic q than if it does not.

2.2 Example Response Based Topic Space

In order for the topic vectors $\mathbf{f}_q$ to be informative they must span the space of all question topics in the test. Thus a topic space needs to be defined. Example responses, which are necessary to train the RNNLM, are used to define a topic space because typical responses to a question will be definitive of the question's topic. Multiple example responses to a particular question are merged into one aggregate response to capture a broad range of response variations and increase the robustness of the vector representation estimation.

By default a topic t is defined for each question q. However, multi-part questions are common, where candidates are given a scenario such as providing tourist information in which individual questions ask about food, hotels or sights. Since the underlying topic is related this can confuse a classifier. The responses for all these related questions could be merged to form a single aggregate vector, but the statistics of the responses to each question can be sufficiently different that less distinct topics are formed. Instead the aggregrate example responses for each question are assigned the same topic label. Thus, a mapping between questions and topics and its inverse is introduced:

$$\mathcal{M} : q \rightarrow t \quad (6)$$

$$\mathcal{M}_t^{-1} : \{q \in Q | \mathcal{M}(q) = t\} \quad (7)$$

A vector representation of a question topic is computed using the aggregate example responses. As mentioned in Section 1, two common representations are LDA and LSA; both are investigated in this work.

LDA is a generative model which allows documents to be modelled as distributions over latent topics $z \in Z$. Each latent topic z is described by a multinomial distribution over words $\mathrm{P}(w_i|z)$, and each word in a document is attributed to a particular latent topic (Blei et al., 2003). Thus, the adaptation vector $\mathbf{f}_\mathbf{w}$ represents a vector of posterior probabilities over latent topics for word sequence $\mathbf{w}$:

$$\mathbf{f}_\mathbf{w} = [\mathrm{P}(z = 1|\mathbf{w}), \cdots, \mathrm{P}(z = K|\mathbf{w})]^T \quad (8)$$

$$\mathrm{P}(z = k|\mathbf{w}) = \frac{\sum_{i=1}^{N} \delta(z_{w_i} = k)}{N} \quad (9)$$

LDA was found to perform better for RNNLM adaptation than other representations in (Mikolov and Zweig, 2012; Chen et al., 2015).

LSA (Landauer et al., 1998) is a popular representation for information retrieval tasks. A word-document matrix $\mathbf{F}$ is constructed using example responses and each word is weighted by its term frequency-inverse document frequency (TF-IDF). Then a low-rank approximation $\mathbf{F}_k$ is computed using Singular Value Decomposition (SVD):

$$\mathbf{F}_k = \mathbf{U}_k \mathbf{\Sigma}_k \mathbf{V}_k^T \quad (10)$$

$$\mathbf{F}_k = [\mathbf{f}_1, \cdots, \mathbf{f}_Q]^T \quad (11)$$

$$\mathbf{f}_\mathbf{w} = \Sigma_k^{-1} \mathbf{U}_k^T \mathbf{f}_{tfidf} \quad (12)$$

Only the k largest singular values of the singular value matrix Σ are kept. $\mathbf{F}_k$ is a representation of the data which retains only the k most significant factors of variation. Each row is a topic vector $\mathbf{f}_q$. New vectors $\mathbf{f}_\mathbf{w}$ for test responses can be projected into the LSA space via equation 12, where $\mathbf{f}_{tfidf}$ is the TF-IDF weighted bag-of-word representation of a test response.

3 Topic Detection and Features

This section discusses the standard, vector similarity feature-based, and the proposed topic adapted RNNLM approaches for topic detection.

3.1 Vector Similarity Features

The essence of the standard approach to topic detection is to assess the semantic distance $\mathtt{D}_{\mathtt{sem}}$ between the test response $\mathbf{w}$ and the aggregate example response $\mathbf{w}_q$ by approximating it using a vector distance metric $\mathtt{D}_{\mathtt{vec}}$ between vector representations of the response $\mathbf{f}_{\mathbf{w}}$ and the topic $\mathbf{f}_q$. Classification is done by selecting the topic closest to the test response:

$$\hat{t}_{\mathbf{w}} = \mathcal{M}\big(\arg\min_q \{\mathtt{D}_{\mathtt{sem}}(\mathbf{w}, \mathbf{w}_q)\}\big) \quad (13)$$

$$\mathtt{D}_{\mathtt{sem}}(\mathbf{w}, \mathbf{w}_q) \approx \mathtt{D}_{\mathtt{vec}}(\mathbf{f}_{\mathbf{w}}, \mathbf{f}_q) \quad (14)$$

The selection of an appropriate distance metric $\mathtt{D}_{\mathtt{vec}}(\mathbf{f}_{\mathbf{w}}, \mathbf{f}_q)$ can have a large effect on the classification outcome. A common metric used in topic classification and information retrieval is cosine similarity, which measures the cosine of the angle between two vectors. A distance metric based on this, *cosine distance*, can be defined as:

$$\mathtt{D}_{\mathtt{cos}}(\mathbf{f}_{\mathbf{w}}, \mathbf{f}_q) = 1 - \frac{\mathbf{f}_{\mathbf{w}}{}^{\mathsf{T}} \mathbf{f}_q}{|\mathbf{f}_{\mathbf{w}}||\mathbf{f}_q|} \quad (15)$$

While topics are robustly defined, this approach fails to capture the range of responses which can be given to a question. A different approach would be to maintain a separate point in this topic space for every example response. This retains the robust topic definition while allowing each topic to be represented by a cloud of points in topic space, thereby capturing a range of responses which can be given. A K-nearest neighbour (KNN) classifier can be used to detect the response topic by computing distances of the test response to each of the training points in topic space. However, classification may become impractical for large data sets, as the number of response points scales with the size of the training data. Low-cost distance measures, such as cosine distance, allow this approach to be used on large data sets before it becomes computationally infeasible. This approach is used as the baseline for comparison with the proposed RNNLM based method. For multi-part questions, topic vectors relating to the same overall topic are simply given the same topic label.

The classification rate can be improved by taking the top N $\hat{\mathbf{t}}_N = \{\hat{t}_1, \cdots, \hat{t}_N\}$ results into account. The KNN classifier can be modified to yield the N-best classification by removing all training points from the 1-best class from the KNN classifier and re-running the classification to get the 2-best results, and so on.

One of the main deficiencies of methods based on computing distances between vector representations is that commonly used representations, such as LSA and LDA, ignore word-order in documents, thereby throwing away information which is potentially useful for topic classification. Additionally, if any of the test or example response utterances are short, then their topic vector representations may not be robustly estimated.

3.2 Topic Adapted RNNLM Framework

The RNNLM based approach to topic detection is based on different principles. By combining equations 2 and 3 the log-probability $\mathtt{L}(q)$ of a response sentence given a particular topic vector $\mathtt{P}_{\mathtt{RNN}}(\mathbf{w}|\mathbf{f}_q)$ is computed. For each response $\mathbf{w}$ in the test set $\mathtt{L}(q)$ is computed (equation 16) for all topic vectors $\mathbf{f}_q$. $\mathtt{L}(q)$ is calculated using equation 17 for multi-part questions with responses $\mathbf{w}_p$ where $p \in t$. Classification is done by ranking log-probability $\mathtt{L}(q)$ for an utterance $\mathbf{w}$ and $\mathtt{L}(q)$ for all $q \in M_t^{-1}$ are averaged (equation 18).

$$\mathtt{L}(q) = \begin{cases} \log[\mathtt{P}_{\mathtt{RNN}}(\mathbf{w}|\mathbf{f}_q)] & (16) \\ \sum_p \frac{1}{N_p} \log[\mathtt{P}_{\mathtt{RNN}}(\mathbf{w}_p|\mathbf{f}_q)] & (17) \end{cases}$$

$$\hat{t}_{\mathbf{w}} = \arg\max_t \{ \frac{1}{|\mathcal{M}_t^{-1}|} \sum_{q \in M_t^{-1}} \mathtt{L}(q)\} \quad (18)$$

It is trivial to extend this approach to yield the N-best solutions by simply taking the top N outputs of equation 18.

The RNNLM approach has several benefits over standard approaches. Firstly, this approach explicitly takes account of word-order in determining the topical similarity of the response. Secondly, there is no need to explicitly select a distance metric. Thirdly, the problems of robustly estimating a vector representation $\mathbf{f}_{\mathbf{w}}$ of the test response are sidestepped. Furthermore, the RNNLM accounts for a broad range of responses because it is trained on individual response utterances which it associates with a question topic vector. This makes

it more scalable than the KNN approach because the number of comparisons which need to be made scales only with the number of questions, not the size of the training data. Thus, arbitrarily large data sets can be used to train the model without affecting classification time.

The RNNLM could be used in a KNN-style approach, where it associates each example response with its *individual* topic vector, using $\mathtt{L}(q)$ as a distance metric. However, this is computationally infeasible since computing $\mathtt{L}(q)$ is significantly more expensive than cosine distance and the previously mentioned scalability would be lost.

4 Data and Experimental Set-up

Data from the Business Language Testing Service (BULATS) English tests is used for training and testing. At test time, each response is recognised using an ASR system and the 1-best hypothesis is passed to the topic classifier. The topic detection system decides whether the candidate has spoken off topic by comparing the classifier output to the topic of the question being answered.

4.1 BULATS Test Format and Data

The BULATS Online Speaking Test has five sections (Chambers and Ingham, 2011):

A Candidates respond to eight simple questions about themselves and their work (e.g. what is your name, where do you come from?).

B Candidates read aloud six short texts appropriate to a work environment.

C Candidates talk about a work-related topic (e.g. the perfect office) with the help of prompts which appear on the screen.

D Candidates must describe a graph or chart such as a pie or a bar chart related to a business situation (e.g. company exports).

E Candidates are asked to respond to five open-ended questions related to a single context prompt. For example a set of five questions about organizing a stall at a trade fair.

Candidates are given a test consisting of 21 questions, however, only the last three sections, consisting of 7 questions, are spontaneously constructed responses to open ended question, and therefore of relevance to this work. Each unique set of 7 questions is a *question script*.

Training, development and evaluation data sets composed of predominantly Gujarati L1 candidates are used in these experiments. The data sets are designed to have an (approximately) even distribution over grades as well as over the different *question scripts*.

During operation the system will detect off-topic responses based on ASR transcriptions, so for the system to be matched it needs to be trained on ASR transcriptions as well. Thus, two training sets are made by using the ASR architecture described in section 4.2 to transcribe candidate responses. Each training set covers the same set of 282 unique topics. The first training set consists of data from 490 candidates, containing 9.9K responses, with an average of 35.1 responses per topic. The second, much larger, training set consists of data from 10004 candidates, containing 202K responses, with an average of 715.5 responses per topic.

Characteristic	Section				
	A	B	C	D	E
# Unique Topics	18	144	17	18	85
# Questions/Section	6	8	1	1	5
Av. # Words/Resp.	10	10	61	77	20

Table 1: Data Characteristics.

As Table 1 shows, the average response length varies across sections due to the nature of the sections. Shorter responses to questions are observed for sections A, B and E, with longer responses to C and D. Estimating topic representations for sections A, B and E questions based on individual responses would be problematic due to the short response lengths. However, by aggregating example responses across candidates, as described in section 2.2, the average length of responses in all sections is significantly longer, allowing the example-response topic space to be robustly defined.

Section E topics correspond to topics of sub-questions relating to an overall question, thus there are only 15 unique questions in section E. However, the sub-questions are sufficiently distinct to merit their own topic vectors. At classification time confusions between sub-questions of an overall section E question are not considered mistakes.

Held-out test sets are used for development, *DEV*, and evaluation, *EVAL*, composed of 84 and 223 candidates, respectively. ASR transcriptions are used for these test sets, as per the operating

scenario. A version of the DEV set with professionally produced transcriptions, *DEV REF*, is also used in training and development.

The publicly available Gibbs-LDA toolkit (Phan and Nguyen, 2007) is used to estimate LDA posterior topic vectors and the scikit-learn 17.0 toolkit (Pedregosa et al., 2011) to estimate LSA topic representations. The topic adapted RNNLM uses a 100-dimensional hidden layer. DEV REF is used as a validation set for early stopping to prevent over-fitting. The CUED RNNLM toolkit v0.1 (Chen et al., 2016) is used for RNNLM training, details of which can be found in (Chen et al., 2014; Mikolov et al., 2010)

4.2 ASR System

A hybrid deep neural network DNN-HMM system is used for ASR (Wang et al., 2015). The acoustic models are trained on 108.6 hours of BULATS test data (Gujarati L1 candidates) using an extended version of the HTK v3.4.1 toolkit (Young et al., 2009; Zhang and Woodland, 2015). A Kneser-Ney trigram LM is trained on 186K words of BULATs test data and interpolated with a general English LM trained on a large broadcast news corpus, using the SRILM toolkit (Stolcke, 2002). Lattices are re-scored with an interpolated trigram+RNN LM (Mikolov et al., 2010) by applying the 4-gram history approximation described in (Liu et al., 2014), where the RNNLM is trained using the CUED RNNLM toolkit (Chen et al., 2016). Interpolation weights are optimized on the DEV REF data set. Table 2 shows the word error rate (WER) on the DEV test set relative to the DEV REF references for each section and the combined spontaneous speech sections (C-E).

% WER					
A	B	C	D	E	C-E
30.6	23.2	32.0	29.9	32.3	31.5

Table 2: ASR performance on DEV.

5 Experiments

Two forms of experiment are conducted in order to assess the performance of the topic-adapted RNNLM. First, a topic classification experiment is run where the ability of the system to accurately recognize the topic of a response is evaluated. Second, a closed-set off-topic response detection experiment is done.

In the experimental configuration used here a response is classified into a topic and the accuracy is measured. The topic of the question being answered is known and all responses are actually on-topic. A label (on-topic/off-topic) is given for each response based on the output of the classifier relative to the question topic. Thus, results presented are in terms of false rejection (FR) and false acceptance (FA) rates rather than precision and recall.

Initial topic detection experiments were run using the DEV test set with both the reference transcriptions (REF) and recognition hypotheses (ASR) to compare different KNN and RNN systems. After this, performance is evaluated on the EVAL test set. The systems were trained using data sets of 490 and 10004 candidates, as described in section 4.1.

5.1 Topic Classification

Performance of the topic-adapted RNNLM is compared to the KNN classifier in Table 3. The RNN1 system outperforms the KNN system by 20-35 % using the LDA topic representation. Furthermore, the KNN system performs worse on section E than it does on section C, while RNN1 performance is better on section E by 7-10% than on section C. The LSA topic representation consistently yields much better performance than LDA by 25-50% for both systems. Thus, the LDA representation is not further investigated in any experiments.

When using the LSA representation the RNN1 system outperforms the KNN system only marginally, due to better performance on section E. Additionally, unlike the KNN-LDA system, the KNN-LSA system does not have a performance degradation on section E relative to section C. Notably, the RNN1 system performs better on section E by 5-13% than on section C. Clearly, section C questions are hardest to assess. Combining both representations through concatenation does not effect performance of the KNN system and slightly degrades RNN1 performance on section C. KNN and RNN1 systems with either topic representation perform comparably on REF and ASR. This suggests that the systems are quite robust to WER rates of 31.5% and the differences are mostly noise.

Training the RNN2 system on 20 times as much data leads to greatly improved performance over KNN and RNN1 systems, almost halving the over-

Topic Repn.	System	# Cands.	C		D		E		ALL (C-E)	
			REF	ASR	REF	ASR	REF	ASR	REF	ASR
LDA	KNN	490	75.0	81.0	37.0	42.0	91.8	91.1	68.0	71.4
	RNN1		61.9	58.3	28.4	25.9	48.8	51.2	46.6	45.4
LSA	KNN	490	32.1	28.6	2.5	3.7	31.3	33.3	22.0	21.9
	RNN1		29.8	31.0	4.9	6.2	23.8	23.8	19.7	20.5
	RNN2	10004	19.0	19.0	3.7	3.7	9.5	10.7	10.8	11.2
LDA +LSA	KNN	490	30.9	29.8	2.5	3.7	31.5	33.3	21.7	22.3
	RNN1		32.1	35.7	4.9	4.9	23.8	22.6	20.5	21.3
	RNN2	10004	25.0	22.6	4.9	4.9	10.7	10.7	13.7	12.9

Table 3: % False rejection in topic detection using KNN classifier with 6 nearest neighbour and distance weights and RNNLM classifier on the DEV test set. 280 dim. topic spaces for LDA and LSA, and 560 dim. for LDA+LSA.

all error rate. The KNN system could not be evaluated effectively in reasonable time using 20 times as many example responses and results are not shown, while RNN2 evaluation times are unaffected. Notably, RNN performance using the LSA representation scales with training data size better than with the LDA+LSA representation. Thus, we further investigate systems only with the LSA representation. Interestingly, section D performance is improved only marginally.

Performance on section D is always best, as section D questions relate to discussion of charts and graphs for different conditions for which the vocabulary is very specific. Section C and E questions are the less distinct because they have free-form answers to broad questions, leading to higher response variability. This makes the linking of topic from the training data to the test data more challenging, particularly for 1-best classification, leading to higher error rates.

Figure 2 shows the topic classification confusion matrix for the RNN1 LSA system. A similar matrix is observed for the KNN LSA system. Most confusions are with topics from the same section. This is because each section has a distinct style of questions and some questions within a section are similar. An example is shown below. Question SC-EX1 relates to personal local events in the workplace. SC-EX2, which relates to similar issues, is often confused with it. On the other hand, SC-EX3 is rarely confused with SC-EX1 as it is about non-personal events on a larger scale.

- SC-EX1: Talk about some advice from a colleague. You should say: what the advice was, how it helped you and whether you would give the same advice to another colleague.
- SC-EX2: Talk about a socially challenging day you had at work. You should say: what was the challenging situation, how you resolved it and why you found it challenging.
- SC-EX3: Talk about a company in your local town which you admire. You should say: what company it is, what they do, why you admire them, and how the company impacts life in your town.

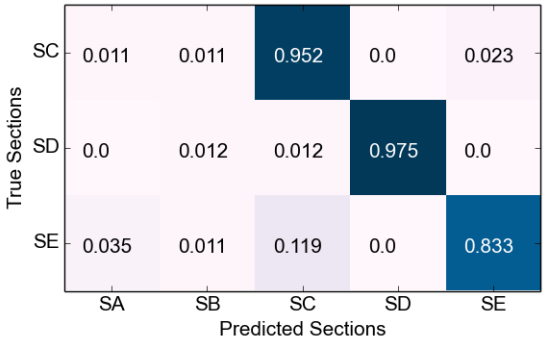


Figure 2: RNN1 LSA confusion matrix on DEV ASR.

System performance can be increased by considering N-best results, as described in Section 3. Results for systems trained on 490 and 10004 candidates are presented in Table 4. The error rate decreases as the value of N increases for all systems. However, performance scales better with N for the RNN systems than for KNN. Notably, for values of $N > 1$ performance of all systems on REF is better, which suggests that ASR errors do have a minor impact on system performance.

5.2 Off-topic response detection

In the second experiment off-topic response detection is investigated. Performance is measured

N	System	# Cands.	REF	ASR
1	KNN	490	22.1	21.9
	RNN1		19.7	20.5
	RNN2	10004	10.8	11.2
2	KNN	490	15.9	16.0
	RNN1		13.7	16.1
	RNN2	10004	6.8	7.6
3	KNN	490	13.5	14.3
	RNN1		10.4	11.2
	RNN2	10004	6.4	7.2
4	KNN	490	11.1	12.5
	RNN1		8.8	10.0
	RNN2	10004	5.2	6.4

Table 4: N-Best % false rejection performance of KNN and RNNLM classifiers with the LSA topic space on the DEV test set

in terms of the false acceptance (FA) probability of an off-topic response and false rejection (FR) probability of an on-topic response. The experiment is run on DEV and EVAL test sets. Since neither DEV nor EVAL contain real off-topic responses, a pool $\mathbf{W}_q$ of such responses is synthetically generated for each question by using valid responses to other questions in the data set. Off-topic responses are then selected from this pool. A selection strategy defines which responses are present in $\mathbf{W}_q$. Rather than using a single selection of off-topic responses, an expected performance over all possible off-topic response selections is estimated. The overall probability of falsely accepting an off-topic response can be expressed using equation 19.

$$\mathtt{P}(\mathtt{FA}) = \sum_{q=1}^{Q} \sum_{\mathbf{w} \in \mathbf{W}_q} \mathtt{P}(\mathtt{FA}|\mathbf{w}, q)\mathtt{P}(\mathbf{w}|q)\mathtt{P}(q) \quad (19)$$

In equation 19, the question q is selected with uniform probability from the set Q of possible questions. The candidate randomly selects with uniform probability $\mathtt{P}(\mathbf{w}|q)$ a response $\mathbf{w}$ from the pool $\mathbf{W}_q$. The correct response to the question is not present in the pool. The conditional probability of false accept $\mathtt{P}(\mathtt{FA}|\mathbf{w}, q) = 1$ if $\mathcal{M}(q) \in \hat{\mathbf{t}}_N$, and $\mathcal{M}(q)$ is not the real topic of the response $\mathbf{w}$, otherwise $\mathtt{P}(\mathtt{FA}|\mathbf{w}, q) = 0$.

As shown in Figure 2, the main confusions will occur if the response is from the same section as the question. Two strategies for selecting off-topic responses are considered based on this: `naive`, where an incorrect response can be selected from any section; and `directed`, where an incorrect response can only be selected from the same section as the question. The `naive` strategy represents candidates who have little knowledge of the system and memorise responses unrelated to the test, while the `directed` strategy represents those who are familiar with the test system and have access to real responses from previous tests.

Test Set	System	% Equal Error Rate	
		Directed	Naive
DEV	KNN	13.5	10.0
	RNN1	10.0	7.5
	RNN2	7.5	6.0
EVAL	KNN	12.5	9.0
	RNN1	8.0	6.0
	RNN2	5.0	4.5

Table 5: % Equal Error Rate for LSA topic space systems on the DEV and EVAL test sets.

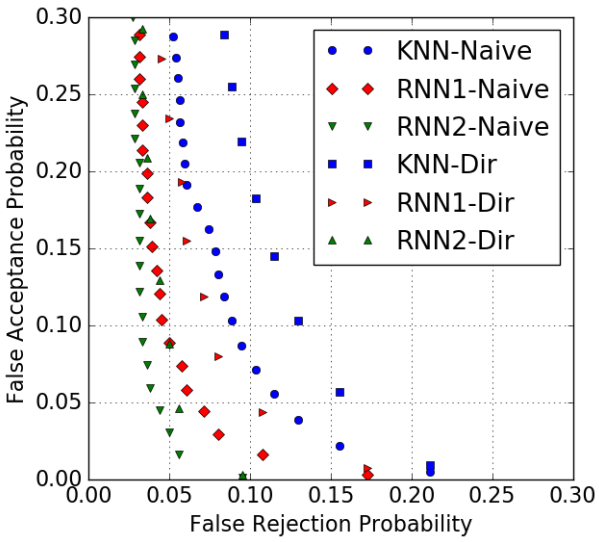


Figure 3: ROC curves of LSA topic space systems on the EVAL test set.

A Receiver Operating Characteristic (ROC) curve (Figure 3) can be constructed by plotting the FA and FR rates for a range of N. The RNN1 system performs better at all operating points than the KNN system for both selection strategies and evaluation test sets. Equal Error Rates (EER), where FA = FR, are given in Table 5. Results on EVAL are more representative of the difference between the KNN and RNN performance, as they are evaluated on nearly 3 times as many candidates. The

RNN2 system achieves the lowest EER. It is interesting that for better systems the difference in performance against the `naive` and `directed` strategies decreases. This indicates that the systems become increasingly better at discriminating between similar questions.

As expected, the equal error rate for the `directed` strategy is higher than for the `naive` strategy. In relation to the stated task of detecting when a test candidate is giving a memorized response, the `naive` strategy represents a lower-bound on realistic system performance, as students are not likely to respond with a valid response to a different question. Most likely they will fail to construct a valid response or will add completely unrelated phrases memorised beforehand, which, unlike responses from other sections, may not come from the same domain as the test (eg: Business for BULATs).

6 Conclusion and Future Work

In this work a novel off-topic content detection framework based on topic-adapted RNNLMs was developed. The system was evaluated on the task of detecting off-topic spoken responses on the BULATS test. The proposed approach achieves better topic classification and off-topic detection performance than the standard approaches.

A limitation of both the standard and proposed approach is that if a new question is created by the test-makers, then it will be necessary to collect example responses before it can be widely deployed. However, since the system can be trained on ASR transcriptions, the example responses do not need to be hand-transcribed. This is an attractive deployment scenario, as only a smaller hand-transcribed data set is needed to train an ASR system with which to cost-effectively transcribe a large number of candidate recordings.

Further exploration of different topic vector representations and their combinations is necessary in future work.

Acknowledgements

This research was funded under the ALTA Institute, University of Cambridge. Thanks to Cambridge English, University of Cambridge, for supporting this research and providing access to the BULATS data.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, March.

Lucy Chambers and Kate Ingham. 2011. The BULATS online speaking test. *Research Notes*, 43:21–25.

Xie Chen, Yongqiang Wang, Xunying Liu, Mark J.F. Gales, and P.C. Woodland. 2014. Efficient GPU-based Training of Recurrent Neural Network Language Models Using Spliced Sentence Bunch. In *Proc. INTERSPEECH*.

Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Mark J.F. Gales, and Philip C. Woodland. 2015. Recurrent Neural Network Language Model Adaptation for Multi-Genre Broadcast Speech Recognition. In *Proc. INTERSPEECH*.

X. Chen, X. Liu, Y. Qian, M.J.F. Gales, and P.C. Woodland. 2016. CUED-RNNLM – an open-source toolkit for efficient training and evaluation of recurrent neural network language models. In *Proc. ICASSP*.

Keelan Evanini and Xinhao Wang. 2014. Automatic detection of plagiarized spoken responses. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*.

Keelan Evanini, Shasha Xie, and Klaus Zechner. 2013. Prompt-based Content Scoring for Automated Spoken Language Assessment. In *Proc. NAACL-HLT*.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding Scientific Topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235.

Thomas K Landauer, Peter W. Foltz, and Darrell Laham. 1998. Introduction to Latent Semantic Analysis. *Discourse Processes*, 25:259–284.

Xunying Liu, Y. Wang, Xie Chen, Mark J.F. Gales, and Philip C. Woodland. 2014. Efficient Lattice Rescoring using Recurrent Neural Network Language Models. In *Proc. INTERSPEECH*.

Angeliki Metallinou and Jian Cheng. 2014. Using Deep Neural Networks to Improve Proficiency Assessment for Children English Language Learners. In *Proc. INTERSPEECH*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context Dependent Recurrent Neural Network Language Model. In *Proc. IEEE Spoken Language Technology Workshop (SLT)*.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network Based Language Model. In *Proc. INTERSPEECH*.

Tomas Mikolov. 2012. *Statistical Language Models Based on Neural Networks*. Ph.D. thesis, Brno University of Technology.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Xuan-Hieu Phan and Cam-Tu Nguyen. 2007. GibbsLDA++: A C/C++ implementation of latent Dirichlet allocation (LDA). `http://gibbslda.sourceforge.net/`.

Barbara Seidlhofer. 2005. English as a lingua franca. *ELT journal*, 59(4):339.

A Stolcke. 2002. SRILM – an extensible language modelling toolkit. In *Proc. ICSLP*.

Haipeng Wang, Anton Ragni, Mark J. F. Gales, Kate M. Knill, Philip C. Woodland, and Chao Zhang. 2015. Joint Decoding of Tandem and Hybrid Systems for Improved Keyword Spotting on Low Resource Languages. In *Proc. INTERSPEECH*.

Shasha Xie, Keelan Evanini, and Klaus Zechner. 2012. Exploring Content Features for Automated Speech Scoring. In *Proc. NAACL-HLT*.

Helen Yannakoudakis. 2013. Automated assessment of English-learner writing. Technical Report UCAM-CL-TR-842, University of Cambridge Computer Laboratory.

Su-Youn Yoon and Shasha Xie. 2014. Similarity-Based Non-Scorable Response Detection for Automated Speech Scoring. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications*.

Steve Young, Gunnar Evermann, Mark J. F. Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) Liu, Gareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Woodland. 2009. *The* HTK *book (for* HTK *Version 3.4.1)*. University of Cambridge.

Klaus Zechner, Derrick Higgins, Xiaoming Xi, and David M. Williamson. 2009. Automatic scoring of non-native spontaneous speech in tests of spoken English. *Speech Communication*, 51(10):883–895. Spoken Language Technology for Education Spoken Language.

Chau Zhang and Philip C. Woodland. 2015. A General Artificial Neural Network Extension for HTK. In *Proc. INTERSPEECH*.

Synthesizing Compound Words for Machine Translation

Austin Matthews and **Eva Schlinger** and **Alon Lavie** and **Chris Dyer**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{austinma,eschling,alavie,cdyer}@cs.cmu.edu

Abstract

Most machine translation systems construct translations from a closed vocabulary of target word forms, posing problems for translating into languages that have productive compounding processes. We present a simple and effective approach that deals with this problem in two phases. First, we build a classifier that identifies spans of the input text that can be translated into a single compound word in the target language. Then, for each identified span, we generate a pool of possible compounds which are added to the translation model as "synthetic" phrase translations. Experiments reveal that (i) we can effectively predict what spans can be compounded; (ii) our compound generation model produces good compounds; and (iii) modest improvements are possible in end-to-end English–German and English–Finnish translation tasks. We additionally introduce KomposEval, a new multi-reference dataset of English phrases and their translations into German compounds.

1 Introduction

Machine translation systems make a closed-vocabulary assumption: with the exception of basic rules for copying unknown word types from the input to the output, they can produce words in the target language only from a fixed, finite vocabulary. While this is always a naïve assumption given the long-tailed distributions that characterize natural language, it is particularly challenging in languages such as German and Finnish that have productive compounding processes.

In such languages, expressing compositions of basic concepts can require an unbounded number of words. For example, English multiword phrases like *market for bananas*, *market for pears*, and *market for plums* are expressed in German with single compound words (respectively, as *Bananenmarkt*, *Birnenmarkt*, and *Pflaumenmarkt*). Second, while they are individually rare, compound words are, on the whole, frequent in native texts (Baroni et al., 2002; Fritzinger and Fraser, 2010). Third, compounds are crucial for translation quality. Not only does generating them make the output seem more natural, but they are content-rich. Since each compound has, by definition, at least two stems, they are intuitively (at least) doubly important for translation adequacy.

Fortunately, compounding is a relatively regular process (as the above examples also illustrate), and it is amenable to modeling. In this paper we introduce a two-stage method (§2) to dynamically generate novel compound word forms given a source language input text and incorporate these as "synthetic rules" in a standard phrase-based translation system (Bhatia et al., 2014; Chahuneau et al., 2013; Tsvetkov et al., 2013). First, a binary classifier examines each source-language sentence and labels each span therein with whether that span could become a compound word when translated into the target language. Second, we transduce the identified phrase into the target language using a word-to-character translation model. This system makes a closed vocabulary assumption, albeit at the character (rather than word) level—thereby enabling new word forms to be generated. Training data for these models is extracted from automatically aligned and compound split parallel corpora (§3).

We evaluate our approach on both intrinsic and extrinsic metrics. Since German compounds are relatively rare, their impact on the standard MT evaluation metrics (e.g., BLEU) is minimal, as we

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1085–1094,
Berlin, Germany, August 7-12, 2016.

show with an oracle experiment, and we find that our synthetic phrase approach obtains only modest improvements in overall translation quality. To better assess its merits, we commissioned a new test set, which we dub KomposEval (from the German word for a compound word, *Komposita*), consisting of a set of 1090 English phrases and their translations as German compound words by a professional English–German translator. The translator was instructed to produce as many compound-word translations as were reasonable (§4). This dataset permits us to evaluate our compound generation component directly, and we show that (i) without mechanisms for generating compound words, MT systems cannot produce the long tail of compounds; and (ii) our method is an effective method for creating correct compounds.

2 Compound Generation via Rule Synthesis

Suppose we want to translate the sentence

the market for bananas has collapsed .

from English into German. In order to produce the following (good) translation,

der bananenmarkt ist abgestürzt .

a phrase-based translation system would need to contain a rule similar to *market for bananas* → *bananenmarkt*. While it is possible that such a rule would be learned from parallel corpora using standard rule extraction techniques, it is likely that such a rule would not exist (unless the system were trained on the translation examples from this paper).

We solve the compound translation problem by "filling in" such missing rule gaps in the phrase table. The process takes place in two parts: first, identifying spans in the input that appear to be translatable as compounds (§2.1), and second, generating candidate compounds for each positively identified span (§2.2). Since synthesized rules compete along side rules which are learned using standard rule extraction techniques (and which are often quite noisy), our rule synthesis system can *overgenerate* rule candidates, a fact which we exploit in both phases.

2.1 Phase I: Classifying Compoundable Spans

Given a source sentence, we classify each span therein (up to some maximum length) as either compoundable or non-compoundable using independent binary predictions. Rather than attempting to hand-engineer features to represent phrases, we use a bidirectional LSTM to learn a fixed-length vector representation $\mathbf{h}_{i,j}$ that is computed by composing representations of the tokens $(f_i, f_{i+1}, \ldots, f_j)$ in the input sentence. The probability that a span is compoundable is then modeled as:

$$p(\text{compoundable? } | f_i, f_{i+1}, \ldots, f_j) = \sigma\left(\mathbf{w}^\top \tanh(\mathbf{V}\mathbf{h}_{i,j} + \mathbf{b}) + a\right),$$

where σ is the logistic sigmoid function, and $\mathbf{w}$, $\mathbf{V}$, $\mathbf{b}$, and a are parameters.

To represent tokens that are inputs to the LSTM, we run a POS tagger (Toutanova et al., 2003), and for each token concatenate a learned embedding of the tag and word. Figure 1 shows the architecture.

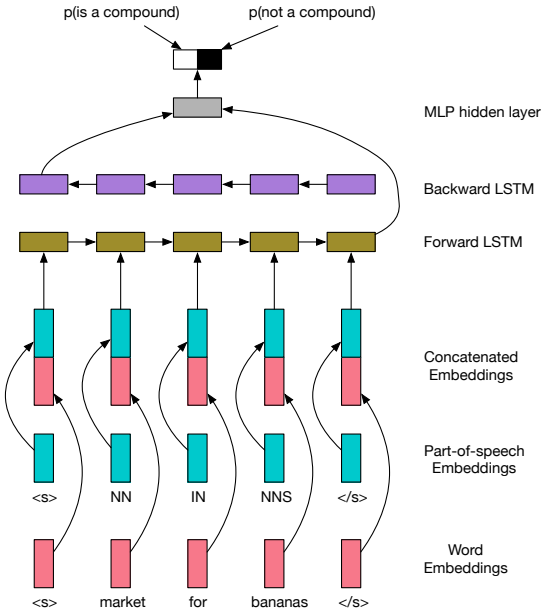


Figure 1: A graphical representation of the neural network used for classifying whether an input source phrase should or should not turn into a compound word in the target language

2.2 Phase II: Generating Compound Words

The second stage of our compound-generating pipeline is to generate hypothesis compound words for each source phrase that was identified as "compoundable" by the classifier just discussed. We do this by using a word-to-character–based machine translation system, which enables us to reuse a standard phrase-based decoder for compound generation.

2.2.1 Generation Model

The cornerstone of our generation approach is the forward and backward lexical translation tables learned by an unsupervised word aligner. We combine these two translation tables to create a word-to-character phrase table compatible with a standard decoder. This table allows our generator to know the correct translations of individual morphemes, but alone does not allow the generator to build full compound words.

To capture the small bits of "phonetic glue" (e.g., the *n* that occurs between *banane* and *markt* in the compound *bananenmarkt*) that may occur when generating compound words, we insert a special `SUF` symbol in between each pair of source words. This symbol will allow us to insert a small suffix in between the translations of source words.

Finally, we insert a special `END` symbol at the end of each source phrase. This symbol will allow the model to generate morphological variants due to suffixes indicating case, number, and agreement that only occur at the end of a whole compound word, but not in between the individual pieces. Some examples of all three types of rules are shown in Table 1.

2.2.2 Reordering and Word Dropping

We observe that in order to generate many compounds, including *bananenmarkts* from "market for bananas", a system must be able to both reorder and drop source words at will. Implemented naïvely, however, these allowances may produce invalid interleavings of source words and `SUF`/`END` tokens. For example, if we (correctly) drop the word "for" from our example, we might feed the decoder the sequence "market `SUF SUF` bananas `END`.

To disallow such bogus input sequences we disable all reordering inside the decoder, and instead encode all possible reorderings in the form of an input lattice (Dyer et al., 2008). Moreover, we allow the decoder to drop non-content words by skipping over them in the lattice. Each edge in our lattices contains a list of features, including the indices, lexical forms, and parts of speech of each word kept or dropped. Each possible sequence in the lattice also encodes features of the full path of source words kept, the full list of source words dropped, the parts of speech of the path and all dropped words, and the order of indices traversed.

With these constraints in place we can train the compound generator as though it were a normal MT system with no decode-time reordering.

3 Training

Our approach to generating compound word forms in translation has two stages. First, we build a classifier that chooses spans of source text that could produce target compounds. Second, we build a compound generator that outputs hypothesis word forms, given a source phrase. We will detail each of these steps in turn.

3.1 Extracting Compounds from Bitext

In order to learn to generate compound words we naturally require training data. Ideally we would like a large list of English phrases with their natural contexts and translations as German compounds. Of course virtually no such data exists, but it is possible to extract from parallel data, using a technique similar to that used by Tsvetkov and Wintner (2012).

To this end, we take our tokenized bitext and pass it through Dyer (2009)'s German compound splitter. We then align the segmented variant using the `fast_align` tool in both the forward and reverse directions, which produces both word alignments and lexical translation tables, which give the probability of a compound part given an English phrase. We then symmetrize the produced pair of alignments with the intersection heuristic. This results in a sparse alignment in which each target word is aligned to either 0 or 1 source words. We then undo any splits performed by the compound splitter, resulting in a corpus where the only words aligned many-to-one are precisely well-aligned compounds.

This process produces two crucially important data. First, a list of English phrase pairs that may become compound words in German on which we train our classifier. Second, the lexical translation tables, trained on compound split German data, which form the basis of our generation approach.

3.2 Training the Compoundability Classifier

The network is trained to maximize cross-entropy of its training data using the Adam optimizer (Kingma and Ba, 2014) until performance on a held-out dev set stops improving.

Due to the fact that we only need to represent the "compoundability" of each source-language word, and not its full semantics, we find that very small (10-dimensional) word and POS em-

Source	Target	Non-Zero Features
bananas	b a n a n e	$\phi_{fwd} = -0.495208$ $\phi_{rev} = -0.455368$
market	m a r k t	$\phi_{fwd} = -0.499118$ $\phi_{rev} = -0.269879$
SUF	n	$\phi_{fwd} = -3.718241$ $\phi_{\text{uses\_suf\_n}} = 1.0$
END	s	$\phi_{fwd} = -2.840721$ $\phi_{\text{uses\_end\_s}} = 1.0$

Table 1: A fragment of the word-to-character rules used in the compound generation system.

beddings work well. The recurrent part of the neural network uses two-layer LSTM (Hochreiter and Schmidhuber, 1997) cells with the hidden layer size set to 10. The final MLP's hidden layer size is also set to 10.

The training data is processed such that each span of length two to four is considered one training example, and is labeled as positive if it is well-aligned (Brown et al., 1993) to a single German compound word. Since most spans do not translate as compounds, we are faced with an extreme class imbalance problem (a ratio of about 300:1). We therefore experiment with down sampling the negative training examples to have an equal number of positive and negative examples.

3.3 Training the Compound Generation Model

As a translation model, there are two components to learning the translation system: learning the rule inventory and their features (§3.3.1) and learning the parameters of the generation model (§3.3.2).

3.3.1 Learning Word to Character Sequence Translation Rules

The possible translations of SUF and END are learned from the list of positive training examples extracted for our classifier. For each example, we find all the possible ways the source words could translate, in accordance with our translation table, into nonoverlapping substrings of the target word. Any left over letters in between pieces become possible translations of SUF, while extra letters at the end of the target string become possible translations of END. Probabilities for each translation are estimated by simply counting and normalizing the number of times each candidate was seen. See Figure 2 for an example of this splitting process.

3.3.2 Learning Generator Feature Weights

Since the generator model is encoded as a phrase-based machine translation system, we can train it using existing tools for this task. We choose to train using MIRA (Crammer and Singer, 2003), and use a 10-gram character-based language model trained on the target side of the positive training examples extracted for the classifier.

4 KomposEval Data Set

To evaluate our compound generator we needed a dataset containing English phrases that should be compounded along with their German translations. To the best of our knowledge, no substantial human-quality dataset existed, so we created one as part of this work.

We took our list of automatically extracted (English phrase, German compound) pairs and manually selected 1090 of them that should compound. We then asked a native German speaker to translate each English phrase into German compounds, and to list as many possibile compound translations as she could think of. The result is a test set consisting of 1090 English phrases, with between 1 and 5 possible German compound translations for each English phrase. This test set is published as supplementary material with this article. Some example translations are shown in Table 2.

Source phrase	Reference(s)
transitional period	Übergangsphase
	Übergangsperiode
	Übergangszeitraum
Chamber of deputies	Abgeordnetenhaus
	Abgeordnetenkammer
self-doubt	Selbstzweifel

Table 2: Examples of human-generated compounds from the KomposEval data set

5 Experiments

Before considering the problem of integrating our compound model with a full machine translation system, we perform an intrinsic evaluation of each of the two steps of our pipeline.

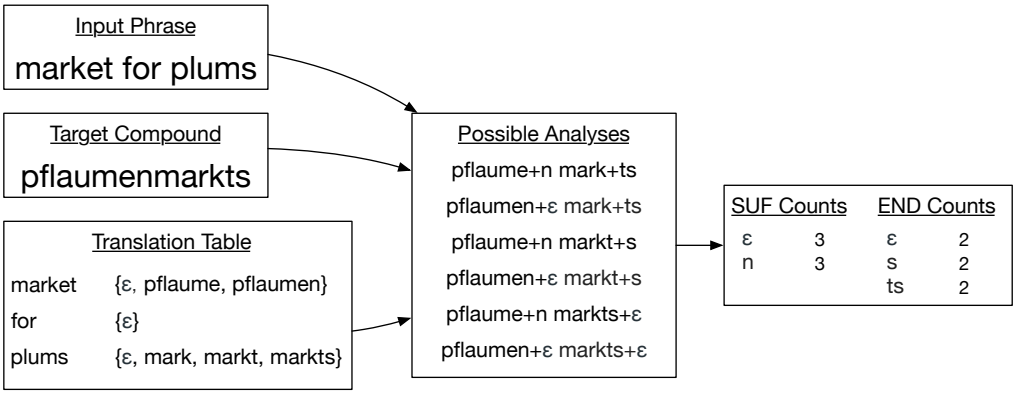


Figure 2: Decomposition of a target compound into possible analyses, given a source phrase and a morpheme-level translation table. This process allows us to learn the "phonetic glue" that can go in between morphemes, as well as the inflections that can attach to the end of a compound word.

5.1 Classifier Intrinsic Evaluation

We evaluate the effectiveness of our classifier, by measuring its precision and recall on the two held out test sets described in §2.1 taken from two language pairs: English–German and English–Finnish. Furthermore, we show results both with down-sampling (balanced data set) and without down-sampling (unbalanced data set).

Our classifier can freely trade off precision and recall by generalizing its requirement to call an example positive from $p(\text{compound} \mid \text{span}) > 0.5$ to $p(\text{compound} \mid \text{span}) > \tau$, for $\tau \in (0, 1)$, allowing us to report full precision-recall curves (Figure 3).

We find that our best results for the unbalanced cases come at $\tau = 0.24$ for German and $\tau = 0.29$ for Finnish, with F-scores of 20.1% and 67.8%, respectively. In the balanced case, we achieve 67.1% and 97.0% F-scores with $\tau = 0.04$ and $\tau = 0.57$ on German and Finnish respectively.

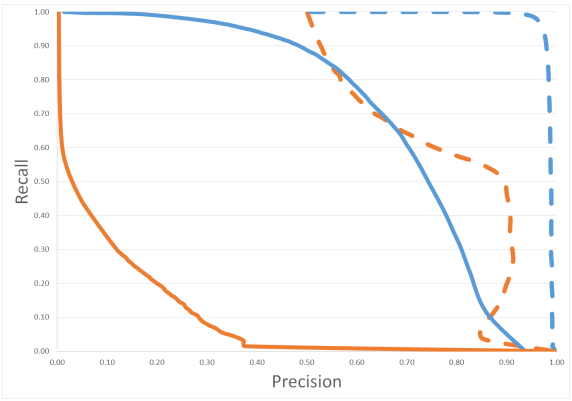


Figure 3: Precision-recall curves for our compound classifier for two languages: German (red) and Finnish (blue). Unbalanced test set results are shown with solid lines. Balanced test set results are shown with dashed lines.

5.2 Generator Instrinsic Evaluation

To evaluate our compound generator, we fed it the source side of our newly created KomposEval corpus and had it output a 100-best list of hypotheses translations for each English phrase. From this we are able to compute many intrinsic quality metrics. We report the following metrics:

- Mean reciprocal rank (MRR); which is one divided by the average over all segments of the position that the reference translation appears in our k-best list.
- Character error rate (CER), or the average number of character-level edits that are required to turn our 1-best hypothesis into the nearest of the reference translations.
- Precision at 1, 5, and 10, which indicate what percentage of the time a reference translation can be found in the top 1, 5, or 10 hypotheses of our k-best list, respectively.

These results can be found in Table 3. We compare to a naïve baseline that is just a standard English–German phrase-based translation system with no special handling of compound word forms. We immediately see that the baseline system is simply unable to generate most of the compound words in the test set, resulting in extraordinarily low metric scores across the board. Its one saving grace is its tolerable CER score, which shows that the system is capable of generating the correct morphemes, but is failing to correctly ad-

	MRR ↑	CER ↓	P@1 ↑	P@5 ↑	P@10 ↑
Baseline	<0.01	3.305	0%	0%	<0.01%
Our model	0.7004	2.506	61.38%	81.47%	84.31%

Table 3: Mean reciprocal rank, character error rate, and precision at K statistics of our baseline MT system and our compound generator.

join them and add the phonological glue required to produce a well-formed compound word. Our system, on the other hand, is capable of reaching at least one of the five references for every single sentence in the test set, and has a reference translation in the top 5 hypotheses in its k-best list over 80% of the time.

Qualitatively, the compounds generated by our model are remarkably good, and very understandable. Major error classes include incorrect word sense, non-compositional phrases, and special non- concatenative effects at word boundaries. An example of each of these errors, along with some examples of good compound generation can be found in Table 4.

5.3 Extrinsic Translation Evaluation

Finally, we use our compound generator as part of a larger machine translation pipeline. We run our compound span classifier on each of our translation system's tune and test sets, and extract our generator's top ten hypotheses for each of the postively identified spans. These English phrases are then added to a synthetic phrase table, along with their German compound translations, and two features: the compound generator's score, and an indicator feature simply showing that the rule represents a synthetic compound. Table 5 shows some example rules of this form. The weights of these features are learned, along with the standard translation system weights, by the MIRA algorithm as part of the MT training procedure.

The underlying translation system is a standard Hiero (Chiang et al., 2005) system using the cdec (Dyer et al., 2010) decoder, trained on all constrained-track WMT English–German data as of the 2014 translation task. Tokenization was done with `cdec`'s `tokenize-anything` script. The first character of each sentence was down cased if the unigram probability of the downcased version of the first word was higher than that of the original casing. Word alignment was performed using `cdec`'s `fast_align` tool,

		BLEU ↑	METR ↑	TER ↓	Len
WMT2012	Baseline	16.2	34.5	64.8	94.1
	+Our Compounds	16.3	34.6	64.9	94.2
	+Oracle Compounds	16.9	35.2	64.6	95.5
WMT2013*	Baseline	18.8	37.3	62.1	93.6
	+Our Compounds	18.9	37.5	62.3	96.7
	+Oracle Compounds	19.7	38.2	61.9	97.6
WMT2014	Baseline	19.6	38.9	64.3	103.5
	+Compounds	19.6	39.0	64.5	103.9
	+Oracle Compounds	21.7	40.9	61.1	100.6

Table 6: Improvements in English–German translation quality using our method of compound generation on WMT 2012, 2013, and 2014. * indicates the set used for tuning the MT system.

and symmetrized using the `grow-diag` heuristic. Training is done using cdec's implementation of the MIRA algorithm. Evaluation was done using MultEval (Clark et al., 2011). A 4-gram language model was estimated using KenLM's `lmplz` tool (Heafield et al., 2013).

In addition to running our full end-to-end pipeline, we run an oracle experiment wherein we run the same pre-processing pipeline (compound splitting, bidirectionally aligning, intersecting, and de-splitting) on each test set to identify which spans do, in fact, turn into compounds, as well as their ideal translations. We then add grammar rules that allow precisely these source spans to translate into these oracle translations. This allows us to get an upper bound on the impact compound generation could have on translation quality.

The results, summarized in Table 6 and Table 7, show that adding these extra compounds has little effect on metric scores compared to our baseline system. Nevertheless, we believe that the qualitative improvements of our methods are more significant than the automatic metrics would indicate. Our method targets a very specific problem that pertains only to dense content-bearing target words that humans find very important. Moreover, BLEU is unable to reasonably evaluate improvements in these long tail phenomena, as it only captures exact lexical matches, and because we are purposely generating fewer target words than a standard translation system.

6 Related Work

Most prior work on compound generation has taken a different approach from the one advo-

Input	Hypothesis	Reference	Comments
cheese specialities	Fachkäse	Käsespezialitäten	Wrong sense of "specialties"
band-aid	Band-hilfe	(Should not compound)	Idiosyncratic meaning
church towers	Kirchentürme	Kirchtürme	Extra word-internal case marking
sugar beet farmers	Zuckerrübenbauern	Zuckerrübenbauern	Perfect
tomato processing	Tomatenverarbeitung	Tomatenverarbeitung	Perfect
generation of electricity	Stromerzeugung	Stromerzeugung	Perfect, including reordering

Table 4: Examples of erroneous (top) and correct (bottom) compounds generated by our system

Source	Target	Non-Zero Features
`market for bananas`	`bananenmarkt`	$\phi_{Compound} = 1$ $\phi_{Score} = -38.9818$
`market for bananas`	`bananenmarktes`	$\phi_{Compound} = 1$ $\phi_{Score} = -49.8976$
`market for bananas`	`marktordnung`	$\phi_{Compound} = 1$ $\phi_{Score} = -53.2197$
`market for bananas`	`bananenmarkts`	$\phi_{Compound} = 1$ $\phi_{Score} = -54.4962$
`market for bananas`	`binnenmarkt`	$\phi_{Compound} = 1$ $\phi_{Score} = -57.6816$

Table 5: Example synthetic rules dynamically added to our system to translate the phrase "market for bananas" into a German compound word. Note that we correctly generate both the nominative form (with no suffix) and the genitive forms (with the -s and -es suffixes).

		BLEU ↑	METR ↑	TER ↓	Len
Dev*	Baseline	12.3	29.0	72.7	96.5
	+Our Compounds	12.3	29.1	72.8	96.8
DevTest	Baseline	11.4	29.9	71.6	96.2
	+Our Compounds	11.6	30.1	71.5	96.4
Test	Baseline	10.8	28.4	73.4	96.7
	+Our Compounds	10.9	28.5	73.3	96.9

Table 7: Improvements in English–Finnish translation quality using our method of compound generation on WMT 2014 tuning, devtest, and test sets. * indicates the set used for tuning the MT system.

cated here, first translating the source language into a morphologically analyzed and segmented variant of the target language, and then performing morphological generation on this sequence (Cap et al., 2014; Irvine and Callison-Burch, 2013; Denkowski et al., 2013; Clifton and Sarkar, 2011; Stymne and Cancedda, 2011).

Requesting multiple translations from a translator has been used in the past, most notably to create HyTER reference lattices (Dreyer and Marcu, 2012). However, in contrast to full-sentence translations the space of possible grammatical compounds is far smaller, substantially simplifying our task.

The splitting of German compound phrases for translation from German into English has been addressed by Koehn and Knight (2001) and Dyer (2009). They elegantly solve the problem of having a large, open vocabulary on the source side by splitting compound words into their constituent morphemes and translating German into English at the morpheme level. Their approach works excellently when translating *out* of a compounding language, but is unable to generate novel compound words in the target language without some sort of post processing.

Dynamic generation of compounds in a target language using such post processing has been examined in the past by Cap et al. (2014) and Clifton and Sarkar (2011). Both perform compound splitting on their parallel data, train a morpheme-based translation system, and then stitch compound words back together using different models. While effective, their approach runs into difficulties if the morphemes that should compound get separated by the reordering model during the translation process. Both address this using more complicated models, whereas our holistic approach handles this problem seamlessly.

Stymne (2012) gives an excellent taxonomy of compound types in Germanic languages, and discusses many different strategies that have been used to split and merge them for the purposes of machine translation. She identifies several difficulties with the split-translate-merge approach and points out some key subtleties, such as handling of bound morphemes that never occur outside of

compounds, that one must bear in mind when doing translation to or from compounding languages.

The idea of using entirely character-based translation systems was introduced by Vilar et al. (2007). While their letter-level translation system alone did not outperform standard phrase-based MT on a Spanish–Catalan task, they demonstrated substantial BLEU gains when combining phrase- and character-based translation models, particularly in low resource scenarios.

7 Conclusion

In this paper we have presented a technique for generating compound words for target languages with open vocabularies by dynamically introducing synthetic translation options that allow spans of source text to translate as a single compound word. Our method for generating such synthetic rules decomposes into two steps. First an RNN classifier detects compoundable spans in the source sentence. Second, a word-to-character machine translation system translates the span of text into a compound word.

By dynamically adding compound words to our translation grammars in this way we allow the decoder, which is in turn informed by the language model, to determine which, if any, of our hypothesized compounds look good in context. Our approach does away with the need for post processing, and avoids complications caused by reordering of morphemes in previous approaches. However, this technique relies heavily on a strong target language model. Therefore, one important extension of our work is to further study the interaction between our model and the underlying language model.

In addition to our generation technique we have presented a new human-quality data set that specifically targets compounding and use it to demonstrate tremendous improvements in our translation system's ability to correctly generalize from compound words found in parallel text to match human translations of unseen compoundable phrases.

Acknowledgements

We thank the anonymous reviewers for their careful reading of the submitted draft of this paper. Furthermore, we thank Isabelle Wolf for her work in creating the KomposEval data set. This research work was supported by a Google faculty research award and by computing resources provided by the NSF-sponsored XSEDE program under grant TG-CCR110017. The statements made herein are solely the responsibility of the authors.

References

Marco Baroni, Johannes Matiasek, and Harald Trost. 2002. Predicting the components of german nominal compounds. In *ECAI*, pages 470–474.

Archna Bhatia, Chu-Cheng Lin, Nathan Schneider, Yulia Tsvetkov, Fatima Talib Al-Raisi, Laleh Roostapour, Jordan Bender, Abhimanu Kumar, Lori Levin, Mandy Simons, et al. 2014. Automatic classification of communicative functions of definiteness. Association for Computational Linguistics.

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

Fabienne Cap, Alexander Fraser, Marion Weller, and Aoife Cahill. 2014. How to produce unseen teddy bears: Improved morphological processing of compounds in SMT. In *Proc. EACL*.

Victor Chahuneau, Eva Schlinger, Noah A Smith, and Chris Dyer. 2013. Translating into morphologically rich languages with synthetic phrases.

David Chiang, Adam Lopez, Nitin Madnani, Christof Monz, Philip Resnik, and Michael Subotin. 2005. The hiero machine translation system: Extensions, evaluation, and analysis. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 779–786. Association for Computational Linguistics.

Jonathan H Clark, Chris Dyer, Alon Lavie, and Noah A Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 176–181. Association for Computational Linguistics.

Ann Clifton and Anoop Sarkar. 2011. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 32–42. Association for Computational Linguistics.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *The Journal of Machine Learning Research*, 3:951–991.

Waleed Ammar Victor Chahuneau Michael Denkowski, Greg Hanneman, Wang Ling Austin Matthews Kenton Murray, Nicola Segall Yulia Tsvetkov, and Alon Lavie Chris Dyer. 2013. The cmu machine translation systems at wmt 2013: Syntax, synthetic translation options, and pseudo-references. In *8th Workshop on Statistical Machine Translation*, page 70.

Markus Dreyer and Daniel Marcu. 2012. Hyter: Meaning-equivalent semantics for translation evaluation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 162–171. Association for Computational Linguistics.

Christopher Dyer, Smaranda Muresan, and Philip Resnik. 2008. Generalizing word lattice translation. Technical report, DTIC Document.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of ACL*.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 406–414. Association for Computational Linguistics.

Fabienne Fritzinger and Alexander Fraser. 2010. How to avoid burning ducks: combining linguistic analysis and corpus statistics for german compound processing. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 224–234. Association for Computational Linguistics.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 690–696, Sofia, Bulgaria, August.

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

Ann Irvine and Chris Callison-Burch. 2013. Supervised bilingual lexicon induction with multiple monolingual signals. In *HLT-NAACL*, pages 518–523.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Philipp Koehn and Kevin Knight. 2001. Knowledge sources for word-level translation models. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, pages 27–35.

Sara Stymne and Nicola Cancedda. 2011. Productive generation of compound words in statistical machine translation. In *Proc. WMT*.

Sara Stymne. 2012. Text harmonization strategies for phrase-based statistical machine translation.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Yulia Tsvetkov and Shuly Wintner. 2012. Extraction of multi-word expressions from small parallel corpora. *Natural Language Engineering*, 18(04):549–573.

Yulia Tsvetkov, Chris Dyer, Lori Levin, and Archna Bhatia. 2013. Generating English determiners in phrase-based translation with synthetic translation options. In *Proc. WMT*.

David Vilar, Jan-T Peter, and Hermann Ney. 2007. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39. Association for Computational Linguistics.

Harnessing Cognitive Features for Sarcasm Detection

Abhijit Mishra[†], **Diptesh Kanojia**[†], **Seema Nagar** [⋆], **Kuntal Dey**[⋆],
Pushpak Bhattacharyya[†]
[†]Indian Institute of Technology Bombay, India
[⋆]IBM Research, India
[†]{abhijitmishra, diptesh, pb}@cse.iitb.ac.in
[⋆]{senagar3, kuntadey}@in.ibm.com

Abstract

In this paper, we propose a novel mechanism for enriching the feature vector, for the task of sarcasm detection, with cognitive features extracted from eye-movement patterns of human readers. Sarcasm detection has been a challenging research problem, and its importance for NLP applications such as review summarization, dialog systems and sentiment analysis is well recognized. Sarcasm can often be traced to *incongruity* that becomes apparent as the full sentence unfolds. This presence of incongruity- implicit or explicit- affects the way readers eyes move through the text. We observe the difference in the behaviour of the eye, while reading sarcastic and non sarcastic sentences. Motivated by this observation, we augment traditional linguistic and stylistic features for sarcasm detection with the cognitive features obtained from readers eye movement data. We perform statistical classification using the enhanced feature set so obtained. The augmented cognitive features improve sarcasm detection by 3.7% (in terms of F-score), over the performance of the best reported system.

1 Introduction

Sarcasm is an intensive, indirect and complex construct that is often intended to express contempt or ridicule [1]. Sarcasm, in speech, is multi-modal, involving tone, body-language and gestures along with linguistic artifacts used in speech. Sarcasm in text, on the other hand, is more restrictive when it comes to such non-linguistic modalities. This makes recognizing textual sarcasm more challenging for both humans and machines.

[1]The Free Dictionary

Sarcasm detection plays an indispensable role in applications like online review summarizers, dialog systems, recommendation systems and sentiment analyzers. This makes automatic detection of sarcasm an important problem. However, it has been quite difficult to solve such a problem with traditional NLP tools and techniques. This is apparent from the results reported by the survey from Joshi et al. (2016). The following discussion brings more insights into this.

Consider a scenario where an online reviewer gives a negative opinion about a movie through sarcasm: *"This is the kind of movie you see because the theater has air conditioning"*. It is difficult for an automatic sentiment analyzer to assign a rating to the movie and, in the absence of any other information, such a system may not be able to comprehend that *prioritizing the air-conditioning facilities of the theater over the movie experience indicates a negative sentiment towards the movie*. This gives an intuition to why, for sarcasm detection, it is necessary to go beyond textual analysis.

We aim to address this problem by exploiting the psycholinguistic side of sarcasm detection, using cognitive features extracted with the help of *eye-tracking*. A motivation to consider cognitive features comes from analyzing human eye-movement trajectories that supports the conjecture: *Reading sarcastic texts induces distinctive eye movement patterns, compared to literal texts.* The cognitive features, derived from human eye movement patterns observed during reading, include two primary feature types:

1. Eye movement characteristic features of readers while reading given text, comprising *gaze-fixaions* (*i.e,*longer stay of gaze on a visual object), forward and backward *saccades* (*i.e.,* quick jumping of gaze between two positions of rest).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1095–1104,
Berlin, Germany, August 7-12, 2016.

2. Features constructed using the statistical and deeper structural information contained in *graph*, created by treating words as vertices and saccades between a pair of words as edges.

The cognitive features, along with textual features used in best available sarcasm detectors, are used to train binary classifiers against given sarcasm labels. Our experiments show significant improvement in classification accuracy over the state of the art, by performing such augmentation.

Feasibility of Our Approach

Since our method requires gaze data from human readers to be available, the methods practicability becomes questionable. We present our views on this below.

Availability of Mobile Eye-trackers

Availability of inexpensive embedded eye-trackers on hand-held devices has come close to reality now. This opens avenues to get eye-tracking data from inexpensive mobile devices from a huge population of online readers non-intrusively, and derive cognitive features to be used in predictive frameworks like ours. For instance, *Cogisen: (http://www.sencogi.com)* has a patent (ID: EP2833308-A1) on "eye-tracking using inexpensive mobile web-cams".

Applicability Scenario

We believe, mobile eye-tracking modules could be a part of mobile applications built for e-commerce, online learning, gaming *etc.* where automatic analysis of online reviews calls for better solutions to detect linguistic nuances like sarcasm. To give an example, let's say a book gets different reviews on Amazon. Our system could watch how readers read the review using mobile eye-trackers, and thereby, decide whether the text contains sarcasm or not. Such an application can horizontally scale across the web and will help in improving automatic classification of online reviews.

Since our approach seeks human mediation, one might be tempted to question the approach of relying upon eye-tracking, an indirect indicator, instead of directly obtaining man-made annotations. We believe, asking a large number of internet audience to annotate/give feedback on each and every sentence that they read online, following a set of annotation instructions, will be extremely intrusive and may not be responded well. Our system, on the other hand, can be seamlessly integrated into existing applications and as the eye-tracking process runs in the background, users will not be interrupted in the middle of the reading. This, thus, offers a more natural setting where human mediation can be availed without intervention.

Getting Users' Consent for Eye-tracking

Eye-tracking technology has already been utilized by leading mobile technology developers (like Samsung) to facilitate richer user experiences through services like *Smart-scroll* (where a user's eye movement determines whether a page has to be scrolled or not) and *Smart-lock* (where user's gaze position decides whether to lock the screen or not). The growing interest of users in using such services takes us to a promising situation where getting users' consent to record eye-movement patterns will not be difficult, though it is yet not the current state of affairs.

Disclaimer: In this work, we focus on detecting sarcasm in *non-contextual* and *short-text* settings prevalent in product reviews and social media. Moreover, our method requires eye-tracking data to be available in the test scenario.

2 Related Work

Sarcasm, in general, has been the focus of research for quite some time. In one of the pioneering works Jorgensen et al. (1984) explained how sarcasm arises when a figurative meaning is used opposite to the literal meaning of the utterance. In the word of Clark and Gerrig (1984), sarcasm processing involves canceling the indirectly negated message and replacing it with the implicated one. Giora (1995), on the other hand, define sarcasm as a mode of indirect negation that requires processing of both negated and implicated messages. Ivanko and Pexman (2003) define sarcasm as a six tuple entity consisting of *a speaker, a listener, Context, Utterance, Literal Proposition* and *Intended Proposition* and study the cognitive aspects of sarcasm processing.

Computational linguists have previously addressed this problem using rule based and statistical techniques, that make use of : **(a)** Unigrams and Pragmatic features (Carvalho et al., 2009; González-Ibánez et al., 2011; Barbieri et al., 2014; Joshi et al., 2015) **(b)** Stylistic patterns (Davidov et al., 2010) and patterns related to *situational disparity* (Riloff et al., 2013) and **(c)** Hastag

interpretations (Liebrecht et al., 2013; Maynard and Greenwood, 2014).

Most of the previously done work on sarcasm detection uses *distant supervision* based techniques (ex: leveraging *hashtags*) and stylistic/pragmatic features (emoticons, laughter expressions such as *"lol" etc*). But, detecting sarcasm in linguistically well-formed structures, in absence of explicit cues or information (like emoticons), proves to be hard using such linguistic/stylistic features alone.

With the advent of sophisticated eye-trackers and electro/magneto-encephalographic (EEG/MEG) devices, it has been possible to delve deep into the cognitive underpinnings of sarcasm understanding. Filik (2014), using a series of eye-tracking and EEG experiments try to show that for unfamiliar ironies, the literal interpretation would be computed first. They also show that a mismatch with context would lead to a re-interpretation of the statement, as being ironic. Camblin et al. (2007) show that in multi-sentence passages, discourse congruence has robust effects on eye movements. This also implies that disrupted processing occurs for discourse incongruent words, even though they are perfectly congruous at the sentence level. In our previous work (Mishra et al., 2016), we augment cognitive features, derived from eye-movement patterns of readers, with textual features to detect whether a human reader has realized the presence of sarcasm in text or not.

The recent advancements in the literature discussed above, motivate us to explore gaze-based cognition for sarcasm detection. As far as we know, our work is the first of its kind.

3 Eye-tracking Database for Sarcasm Analysis

Sarcasm often emanates from *incongruity* (Campbell and Katz, 2012), which enforces the brain to reanalyze it (Kutas and Hillyard, 1980). This, in turn, affects the way eyes move through the text. Hence, **distinctive eye-movement patterns may be observed in the case of successful processing of sarcasm in text in contrast to literal texts**. This hypothesis forms the crux of our method for sarcasm detection and we validate this using our previously released freely available sarcasm dataset[2] (Mishra et al., 2016) enriched with gaze information.

[2]http://www.cfilt.iitb.ac.in/cognitive-nlp

	μ_S	σ_S	μ_NS	σ_NS	t	p
P1	319	145	196	97	14.1	5.84E-39
P2	415	192	253	130	14.0	1.71E-38
P3	322	173	214	160	9.5	3.74E-20
P4	328	170	191	96	13.9	1.89E-37
P5	291	151	183	76	11.9	2.75E-28
P6	230	118	136	84	13.2	6.79E-35
P7	488	268	252	141	15.3	3.96E-43

Table 1: T-test statistics for average fixation duration time per word (in ms) for presence of sarcasm (represented by *S*) and its absence (*NS*) for participants P1-P7.

3.1 Document Description

The database consists of 1,000 short texts, each having 10-40 words. Out of these, 350 are sarcastic and are collected as follows: (a) 103 sentences are from two popular sarcastic quote websites[3], (b) 76 sarcastic short movie reviews are manually extracted from the *Amazon Movie Corpus* (Pang and Lee, 2004) by two linguists. (c) 171 tweets are downloaded using the hashtag *#sarcasm* from Twitter. The 650 non-sarcastic texts are either downloaded from Twitter or extracted from the Amazon Movie Review corpus. The sentences do not contain words/phrases that are *highly* topic or culture specific. The tweets were normalized to make them linguistically well formed to avoid difficulty in interpreting social media lingo. Every sentence in our dataset carries positive or negative opinion about specific "aspects". For example, the sentence *"The movie is extremely well cast"* has positive sentiment about the aspect "cast".

The annotators were seven graduate students with science and engineering background, and possess good English proficiency. They were given a set of instructions beforehand and are advised to seek clarifications before they proceed. The instructions mention the nature of the task, annotation input method, and necessity of head movement minimization during the experiment.

3.2 Task Description

The task assigned to annotators was to read sentences one at a time and label them with with binary labels indicating the polarity (*i.e.,* positive/negative). Note that, the participants were not

[3]http://www.sarcasmsociety.com, http://www.themarysue.com/funny-amazon-reviews

instructed to annotate whether a sentence is sarcastic or not., to rule out the *Priming Effect* (*i.e.,* if sarcasm is expected beforehand, processing incongruity becomes relatively easier (Gibbs, 1986)). The setup ensures its "ecological validity" in two ways: *(1)* Readers are not given any clue that they have to treat sarcasm with special attention. This is done by setting the task to polarity annotation (instead of sarcasm detection). *(2)* Sarcastic sentences are mixed with non sarcastic text, which does not give prior knowledge about whether the forthcoming text will be sarcastic or not.

The eye-tracking experiment is conducted by following the standard norms in eye-movement research (Holmqvist et al., 2011). At a time, one sentence is displayed to the reader along with the "aspect" with respect to which the annotation has to be provided. While reading, an SR-Research Eyelink-1000 eye-tracker (monocular remote mode, sampling rate 500Hz) records several eye-movement parameters like fixations (a long stay of gaze) and saccade (quick jumping of gaze between two positions of rest) and pupil size.

The accuracy of polarity annotation varies between 72%-91% for sarcastic texts and 75%-91% for non-sarcastic text, showing the inherent difficulty of sentiment annotation, when sarcasm is present in the text under consideration. Annotation errors may be attributed to: (a) lack of patience/attention while reading, (b) issues related to text comprehension, and (c) confusion/indecisiveness caused due to lack of context.

For our analysis, we do not discard the incorrect annotations present in the database. Since our system eventually aims to involve online readers for sarcasm detection, it will be hard to segregate readers who misinterpret the text. We make a rational assumption that, for a particular text, most of the readers, from a fairly large population, will be able to identify sarcasm. Under this assumption, the eye-movement parameters, averaged across all readers in our setting, may not be significantly distorted by a few readers who would have failed to identify sarcasm. This assumption is applicable for both regular and multi-instance based classifiers explained in section 6.

4 Analysis of Eye-movement Data

We observe distinct behavior during sarcasm reading, by analyzing the "fixation duration on the text" (also referred to as "dwell time" in the literature) and "scanpaths" of the readers.

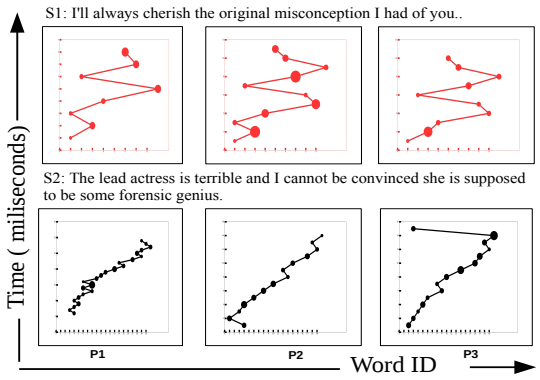


Figure 1: Scanpaths of three participants for two negatively polar sentences sentence *S1* and *S2*. Sentence *S1* is sarcastic but *S2* is not.

4.1 Variation in the Average Fixation Duration per Word

Since sarcasm in text can be expected to induce cognitive load, it is reasonable to believe that it would require more processing time (Ivanko and Pexman, 2003). Hence, fixation duration normalized over total word count should usually be higher for a sarcastic text than for a non-sarcastic one. We observe this for all participants in our dataset, with the *average fixation duration per word* for sarcastic texts being at least 1.5 times more than that of non-sarcastic texts. To test the statistical significance, we conduct a two-tailed t-test (assuming unequal variance) to compare the average fixation duration per word for sarcastic and non-sarcastic texts. The hypothesized mean difference is set to 0 and the error tolerance limit (α) is set to 0.05. The t-test analysis, presented in Table 1, shows that for all participants, a statistically significant difference exists between the average fixation duration per word for sarcasm (higher average fixation duration) and non-sarcasm (lower average fixation duration). This affirms that the presence of sarcasm affects the duration of fixation on words.

It is important to note that longer fixations may also be caused by other linguistic subtleties (such as difficult words, ambiguity and syntactically complex structures) causing delay in comprehension, or occulomotor control problems forcing readers to spend time adjusting eye-muscles. So, an elevated average fixation duration per word may not sufficiently indicate the presence of sarcasm. But we would also like to share that, for our

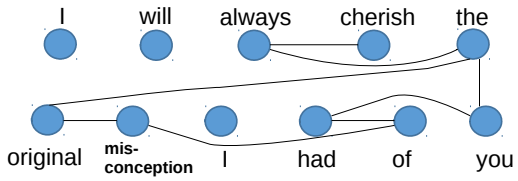


Figure 2: Saliency graph of participant *P1* for the sentence *I will always cherish the original misconception I had of you.*

dataset, when we considered *readability* (Flesch readability ease-score (Flesch, 1948)), *number of words in a sentence* and *average character per word* along with the *sarcasm label* as the predictors of average fixation duration following a linear mixed effect model (Barr et al., 2013), *sarcasm label* turned out to be the most significant predictor with a maximum slope. This indicates that average fixation duration per word has a strong connection with the text being sarcastic, at least in our dataset.

We now analyze *scanpaths* to gain more insights into the sarcasm comprehension process.

4.2 Analysis of Scanpaths

Scanpaths are line-graphs that contain fixations as nodes and saccades as edges; the radii of the nodes represent the fixation duration. A scanpath corresponds to a participant's eye-movement pattern while reading a particular sentence. Figure 1 presents scanpaths of three participants for the sarcastic sentence *S1* and the non-sarcastic sentence *S2*. The x-axis of the graph represents the sequence of words a reader reads, and the y-axis represents a temporal sequence in milliseconds.

Consider a sarcastic text containing incongruous phrases *A* and *B*. Our qualitative scanpath-analysis reveals that scanpaths with respect to sarcasm processing have two typical characteristics. Often, a long *regression* - a saccade that goes to a previously visited segment - is observed when a reader starts reading *B* after skimming through *A*. In a few cases, the fixation duration on *A* and *B* are significantly higher than the average fixation duration per word. In sentence *S1*, we see long and multiple regressions from the two incongruous phrases *"misconception"* and *"cherish"*, and a few instances where phrases *"always cherish"* and *"original misconception"* are fixated longer than usual. Such eye-movement behaviors are not seen for *S2*.

Though sarcasm induces distinctive scanpaths like the ones depicted in Figure 1 in the observed examples, presence of such patterns is not sufficient to guarantee sarcasm; such patterns may also possibly arise from literal texts. We believe that a combination of linguistic features, readability of text and features derived from scanpaths would help discriminative machine learning models learn sarcasm better.

5 Features for Sarcasm Detection

We describe the features used for sarcasm detection in Table 2. The features enlisted under *lexical*,*implicit incongruity* and *explicit incongruity* are borrowed from various literature (predominantly from Joshi et al. (2015)). These features are essential to separate sarcasm from other forms semantic incongruity in text (for example ambiguity arising from *semantic ambiguity* or from *metaphors*). Two additional *textual* features *viz. readability* and *word count* of the text are also taken under consideration. These features are used to reduce the effect of text hardness and text length on the eye-movement patterns.

5.1 Simple Gaze Based Features

Readers' eye-movement behavior, characterized by fixations, forward saccades, skips and regressions, can be directly quantified by simple statistical aggregation (*i.e.,* either computing features for individual participants and then averaging or performing a multi-instance based learning as explained in section 6). Since these eye-movement attributes relate to the cognitive process in reading (Rayner and Sereno, 1994), we consider these as features in our model. Some of these features have been reported by Mishra et al. (2016) for modeling sarcasm understandability of readers. However, as far as we know, these features are being introduced in NLP tasks like textual sarcasm detection for the first time. The values of these features are believed to increase with the increase in the degree of surprisal caused by incongruity in text (except *skip count*, which will decrease).

5.2 Complex Gaze Based Features

For these features, we rely on a graph structure, namely "saliency graphs", derived from eye-gaze information and word sequences in the text.

Constructing Saliency Graphs:

For each reader and each sentence, we construct a "saliency graph", representing the reader's atten-

Subcategory	Feature Name	Type	Intent
Category: Textual Sarcasm Features, Source: Joshi et. al.			
Lexical	Presence of Unigrams (UNI)	Boolean	Unigrams in the training corpus
	Punctuations (PUN)	Real	Count of punctuation marks
Implicit Incongruity	Implicit Incongruity (IMP)	Boolean	Incongruity of extracted implicit phrases (Rilof et.al, 2013)
Explicit Incongruity	Explicit Incongruity (EXP)	Integer	Number of times a word follows a word of opposite polarity
	Largest Pos/Neg Subsequence (LAR)	Integer	Length of the largest series of words with polarities unchanged
	Positive words (+VE)	Integer	Number of positive words
	Negative words (-VE)	Integer	Number of negative words
	Lexical Polarity (LP)	Integer	Sentence polarity found by supervised logistic regression
Category: Cognitive Features. We introduce these features for sarcasm detection.			
Textual	Readability (RED)	Real	Flesch Readability Ease (Flesch, 1948) score of the sentence
	Number of Words (LEN)	Integer	Number of words in the sentence
Simple Gaze Based	Avg. Fixation Duration (FDUR)	Real	Sum of fixation duration divided by word count
	Avg. Fixation Count (FC)	Real	Sum of fixation counts divided by word count
	Avg. Saccade Length (SL)	Real	Sum of saccade lengths (measured by number of words) divided by word count
	Regression Count (REG)	Real	Total number of gaze regressions
	Skip count (SKIP)	Real	Number of words skipped divided by total word count
	Count of regressions from second half to first half of the sentence (RSF)	Real	Number of regressions from second half of the sentence to the first half of the sentence (given the sentence is divided into two equal half of words)
	Largest Regression Position (LREG)	Real	Ratio of the absolute position of the word from which a regression with the largest amplitude (number of pixels) is observed, to the total word count of sentence
Complex Gaze Based	Edge density of the saliency gaze graph (ED)	Real	Ratio of the number of directed edges to vertices in the saliency gaze graph (SGG)
	Fixation Duration at Left/Source (F1H, F1S)	Real	Largest weighted degree (LWD) and second largest weighted degree (SWD) of the SGG considering the fixation duration of word i of edge E_{ij}
	Fixation Duration at Right/Target (F2H, F2S)	Real	LWD and SWD of the SGG considering the fixation duration of word j of edge E_{ij}
	Forward Saccade Word Count of Source (PSH, PSS)	Real	LWD and SWD of the SGG considering the number of forward saccades between words i and j of an edge E_{ij}
	Forward Saccade Word Count of Destination (PSDH, PSDS)	Real	LWD and SWD of the SGG considering the total distance (word count) of forward saccades between words i and j of an edge E_{ij}
	Regressive Saccade Word Count of Source (RSH, RSS)	Real	LWD and SWD of the SGG considering the number of regressive saccades between words i and j of an edge E_{ij}
	Regressive Saccade Word Count of Destination (RSDH, RSDS)	Real	LWD and SWD of the SGG considering the total distance (word count) of regressive saccades between words i and j of an edge E_{ij}

Table 2: The complete set of features used in our system.

tion characteristics. A saliency graph for a sentence S for a reader R, represented as $G = (V, E)$, is a graph with vertices (V) and edges (E) where each vertex $v \in V$ corresponds to a word in S (may not be unique) and there exists an edge $e \in E$ between vertices v_1 and v_2 if R performs at least one saccade between the words corresponding to $v1$ and $v2$.

Figure 2 shows an example of a saliency graph.A saliency graph may be weighted, but not necessarily connected, for a given text (as there may be words in the given text with no fixation on them). The "complex" gaze features derived from saliency graphs are also motivated by the theory of incongruity. For instance, *Edge Density* of a saliency graph increases with the number of distinct saccades, which could arise from the complexity caused by presence of sarcasm. Similarly, the highest weighted degree of a graph is expected to be higher, if the node corresponds to a phrase, incongruous to some other phrase in the text.

6 The Sarcasm Classifier

We interpret sarcasm detection as a binary classification problem. The training data constitutes

Features	P(1)	P(-1)	P(avg)	R(1)	R(-1)	R(avg)	F(1)	F(-1)	F(avg)	Kappa
Multi Layered Neural Network										
Unigram	53.1	74.1	66.9	51.7	75.2	66.6	52.4	74.6	66.8	0.27
Sarcasm (Joshi et. al.)	59.2	75.4	69.7	51.7	80.6	70.4	55.2	77.9	69.9	0.33
Gaze	62.4	76.7	71.7	54	82.3	72.3	57.9	79.4	71.8	0.37
Gaze+Sarcasm	63.4	75	70.9	48	84.9	71.9	54.6	79.7	70.9	0.34
Näive Bayes										
Unigram	45.6	82.4	69.4	81.4	47.2	59.3	58.5	60	59.5	0.24
Sarcasm (Joshi et. al.)	46.1	81.6	69.1	79.4	49.5	60.1	58.3	61.6	60.5	0.25
Gaze	57.3	82.7	73.8	72.9	70.5	71.3	64.2	76.1	71.9	0.41
Gaze+Sarcasm	46.7	82.1	69.6	79.7	50.5	60.8	58.9	62.5	61.2	0.26
Original system by Riloff et.al. : Rule Based with implicit incongruity										
Ordered	60	30	49	50	39	46	54	34	47	0.10
Unordered	56	28	46	40	42	41	46	33	42	0.16
Original system by Joshi et.al. : SVM with RBF Kernel										
Sarcasm (Joshi et. al.)	73.1	69.4	70.7	22.6	95.5	69.8	34.5	80.4	64.2	0.21
SVM Linear: with default parameters										
Unigram	56.5	77	69.8	58.6	75.5	69.5	57.5	76.2	69.6	0.34
Sarcasm (Joshi et. al.)	*59.9*	*78.7*	*72.1*	*61.4*	*77.6*	*71.9*	*60.6*	*78.2*	*72*	*0.39*
Gaze	**65.9**	75.9	72.4	49.7	86	73.2	56.7	80.6	72.2	0.38
Gaze+Sarcasm	63.7	79.5	74	61.7	80.9	74.1	62.7	80.2	74	0.43
Multi Instance Logistic Regression: Best Performing Classifier										
Gaze	65.3	77.2	73	53	**84.9**	73.8	58.5	**80.8**	73.1	0.41
Gaze+Sarcasm	62.5	**84**	**76.5**	**72.6**	76.7	**75.3**	**67.2**	80.2	**75.7**	**0.47**

Table 3: Classification results for different feature combinations. P→ Precision, R→ Recall, F→ F˙score, Kappa→ Kappa statistics show *agreement with the gold labels*. Subscripts 1 and -1 correspond to sarcasm and non-sarcasm classes respectively.

Sentence	Gold	Sarcasm	Gaze	Gaze+Sarcasm
1. I would like to live in Manchester, England. The transition between Manchester and death would be unnoticeable.	S	NS	S	S
2. Helped me a lot with my panic attacks. I took 6 mg a day for almost 20 years. Can't stop of course but it makes me feel very comfortable.	NS	S	NS	NS
3. Forgot to bring my headphones to the gym this morning, the music they play in this gym pumps me up so much!	S	S	NS	NS
4. Best show on satellite radio!! No doubt about it. The little doggy company has nothing even close.	NS	S	NS	S

Table 4: Example test-cases with *S* and *NS* representing labels for sarcastic and not-sarcastic respectively.

994 examples created using our eye-movement database for sarcasm detection. To check the effectiveness of our feature set, we observe the performance of multiple classification techniques on our dataset through a *stratified* 10-fold cross validation. We also compare the classification accuracy of our system and the best available systems proposed by Riloff et al. (2013) and Joshi et al. (2015) on our dataset. Using Weka (Hall et al., 2009) and LibSVM (Chang and Lin, 2011) APIs, we implement the following classifiers:

- Näive Bayes classifier
- Support Vector Machines (Cortes and Vapnik, 1995) with default hyper-paramaters
- Multilayer Feed Forward Neural Network
- Multi Instance Logistic Regression (MILR) (Xu and Frank, 2004)

6.1 Results

Table 3 shows the classification results considering various feature combinations for different classifiers and other systems. These are:

- *Unigram* (with principal components of unigram feature vectors),
- *Sarcasm* (the feature-set reported by Joshi et al. (2015) subsuming unigram features and features from other reported systems)
- *Gaze* (the simple and complex cognitive features we introduce, along with readability and word count features), and
- *Gaze+Sarcasm* (the complete set of features).

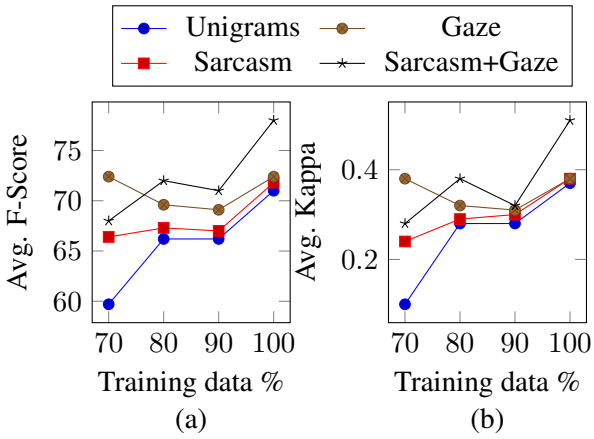


Figure 3: Effect of training data size on classification in terms of (a) F-score and (b) *Kappa* statistics

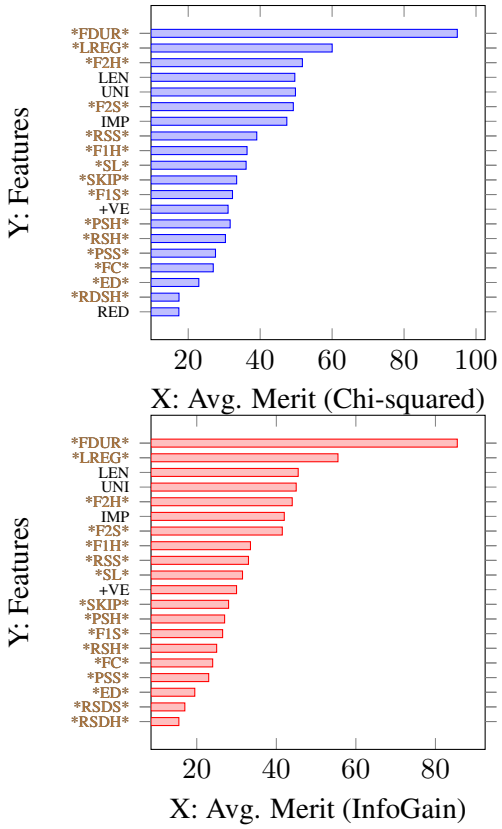


Figure 4: Significance of features observed by ranking the features using *Attribute Evaluation based on Information Gain* and *Attribute Evaluation based on Chi-squared test*. The length of the bar corresponds to the average merit of the feature. Features marked with * are gaze features.

For all regular classifiers, the gaze features are averaged across participants and augmented with linguistic and sarcasm related features. For the MILR classifier, the gaze features derived from each participant are augmented with linguistic features and thus, a multi instance "bag" of features is formed for each sentence in the training data. This multi-instance dataset is given to an MILR classifier, which follows the *standard multi instance assumption* to derive class-labels for each bag.

For all the classifiers, our feature combination outperforms the baselines (considering only unigram features) as well as (Joshi et al., 2015), with the MILR classifier getting an F-score improvement of **3.7%** and *Kappa* difference of **0.08**. We also achieve an improvement of **2%** over the baseline, using SVM classifier, when we employ our feature set. We also observe that the gaze features alone, also capture the differences between sarcasm and non-sarcasm classes with a high-precision but a low recall.

To see if the improvement obtained is statistically significant over the state-of-the art system with textual sarcasm features alone, we perform **McNemar test**. The output of the SVM classifier using only linguistic features used for sarcasm detection by Joshi et al. (2015) and the output of the MILR classifier with the complete set of features are compared, setting threshold $\alpha = 0.05$. There was a significant difference in the classifier's accuracy with **p(two-tailed) = 0.02** with an odds-ratio of **1.43**, showing that the classification accuracy improvement is unlikely to be observed by chance in 95% confidence interval.

6.2 Considering Reading Time as a Cognitive Feature along with Sarcasm Features

One may argue that, considering simple measures of reading effort like "reading time" as cognitive feature instead of the expensive eye-tracking features for sarcasm detection may be a cost-effective solution. To examine this, we repeated our experiments with "reading time" considered as the only cognitive feature, augmented with the textual features. The F-scores of all the classifiers turn out to be close to that of the classifiers considering sarcasm feature alone and the difference in the improvement is not statistically significant ($p > 0.05$). One the other hand, F-scores with gaze features are superior to the F-scores when reading time is considered as a cognitive feature.

6.3 How Effective are the Cognitive Features

We examine the effectiveness of cognitive features on the classification accuracy by varying the input training data size. To examine this, we create a

stratified (keeping the class ratio constant) random train-test split of 80%:20%. We train our classifier with 100%, 90%, 80% and 70% of the training data with our whole feature set, and the feature combination from Joshi et al. (2015). The goodness of our system is demonstrated by improvements in F-score and Kappa statistics, shown in Figure 3.

We further analyze the importance of features by ranking the features based on **(a)** Chi squared test, and **(b)** Information Gain test, using Weka's attribute selection module. Figure 4 shows the top 20 ranked features produced by both the tests. For both the cases, we observe 16 out of top 20 features to be gaze features. Further, in each of the cases, *Average Fixation Duration per Word* and *Largest Regression Position* are seen to be the two most significant features.

6.4 Example Cases

Table 4 shows a few example cases from the experiment with stratified 80%-20% train-test split.

- Example sentence 1 is sarcastic, and requires extra-linguistic knowledge (about poor living conditions at Manchester). Hence, the sarcasm detector relying only on textual features is unable to detect the underlying incongruity. However, our system predicts the label successfully, possibly helped by the gaze features.

- Similarly, for sentence 2, the false sense of presence of incongruity (due to phrases like "Helped me" and "Can't stop") affects the system with only linguistic features. Our system, though, performs well in this case also.

- Sentence 3 presents a false-negative case where it was hard for even humans to get the sarcasm. This is why our gaze features (and subsequently the complete set of features) account for erroneous prediction.

- In sentence 4, gaze features alone false-indicate presence of incongruity, whereas the system predicts correctly when gaze and linguistic features are taken together.

From these examples, it can be inferred that, only gaze features would not have sufficed to rule out the possibility of detecting other forms of incongruity that do not result in sarcasm.

6.5 Error Analysis

Errors committed by our system arise from multiple factors, starting from limitations of the eye-tracker hardware to errors committed by linguistic tools and resources. Also, aggregating various eye-tracking parameters to extract the cognitive features may have caused information loss in the regular classification setting.

7 Conclusion

In the current work, we created a novel framework to detect sarcasm, that derives insights from human cognition, that manifests over eye movement patterns. We hypothesized that distinctive eye-movement patterns, associated with reading sarcastic text, enables improved detection of sarcasm. We augmented traditional linguistic features with cognitive features obtained from readers' eye-movement data in the form of simple gaze-based features and complex features derived from a graph structure. This extended feature-set improved the success rate of the sarcasm detector by 3.7%, over the best available system. Using cognitive features in an NLP Processing system like ours is the first proposal of its kind.

Our general approach may be useful in other NLP sub-areas like sentiment and emotion analysis, text summarization and question answering, where considering textual clues alone does not prove to be sufficient. We propose to augment this work in future by exploring deeper graph and gaze features. We also propose to develop models for the purpose of learning complex gaze feature representation, that accounts for the power of individual eye movement patterns along with the aggregated patterns of eye movements.

Acknowledgments

We thank the members of CFILT Lab, especially Jaya Jha and Meghna Singh, and the students of IIT Bombay for their help and support.

References

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. *ACL 2014*, page 50.

Dale J Barr, Roger Levy, Christoph Scheepers, and Harry J Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of memory and language*, 68(3):255–278.

C. Christine Camblin, Peter C. Gordon, and Tamara Y. Swaab. 2007. The interplay of discourse congruence and lexical association during sentence processing: Evidence from {ERPs} and eye tracking. *Journal of Memory and Language*, 56(1):103 – 128.

John D Campbell and Albert N Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes*, 49(6):459–480.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Herbert H Clark and Richard J Gerrig. 1984. On the pretense theory of irony.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine learning*, 20(3):273–297.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics.

Hartmut; Wallington Katie; Page Jemma Filik, Ruth; Leuthold. 2014. Testing theories of irony processing using eye-tracking and erps. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 40(3):811–828.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Raymond W. Gibbs. 1986. Comprehension and memory for nonliteral utterances: The problem of sarcastic indirect requests. *Acta Psychologica*, 62(1):41 – 57.

Rachel Giora. 1995. On irony and negation. *Discourse processes*, 19(2):239–264.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. 2009. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18.

Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. 2011. *Eye tracking: A comprehensive guide to methods and measures*. Oxford University Press.

Stacey L Ivanko and Penny M Pexman. 2003. Context incongruity and irony processing. *Discourse Processes*, 35(3):241–279.

Julia Jorgensen, George A Miller, and Dan Sperber. 1984. Test of the mention theory of irony. *Journal of Experimental Psychology: General*, 113(1):112.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. Harnessing context incongruity for sarcasm detection. *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics, Beijing, China*, page 757.

Aditya Joshi, Pushpak Bhattacharyya, and Mark James Carman. 2016. Automatic sarcasm detection: A survey. *CoRR*, abs/1602.03426.

Marta Kutas and Steven A Hillyard. 1980. Reading senseless sentences: Brain potentials reflect semantic incongruity. *Science*, 207(4427):203–205.

Christine Liebrecht, Florian Kunneman, and Antal van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets# not. *WASSA 2013*, page 29.

Diana Maynard and Mark A Greenwood. 2014. Who cares about sarcastic tweets? investigating the impact of sarcasm on sentiment analysis. In *Proceedings of LREC*.

Abhijit Mishra, Diptesh Kanojia, and Pushpak Bhattacharyya. 2016. Predicting readers' sarcasm understandability by modeling gaze behavior. In *Proceedings of AAAI*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, page 271. Association for Computational Linguistics.

Keith Rayner and Sara C Sereno. 1994. Eye movements in reading: Psycholinguistic studies.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *EMNLP*, pages 704–714.

Xin Xu and Eibe Frank. 2004. Logistic regression and boosting for labeled bags of instances. In *Advances in knowledge discovery and data mining*, pages 272–281. Springer.

End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures

Makoto Miwa
Toyota Technological Institute
Nagoya, 468-8511, Japan
makoto-miwa@toyota-ti.ac.jp

Mohit Bansal
Toyota Technological Institute at Chicago
Chicago, IL, 60637, USA
mbansal@ttic.edu

Abstract

We present a novel end-to-end neural model to extract entities and relations between them. Our recurrent neural network based model captures both word sequence and dependency tree substructure information by stacking bidirectional tree-structured LSTM-RNNs on bidirectional sequential LSTM-RNNs. This allows our model to jointly represent both entities and relations with shared parameters in a single model. We further encourage detection of entities during training and use of entity information in relation extraction via entity pretraining and scheduled sampling. Our model improves over the state-of-the-art feature-based model on end-to-end relation extraction, achieving 12.1% and 5.7% relative error reductions in F1-score on ACE2005 and ACE2004, respectively. We also show that our LSTM-RNN based model compares favorably to the state-of-the-art CNN based model (in F1-score) on nominal relation classification (SemEval-2010 Task 8). Finally, we present an extensive ablation analysis of several model components.

1 Introduction

Extracting semantic relations between entities in text is an important and well-studied task in information extraction and natural language processing (NLP). Traditional systems treat this task as a pipeline of two separated tasks, i.e., named entity recognition (NER) (Nadeau and Sekine, 2007; Ratinov and Roth, 2009) and relation extraction (Zelenko et al., 2003; Zhou et al., 2005), but recent studies show that end-to-end (joint) modeling of entity and relation is important for high performance (Li and Ji, 2014; Miwa and Sasaki, 2014) since relations interact closely with entity information. For instance, to learn that *Toefting* and *Bolton* have an *Organization-Affiliation (ORG-AFF)* relation in the sentence *Toefting transferred to Bolton*, the entity information that *Toefting* and *Bolton* are *Person* and *Organization* entities is important. Extraction of these entities is in turn encouraged by the presence of the context words *transferred to*, which indicate an employment relation. Previous joint models have employed feature-based structured learning. An alternative approach to this end-to-end relation extraction task is to employ automatic feature learning via neural network (NN) based models.

There are two ways to represent relations between entities using neural networks: recurrent/recursive neural networks (RNNs) and convolutional neural networks (CNNs). Among these, RNNs can *directly* represent essential linguistic structures, i.e., word sequences (Hammerton, 2001) and constituent/dependency trees (Tai et al., 2015). Despite this representation ability, for relation classification tasks, the previously reported performance using long short-term memory (LSTM) based RNNs (Xu et al., 2015b; Li et al., 2015) is worse than one using CNNs (dos Santos et al., 2015). These previous LSTM-based systems mostly include limited linguistic structures and neural architectures, and do not model entities and relations jointly. We are able to achieve improvements over state-of-the-art models via end-to-end modeling of entities and relations based on richer LSTM-RNN architectures that incorporate complementary linguistic structures.

Word sequence and tree structure are known to be complementary information for extracting relations. For instance, dependencies between words

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1105–1116,
Berlin, Germany, August 7-12, 2016.

are not enough to predict that *source* and *U.S.* have an *ORG-AFF* relation in the sentence *"This is ...", one U.S. source said*, and the context word *said* is required for this prediction. Many traditional, feature-based relation classification models extract features from both sequences and parse trees (Zhou et al., 2005). However, previous RNN-based models focus on only one of these linguistic structures (Socher et al., 2012).

We present a novel end-to-end model to extract relations between entities on both word sequence and dependency tree structures. Our model allows joint modeling of entities and relations in a single model by using both bidirectional sequential (left-to-right and right-to-left) and bidirectional tree-structured (bottom-up and top-down) LSTM-RNNs. Our model first detects entities and then extracts relations between the detected entities using a single incrementally-decoded NN structure, and the NN parameters are jointly updated using both entity and relation labels. Unlike traditional incremental end-to-end relation extraction models, our model further incorporates two enhancements into training: entity pretraining, which pretrains the entity model, and scheduled sampling (Bengio et al., 2015), which replaces (unreliable) predicted labels with gold labels in a certain probability. These enhancements alleviate the problem of low-performance entity detection in early stages of training, as well as allow entity information to further help downstream relation classification.

On end-to-end relation extraction, we improve over the state-of-the-art feature-based model, with 12.1% (ACE2005) and 5.7% (ACE2004) relative error reductions in F1-score. On nominal relation classification (SemEval-2010 Task 8), our model compares favorably to the state-of-the-art CNN-based model in F1-score. Finally, we also ablate and compare our various model components, which leads to some key findings (both positive and negative) about the contribution and effectiveness of different RNN structures, input dependency relation structures, different parsing models, external resources, and joint learning settings.

2 Related Work

LSTM-RNNs have been widely used for sequential labeling, such as clause identification (Hammerton, 2001), phonetic labeling (Graves and Schmidhuber, 2005), and NER (Hammerton, 2003). Recently, Huang et al. (2015) showed that building a conditional random field (CRF) layer on top of bidirectional LSTM-RNNs performs comparably to the state-of-the-art methods in the part-of-speech (POS) tagging, chunking, and NER.

For relation classification, in addition to traditional feature/kernel-based approaches (Zelenko et al., 2003; Bunescu and Mooney, 2005), several neural models have been proposed in the SemEval-2010 Task 8 (Hendrickx et al., 2010), including embedding-based models (Hashimoto et al., 2015), CNN-based models (dos Santos et al., 2015), and RNN-based models (Socher et al., 2012). Recently, Xu et al. (2015a) and Xu et al. (2015b) showed that the shortest dependency paths between relation arguments, which were used in feature/kernel-based systems (Bunescu and Mooney, 2005), are also useful in NN-based models. Xu et al. (2015b) also showed that LSTM-RNNs are useful for relation classification, but the performance was worse than CNN-based models. Li et al. (2015) compared separate sequence-based and tree-structured LSTM-RNNs on relation classification, using basic RNN model structures.

Research on tree-structured LSTM-RNNs (Tai et al., 2015) fixes the direction of information propagation from bottom to top, and also cannot handle an arbitrary number of typed children as in a typed dependency tree. Furthermore, no RNN-based relation classification model simultaneously uses word sequence and dependency tree information. We propose several such novel model structures and training settings, investigating the simultaneous use of bidirectional sequential and bidirectional tree-structured LSTM-RNNs to jointly capture linear and dependency context for end-to-end extraction of relations between entities.

As for end-to-end (joint) extraction of relations between entities, all existing models are feature-based systems (and no NN-based model has been proposed). Such models include structured prediction (Li and Ji, 2014; Miwa and Sasaki, 2014), integer linear programming (Roth and Yih, 2007; Yang and Cardie, 2013), card-pyramid parsing (Kate and Mooney, 2010), and global probabilistic graphical models (Yu and Lam, 2010; Singh et al., 2013). Among these, structured prediction methods are state-of-the-art on several corpora. We present an improved, NN-based alternative for the end-to-end relation extraction.

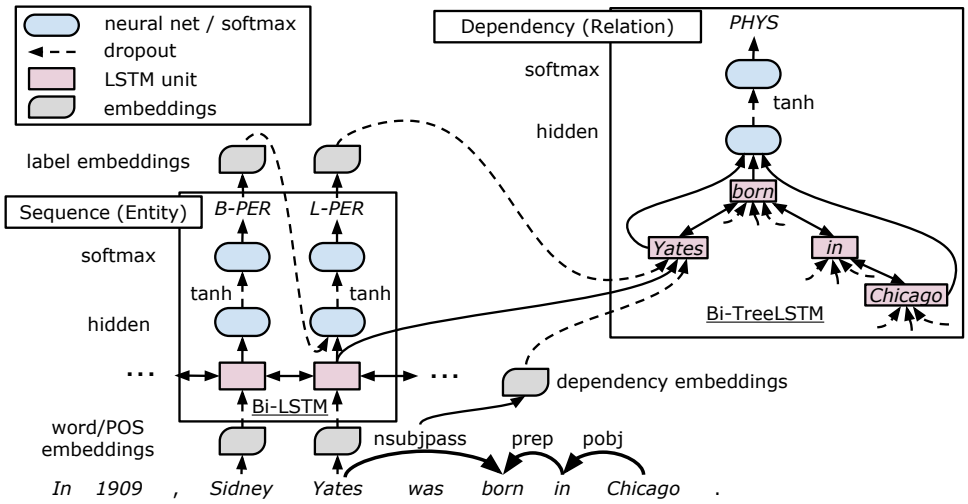


Fig. 1: Our incrementally-decoded end-to-end relation extraction model, with bidirectional sequential and bidirectional tree-structured LSTM-RNNs.

3 Model

We design our model with LSTM-RNNs that represent both word sequences and dependency tree structures, and perform end-to-end extraction of relations between entities on top of these RNNs. Fig. 1 illustrates the overview of the model. The model mainly consists of three representation layers: a word embeddings layer (embedding layer), a word sequence based LSTM-RNN layer (sequence layer), and finally a dependency subtree based LSTM-RNN layer (dependency layer). During decoding, we build greedy, left-to-right entity detection on the sequence layer and realize relation classification on the dependency layers, where each subtree based LSTM-RNN corresponds to a relation candidate between two detected entities. After decoding the entire model structure, we update the parameters simultaneously via backpropagation through time (BPTT) (Werbos, 1990). The dependency layers are stacked on the sequence layer, so the embedding and sequence layers are **shared** by both entity detection and relation classification, and the shared parameters are affected by both entity and relation labels.

3.1 Embedding Layer

The embedding layer handles embedding representations. n_w, n_p, n_d and n_e-dimensional vectors $v^{(w)}$, $v^{(p)}$, $v^{(d)}$ and $v^{(e)}$ are embedded to words, part-of-speech (POS) tags, dependency types, and entity labels, respectively.

3.2 Sequence Layer

The sequence layer represents words in a linear sequence using the representations from the embedding layer. This layer represents sentential context information and maintains entities, as shown in bottom-left part of Fig. 1.

We represent the word sequence in a sentence with bidirectional LSTM-RNNs (Graves et al., 2013). The LSTM unit at t-th word consists of a collection of n_{l_s}-dimensional vectors: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t, and a hidden state h_t. The unit receives an n-dimensional input vector x_t, the previous hidden state h_{t-1}, and the memory cell c_{t-1}, and calculates the new vectors using the following equations:

$$
\begin{aligned}
i_t &= \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}\right), \qquad (1)\\
f_t &= \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}\right),\\
o_t &= \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}\right),\\
u_t &= \tanh\left(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}\right),\\
c_t &= i_t \odot u_t + f_t \odot c_{t-1},\\
h_t &= o_t \odot \tanh(c_t),
\end{aligned}
$$

where σ denotes the logistic function, $\odot$ denotes element-wise multiplication, W and U are weight matrices, and b are bias vectors. The LSTM unit at t-th word receives the concatenation of word and POS embeddings as its input vector: $x_t = \left[v_t^{(w)}; v_t^{(p)}\right]$. We also concatenate the hidden state vectors of the two directions' LSTM units corresponding to each word (denoted as $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$) as

its output vector, $s_t = \left[\overrightarrow{h_t}; \overleftarrow{h_t}\right]$, and pass it to the subsequent layers.

3.3 Entity Detection

We treat entity detection as a sequence labeling task. We assign an entity tag to each word using a commonly used encoding scheme BILOU (Begin, Inside, Last, Outside, Unit) (Ratinov and Roth, 2009), where each entity tag represents the entity type and the position of a word in the entity. For example, in Fig. 1, we assign *B-PER* and *L-PER* (which denote the beginning and last words of a person entity type, respectively) to each word in *Sidney Yates* to represent this phrase as a *PER* (person) entity type.

We perform entity detection on top of the sequence layer. We employ a two-layered NN with an n_{h_e}-dimensional hidden layer $h^{(e)}$ and a softmax output layer for entity detection.

$$h_t^{(e)} = \tanh\left(W^{(e_h)}[s_t; v_{t-1}^{(e)}] + b^{(e_h)}\right) \quad (2)$$

$$y_t = \text{softmax}\left(W^{(e_y)} h_t^{(e)} + b^{(e_y)}\right) \quad (3)$$

Here, W are weight matrices and b are bias vectors.

We assign entity labels to words in a greedy, left-to-right manner.[1] During this decoding, we use the predicted label of a word to predict the label of the next word so as to take label dependencies into account. The NN above receives the concatenation of its corresponding outputs in the sequence layer and the label embedding for its previous word (Fig. 1).

3.4 Dependency Layer

The dependency layer represents a relation between a pair of two target words (corresponding to a relation candidate in relation classification) in the dependency tree, and is in charge of relation-specific representations, as is shown in top-right part of Fig. 1. This layer mainly focuses on the *shortest path* between a pair of target words in the dependency tree (i.e., the path between the least common node and the two target words) since these paths are shown to be effective in relation classification (Xu et al., 2015a). For example, we show the shortest path between *Yates* and *Chicago* in the bottom of Fig. 1, and this path well captures the key phrase of their relation, i.e., *born in*.

[1] We also tried beam search but this did not show improvements in initial experiments.

We employ bidirectional tree-structured LSTM-RNNs (i.e., bottom-up and top-down) to represent a relation candidate by capturing the dependency structure around the target word pair. This bidirectional structure propagates to each node not only the information from the leaves but also information from the root. This is especially important for relation classification, which makes use of argument nodes near the bottom of the tree, and our top-down LSTM-RNN sends information from the top of the tree to such near-leaf nodes (unlike in standard bottom-up LSTM-RNNs).[2] Note that the two variants of tree-structured LSTM-RNNs by Tai et al. (2015) are not able to represent our target structures which have a variable number of typed children: the Child-Sum Tree-LSTM does not deal with types and the N-ary Tree assumes a fixed number of children. We thus propose a new variant of tree-structured LSTM-RNN that shares weight matrices Us for same-type children and also allows variable number of children. For this variant, we calculate n_{l_t}-dimensional vectors in the LSTM unit at t-th node with $C(t)$ children using following equations:

$$i_t = \sigma\left(W^{(i)}x_t + \sum_{l \in C(t)} U_{m(l)}^{(i)} h_{tl} + b^{(i)}\right), \quad (4)$$

$$f_{tk} = \sigma\left(W^{(f)}x_t + \sum_{l \in C(t)} U_{m(k)m(l)}^{(f)} h_{tl} + b^{(f)}\right),$$

$$o_t = \sigma\left(W^{(o)}x_t + \sum_{l \in C(t)} U_{m(l)}^{(o)} h_{tl} + b^{(o)}\right),$$

$$u_t = \tanh\left(W^{(u)}x_t + \sum_{l \in C(t)} U_{m(l)}^{(u)} h_{tl} + b^{(u)}\right),$$

$$c_t = i_t \odot u_t + \sum_{l \in C(t)} f_{tl} \odot c_{tl},$$

$$h_t = o_t \odot \tanh(c_t),$$

where $m(\cdot)$ is a type mapping function.

To investigate appropriate structures to represent relations between two target word pairs, we experiment with three structure options. We primarily employ the shortest path structure (**SP-Tree**), which captures the core dependency path between a target word pair and is widely used in relation classification models, e.g., (Bunescu and

[2] We also tried to use one LSTM-RNN by connecting the root (Paulus et al., 2014), but preparing two LSTM-RNNs showed slightly better performance in our initial experiments.

Mooney, 2005; Xu et al., 2015a). We also try two other dependency structures: **SubTree** and **FullTree**. SubTree is the subtree under the lowest common ancestor of the target word pair. This provides additional modifier information to the path and the word pair in SPTree. FullTree is the full dependency tree. This captures context from the entire sentence. While we use one node type for SPTree, we define two node types for SubTree and FullTree, i.e., one for nodes on shortest paths and one for all other nodes. We use the type mapping function $m(\cdot)$ to distinguish these two nodes types.

3.5 Stacking Sequence and Dependency Layers

We stack the dependency layers (corresponding to relation candidates) on top of the sequence layer to incorporate both word sequence and dependency tree structure information into the output. The dependency-layer LSTM unit at the t-th word receives as input $x_t = \left[s_t; v_t^{(d)}; v_t^{(e)}\right]$, i.e., the concatenation of its corresponding hidden state vectors s_t in the sequence layer, dependency type embedding $v_t^{(d)}$ (denotes the type of dependency to the parent[3]), and label embedding $v_t^{(e)}$ (corresponds to the predicted entity label).

3.6 Relation Classification

We incrementally build relation candidates using all possible combinations of the last words of detected entities, i.e., words with L or U labels in the BILOU scheme, during decoding. For instance, in Fig. 1, we build a relation candidate using *Yates* with an *L-PER* label and *Chicago* with an *U-LOC* label. For each relation candidate, we realize the dependency layer d_p (described above) corresponding to the path between the word pair p in the relation candidate, and the NN receives a relation candidate vector constructed from the output of the dependency tree layer, and predicts its relation label. We treat a pair as a negative relation when the detected entities are wrong or when the pair has no relation. We represent relation labels by type and direction, except for negative relations that have no direction.

The relation candidate vector is constructed as the concatenation $d_p = [\uparrow h_{p_A}; \downarrow h_{p_1}; \downarrow h_{p_2}]$, where $\uparrow h_{p_A}$ is the hidden state vector of the top LSTM unit in the bottom-up LSTM-RNN (representing the lowest common ancestor of the target word pair p), and $\downarrow h_{p_1}$, $\downarrow h_{p_2}$ are the hidden state vectors of the two LSTM units representing the first and second target words in the top-down LSTM-RNN.[4] All the corresponding arrows are shown in Fig. 1.

Similarly to the entity detection, we employ a two-layered NN with an n_{h_r}-dimensional hidden layer $h^{(r)}$ and a softmax output layer (with weight matrices W, bias vectors b).

$$h_p^{(r)} = \tanh\left(W^{(r_h)} d_p + b^{(r_h)}\right) \quad (5)$$

$$y_p = \text{softmax}\left(W^{(r_y)} h_t^{(r)} + b^{(r_y)}\right) \quad (6)$$

We construct the input d_p for relation classification from tree-structured LSTM-RNNs stacked on sequential LSTM-RNNs, so the contribution of sequence layer to the input is indirect. Furthermore, our model uses words for representing entities, so it cannot fully use the entity information. To alleviate these problems, we directly concatenate the average of hidden state vectors for each entity from the sequence layer to the input d_p to relation classification, i.e., $d'_p = \left[d_p; \frac{1}{|I_{p_1}|}\sum_{i\in I_{p_1}} s_i; \frac{1}{|I_{p_2}|}\sum_{i\in I_{p_2}} s_i\right]$ (**Pair**), where I_{p_1} and I_{p_2} represent sets of word indices in the first and second entities.[5]

Also, we assign two labels to each word pair in prediction since we consider both left-to-right and right-to-left directions. When the predicted labels are inconsistent, we select the positive and more confident label, similar to Xu et al. (2015a).

3.7 Training

We update the model parameters including weights, biases, and embeddings by BPTT and Adam (Kingma and Ba, 2015) with gradient clipping, parameter averaging, and L2-regularization (we regularize weights W and U, not the bias terms b). We also apply dropout (Srivastava et al., 2014) to the embedding layer and to the final hidden layers for entity detection and relation classification.

We employ two enhancements, **scheduled sampling** (Bengio et al., 2015) and **entity pretraining**, to alleviate the problem of unreliable prediction of entities in the early stage of training,

[3] We use the dependency to the parent since the number of children varies. Dependency types can also be incorporated into $m(\cdot)$, but this did not help in initial experiments.

[4] Note that the order of the target words corresponds to the direction of the relation, not the positions in the sentence.

[5] Note that we do not show this **Pair** in Fig.1 for simplicity.

and to encourage building positive relation instances from the detected entities. In scheduled sampling, we use gold labels as prediction in the probability of ϵ_i that depends on the number of epochs i during training if the gold labels are legal. As for ϵ_i, we choose the inverse sigmoid decay $\epsilon_i = k/(k + \exp(i/k))$, where $k(\geq 1)$ is a hyper-parameter that adjusts how often we use the gold labels as prediction. Entity pretraining is inspired by (Pentina et al., 2015), and we pretrain the entity detection model using the training data before training the entire model parameters.

4 Results and Discussion

4.1 Data and Task Settings

We evaluate on three datasets: ACE05 and ACE04 for end-to-end relation extraction, and SemEval-2010 Task 8 for relation classification. We use the first two datasets as our primary target, and use the last one to thoroughly analyze and ablate the relation classification part of our model.

ACE05 defines 7 coarse-grained entity types and 6 coarse-grained relation types between entities. We use the same data splits, preprocessing, and task settings as Li and Ji (2014). We report the primary micro F1-scores as well as micro precision and recall on both entity and relation extraction to better explain model performance. We treat an entity as correct when its type and the region of its head are correct. We treat a relation as correct when its type and argument entities are correct; we thus treat all non-negative relations on wrong entities as false positives.

ACE04 defines the same 7 coarse-grained entity types as ACE05 (Doddington et al., 2004), but defines 7 coarse-grained relation types. We follow the cross-validation setting of Chan and Roth (2011) and Li and Ji (2014), and the preprocessing and evaluation metrics of ACE05.

SemEval-2010 Task 8 defines 9 relation types between nominals and a tenth type *Other* when two nouns have none of these relations (Hendrickx et al., 2010). We treat this *Other* type as a negative relation type, and no direction is considered. The dataset consists of 8,000 training and 2,717 test sentences, and each sentence is annotated with a relation between two given nominals. We randomly selected 800 sentences from the training set as our development set. We followed the official task setting, and report the official macro-averaged F1-score (Macro-F1) on the 9 relation types.

For more details of the data and task settings, please refer to the supplementary material.

4.2 Experimental Settings

We implemented our model using the *cnn* library.[6] We parsed the texts using the Stanford neural dependency parser[7] (Chen and Manning, 2014) with the original Stanford Dependencies. Based on preliminary tuning, we fixed embedding dimensions n_w to 200, n_p, n_d, n_e to 25, and dimensions of intermediate layers (n_{l_s}, n_{l_t} of LSTM-RNNs and n_{h_e}, n_{h_r} of hidden layers) to 100. We initialized word vectors via word2vec (Mikolov et al., 2013) trained on Wikipedia[8] and randomly initialized all other parameters. We tuned hyper-parameters using development sets for ACE05 and SemEval-2010 Task 8 to achieve high primary (Micro- and Macro-) F1-scores.[9] For ACE04, we directly employed the best parameters for ACE05. The hyper-parameter settings are shown in the supplementary material. For SemEval-2010 Task 8, we also omitted the entity detection and label embeddings since only target nominals are annotated and the task defines no entity types. Our statistical significance results are based on the Approximate Randomization (AR) test (Noreen, 1989).

4.3 End-to-end Relation Extraction Results

Table 1 compares our model with the state-of-the-art feature-based model of Li and Ji (2014)[10] on final test sets, and shows that our model performs better than the state-of-the-art model.

To analyze the contributions and effects of the various components of our end-to-end relation extraction model, we perform ablation tests on the ACE05 development set (Table 2). The performance slightly degraded without scheduled sampling, and the performance significantly degraded when we removed entity pretraining or removed both ($p<0.05$). This is reasonable because the model can only create relation instances when both of the entities are found and, without these enhancements, it may get too late to find some relations. Removing label embeddings did not affect

[6] https://github.com/clab/cnn

[7] http://nlp.stanford.edu/software/stanford-corenlp-full-2015-04-20.zip

[8] https://dumps.wikimedia.org/enwiki/20150901/

[9] We did not tune the precision-recall trade-offs, but doing so can specifically improve precision further.

[10] Other work on ACE is not comparable or performs worse than the model by Li and Ji (2014).

Corpus	Settings	Entity			Relation		
		P	R	F1	P	R	F1
ACE05	Our Model (SPTree)	0.829	**0.839**	**0.834**	0.572	**0.540**	**0.556**
	Li and Ji (2014)	**0.852**	0.769	0.808	**0.654**	0.398	0.495
ACE04	Our Model (SPTree)	0.808	**0.829**	**0.818**	0.487	**0.481**	**0.484**
	Li and Ji (2014)	**0.835**	0.762	0.797	**0.608**	0.361	0.453

Table 1: Comparison with the state-of-the-art on the ACE05 test set and ACE04 dataset.

Settings	Entity			Relation		
	P	R	F1	P	R	F1
Our Model (SPTree)	0.815	0.821	0.818	0.506	0.529	0.518
−Entity pretraining (EP)	0.793	0.798	0.796	0.494	0.491	0.492*
−Scheduled sampling (SS)	0.812	0.818	0.815	0.522	0.490	0.505
−Label embeddings (LE)	0.811	0.821	0.816	0.512	0.499	0.505
−Shared parameters (Shared)	0.796	0.820	0.808	0.541	0.482	0.510
−EP, SS	0.781	0.804	0.792	0.509	0.479	0.494*
−EP, SS, LE, Shared	0.800	0.815	0.807	0.520	0.452	0.484**

Table 2: Ablation tests on the ACE05 development dataset. * denotes significance at $p<0.05$, ** denotes $p<0.01$.

Settings	Entity			Relation		
	P	R	F1	P	R	F1
SPTree	0.815	0.821	0.818	0.506	0.529	0.518
SubTree	0.812	0.818	0.815	0.525	0.506	0.515
FullTree	0.806	0.816	0.811	0.536	0.507	0.521
SubTree (-SP)	0.803	0.816	0.810	0.533	0.495	0.514
FullTree (-SP)	0.804	0.817	0.811	0.517	0.470	0.492*
Child-Sum	0.806	0.819	0.8122	0.514	0.499	0.506
SPSeq	0.801	0.813	0.807	0.500	0.523	0.511
SPXu	0.809	0.818	0.813	0.494	0.522	0.508

Table 3: Comparison of LSTM-RNN structures on the ACE05 development dataset.

the entity detection performance, but this degraded the recall in relation classification. This indicates that entity label information is helpful in detecting relations.

We also show the performance without sharing parameters, i.e., embedding and sequence layers, for detecting entities and relations (−**Shared parameters**); we first train the entity detection model, detect entities with the model, and build a *separate* relation extraction model using the detected entities, i.e., without entity detection. This setting can be regarded as a pipeline model since two separate models are trained sequentially. Without the shared parameters, both the performance in entity detection and relation classification drops slightly, although the differences are not significant. When we removed all the enhancements, i.e., scheduled sampling, entity pretraining, label embedding, and shared parameters, the performance is significantly worse than SPTree ($p<0.01$), showing that these enhancements provide complementary benefits to end-to-end relation extraction.

Next, we show the performance with different LSTM-RNN structures in Table 3. We first compare the three input dependency structures (SPTree, SubTree, FullTree) for tree-structured LSTM-RNNs. Performances on these three structures are almost same when we distinguish the nodes in the shortest paths from other nodes, but when we do not distinguish them (-SP), the information outside of the shortest path, i.e.,

FullTree (-SP), significantly hurts performance ($p<0.05$). We then compare our tree-structured LSTM-RNN (SPTree) with the Child-Sum tree-structured LSTM-RNN on the shortest path of Tai et al. (2015). Child-Sum performs worse than our SPTree model, but not with as big of a decrease as above. This may be because the difference in the models appears only on nodes that have multiple children and all the nodes except for the least common node have one child.

We finally show results with two counterparts of sequence-based LSTM-RNNs using the shortest path (last two rows in Table 3). **SPSeq** is a bidirectional LSTM-RNN on the shortest path. The LSTM unit receives input from the sequence layer concatenated with embeddings for the surrounding dependency types and directions. We concatenate the outputs of the two RNNs for the relation candidate. **SPXu** is our adaptation of the shortest path LSTM-RNN proposed by Xu et al. (2015b) to match our sequence-layer based model.[11] This has two LSTM-RNNs for the left and right subpaths of the shortest path. We first calculate the max pooling of the LSTM units for each of these two RNNs, and then concatenate the outputs of the pooling for the relation candidate. The comparison with these sequence-based LSTM-RNNs indicates that a tree-structured LSTM-RNN is comparable to sequence-based ones in representing shortest paths.

Overall, the performance comparison of the LSTM-RNN structures in Table 3 show that for end-to-end relation extraction, *selecting the appropriate tree structure representation of the input (i.e., the shortest path) is more important than the choice of the LSTM-RNN structure on that input (i.e., sequential versus tree-based).*

4.4 Relation Classification Analysis Results

To thoroughly analyze the relation classification part alone, e.g., comparing different LSTM structures, architecture components such as hidden layers and input information, and classification task settings, we use the SemEval-2010 Task 8. This dataset, often used to evaluate NN models for relation classification, annotates only relation-related nominals (unlike ACE datasets), so we can focus cleanly on the relation classification part.

[11] This is different from the original one in that we use the sequence layer and we concatenate the embeddings for the input, while the original one prepared individual LSTM-RNNs for different inputs and concatenated their outputs.

Settings	Macro-F1
No External Knowledge Resources	
Our Model (SPTree)	**0.844**
dos Santos et al. (2015)	0.841
Xu et al. (2015a)	0.840
+WordNet	
Our Model (SPTree + WordNet)	0.855
Xu et al. (2015a)	**0.856**
Xu et al. (2015b)	0.837

Table 4: Comparison with state-of-the-art models on SemEval-2010 Task 8 test-set.

Settings	Macro-F1
SPTree	0.851
SubTree	0.839
FullTree	0.829∗
SubTree (-SP)	0.840
FullTree (-SP)	0.828∗
Child-Sum	0.838
SPSeq	0.844
SPXu	0.847

Table 5: Comparison of LSTM-RNN structures on SemEval-2010 Task 8 development set.

We first report official test set results in Table 4. Our novel LSTM-RNN model is comparable to both the state-of-the-art CNN-based models on this task with or without external sources, i.e., WordNet, unlike the previous best LSTM-RNN model (Xu et al., 2015b).[12]

Next, we compare different LSTM-RNN structures in Table 5. As for the three input dependency structures (SPTree, SubTree, FullTree), FullTree performs significantly worse than other structures regardless of whether or not we distinguish the nodes in the shortest paths from the other nodes, which hints that the information outside of the shortest path significantly hurts the performance ($p<0.05$). We also compare our tree-structured LSTM-RNN (SPTree) with sequence-based LSTM-RNNs (SPSeq and SPXu) and tree-structured LSTM-RNNs (Child-Sum). All these LSTM-RNNs perform slightly worse than our SP-

[12] When incorporating WordNet information into our model, we prepared embeddings for WordNet hypernyms extracted by SuperSenseTagger (Ciaramita and Altun, 2006) and concatenated the embeddings to the input vector (the concatenation of word and POS embeddings) of the sequence LSTM. We tuned the dimension of the WordNet embeddings and set it to 15 using the development dataset.

Settings	Macro-F1
SPTree	0.851
−Hidden layer	0.839
−Sequence layer	0.840
−Pair	0.844
−Pair, Sequence layer	0.827∗
Stanford PCFG	0.844
+WordNet	0.854
Left-to-right candidates	0.843
Neg. sampling (Xu et al., 2015a)	0.848

Table 6: Model setting ablations on SemEval-2010 development set.

Tree model, but the differences are small.

Overall, for relation classification, although the performance comparison of the LSTM-RNN structures in Table 5 produces different results on FullTree as compared to the results on ACE05 in Table 3, the trend still holds that selecting the appropriate tree structure representation of the input is more important than the choice of the LSTM-RNN structure on that input.

Finally, Table 6 summarizes the contribution of several model components and training settings on SemEval relation classification. We first remove the hidden layer by directly connecting the LSTM-RNN layers to the softmax layers, and found that this slightly degraded performance, but the difference was small. We then skip the sequence layer and directly use the word and POS embeddings for the dependency layer. Removing the sequence layer[13] or entity-related information from the sequence layer (−Pair) slightly degraded performance, and, on removing both, the performance dropped significantly ($p<0.05$). This indicates that the sequence layer is necessary but the last words of nominals are almost enough for expressing the relations in this task.

When we replace the Stanford neural dependency parser with the Stanford lexicalized PCFG parser (Stanford PCFG), the performance slightly dropped, but the difference was small. This indicates that the selection of parsing models is not critical. We also included WordNet, and this slightly improved the performance (+WordNet), but the difference was small. Lastly, for the generation of relation candidates, generating only left-to-right candidates slightly degraded the performance, but the difference was small and hence the creation of right-to-left candidates was not critical. Treating the inverse relation candidate as a negative instance (Negative sampling) also performed comparably to other generation methods in our model (unlike Xu et al. (2015a), which showed a significance improvement over generating only left-to-right candidates).

[13]Note that this setting still uses some sequence layer information since it uses the entity-related information (Pair).

5 Conclusion

We presented a novel end-to-end relation extraction model that represents both word sequence and dependency tree structures by using bidirectional sequential and bidirectional tree-structured LSTM-RNNs. This allowed us to represent both entities and relations in a single model, achieving gains over the state-of-the-art, feature-based system on end-to-end relation extraction (ACE04 and ACE05), and showing favorably comparable performance to recent state-of-the-art CNN-based models on nominal relation classification (SemEval-2010 Task 8).

Our evaluation and ablation led to three key findings. First, the use of both word sequence and dependency tree structures is effective. Second, training with the shared parameters improves relation extraction accuracy, especially when employed with entity pretraining, scheduled sampling, and label embeddings. Finally, the shortest path, which has been widely used in relation classification, is also appropriate for representing tree structures in neural LSTM models.

Acknowledgments

We thank Qi Li, Kevin Gimpel, and the anonymous reviewers for dataset details and helpful discussions.

References

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *arXiv preprint arXiv:1506.03099*.

Razvan C Bunescu and Raymond Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. ACL.

Yee Seng Chan and Dan Roth. 2011. Exploiting syntactico-semantic structures for relation extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 551–560, Portland, Oregon, USA, June. ACL.

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

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 594–602, Sydney, Australia, July. ACL.

George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. 2004. The automatic content extraction (ace) program – tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*, Lisbon, Portugal, May. European Language Resources Association (ELRA).

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 626–634, Beijing, China, July. ACL.

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

Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE.

James Hammerton. 2001. Clause identification with long short-term memory. In *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, page 22. ACL.

James Hammerton. 2003. Named entity recognition with long short-term memory. In Walter Daelemans and Miles Osborne, editors, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 172–175. ACL.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 268–278, Beijing, China, July. ACL.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden, July. ACL.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212, Uppsala, Sweden, July. ACL.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*, San Diego, CA, May.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. ACL.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal, September. ACL.

Wei Lu and Dan Roth. 2015. Joint mention extraction and classification with mention hypergraphs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 857–867, Lisbon, Portugal, September. ACL.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar, October. ACL.

David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

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

Romain Paulus, Richard Socher, and Christopher D Manning. 2014. Global belief recursive neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2888–2896. Curran Associates, Inc.

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. 2015. Curriculum learning of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR*, pages 5492–5500, Boston, MA, USA, June.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. ACL.

Dan Roth and Wen-Tau Yih, 2007. *Global Inference for Entity and Relation Identification via a Linear Programming Formulation*. MIT Press.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 1–6. ACM.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211, Jeju Island, Korea, July. ACL.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

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

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

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 536–540, Lisbon, Portugal, September. ACL.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks

along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal, September. ACL.

Bishan Yang and Claire Cardie. 2013. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August. ACL.

Xiaofeng Yu and Wai Lam. 2010. Jointly identifying entities and extracting relations in encyclopedia text via a graphical model approach. In *Coling 2010: Posters*, pages 1399–1407, Beijing, China, August. Coling 2010 Organizing Committee.

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *The Journal of Machine Learning Research*, 3:1083–1106.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 427–434, Ann Arbor, Michigan, June. ACL.

A Supplemental Material

A.1 Data and Task Settings

ACE05 defines 7 coarse-grained entity types: Facility (*FAC*), Geo-Political Entities (*GPE*), Location (*LOC*), Organization (*ORG*), Person (*PER*), Vehicle (*VEH*) and Weapon (*WEA*), and 6 coarse-grained relation types between entities: Artifact (*ART*), Gen-Affiliation (*GEN-AFF*), Org-Affiliation (*ORG-AFF*), Part-Whole (*PART-WHOLE*), Person-Social (*PER-SOC*) and Physical (*PHYS*). We removed the *cts*, *un* subsets, and used a 351/80/80 train/dev/test split. We removed duplicated entities and relations, and resolved nested entities. We used head spans for entities. We follow the settings by (Li and Ji, 2014), and we did not use the full mention boundary unlike Lu and Roth (2015). We use *entities* and *relations* to refer to *entity mentions* and *relation mentions* in ACE for brevity.

ACE04 defines the same 7 coarse-grained entity types as ACE05 (Doddington et al., 2004), but defines 7 coarse-grained relation types: *PYS*, *PER-SOC*, Employment / Membership / Subsidiary (*EMP-ORG*), *ART*, *PER/ORG* affiliation (*Other-AFF*), *GPE* affiliation (*GPE-AFF*), and Discourse (*DISC*). We follow the cross-validation setting of Chan and Roth (2011) and Li and Ji (2014). We removed *DISC* and did 5-fold CV on *bnews* and *nwire* subsets (348 documents). We use the same preprocessing and evaluation metrics of ACE05.

SemEval-2010 Task 8 defines 9 relation types between nominals (*Cause-Effect*, *Instrument-Agency*, *Product-Producer*, *Content-Container*, *Entity-Origin*, *Entity-Destination*, *Component-Whole*, *Member-Collection* and *Message-Topic*), and a tenth type *Other* when two nouns have none of these relations (Hendrickx et al., 2010). We treat this *Other* type as a negative relation type, and no direction is considered. The dataset consists of 8,000 training and 2,717 test sentences, and each sentence is annotated with a relation between two given nominals. We randomly selected 800 sentences from the training set as our development set. We followed the official task setting, and report the official macro-averaged F1-score (Macro-F1) on the 9 relation types.

A.2 Hyper-parameter Settings

Here we show the hyper-parameters and the range tried for the hyper-parameters in parentheses. Hyper-parameters include the initial learning rate (5e-3, 2e-3, 1e-3, 5e-4, 2e-4, 1e-4), the regularization parameter (1e-4, 1e-5, 1e-6, 1e-7), dropout probabilities (0.0, 0.1, 0.2, 0.3, 0.4, 0.5), the size of gradient clipping (1, 5, 10, 50, 100), scheduled sampling parameter k (1, 5, 10, 50, 100), the number of epochs for training and entity pretraining ($\leq$ 100), and the embedding dimension of WordNet hypernym (5, 10, 15, 20, 25, 30).

A short proof that $\mathbf{O}_2$ is an MCFL

Mark-Jan Nederhof
School of Computer Science
University of St Andrews, UK

Abstract

We present a new proof that $\mathbf{O}_2$ is a multiple context-free language. It contrasts with a recent proof by Salvati (2015) in its avoidance of concepts that seem specific to two-dimensional geometry, such as the complex exponential function. Our simple proof creates realistic prospects of widening the results to higher dimensions. This finding is of central importance to the relation between extreme free word order and classes of grammars used to describe the syntax of natural language.

1 Introduction

The alphabet of the *MIX language* has three symbols, a, b and c. A string is in the language if and only if the number of a's, the number of b's, and the number of c's are all the same. A different way of defining the MIX language is as permutation closure of the regular language $(abc)^*$, as noted by Bach (1981); see also Pullum (1983).

If a, b and c represent, say, a transitive verb and its subject and its object, then a string in MIX represents a sentence with any number of triples of these constituents, in a hypothetical language with extreme free word order. This is admittedly rather unlike any actual natural language. Joshi (1985) argued that because of this, grammatical formalisms for describing natural languages should *not* be capable of generating MIX. He also conjectured that MIX was beyond the generative capacity of one particular formalism, namely the tree adjoining grammars. Several decades passed before Kanazawa and Salvati (2012) finally proved this conjecture.

MIX has been studied in the context of several other formalisms. Joshi et al. (1991) showed that MIX is generated by a generalization of tree adjoining grammars that decouples local domination for linear precedence. Boullier (1999) showed that MIX is generated by a range concatenation grammar. Negative results were addressed by Sorokin (2014) for well-nested multiple context-free grammars, and by Capelletti and Tamburini (2009) for a class of categorial grammars. The MIX language is also of interest outside of computational linguistics, e.g. in computational group theory (Gilman, 2005).

A considerable advance in the understanding of the MIX language is due to Salvati (2015), who showed that MIX is generated by a multiple context-free grammar (MCFG). The main part of the proof shows that the language $\mathbf{O}_2$ is generated by a MCFG. This language has four symbols, a, $\overline{a}$, b and $\overline{b}$. A string is in the language if and only if the number of a's equals the number of $\overline{a}$'s, and the number of b's equals the number of $\overline{b}$'s. MIX and $\mathbf{O}_2$ are rationally equivalent, which means that if one is generated by a multiple context-free grammar, then so is the other.

The proof by Salvati (2015) is remarkable, in that it is one of the few examples of geometry being used to prove a statement about formal languages. The proof has two related disadvantages however. The first is that a key element of the proof, that of the complex exponential function, is not immediately understood without background in geometry. The second is that this also seems to restrict the proof technique to two dimensions, and there is no obvious avenue to generalize the result to a variant of MIX with four or five symbols. We hope to remedy this by an alternative, self-contained proof that avoids the complex exponential function. The core idea is a straightforward normalization of paths in two dimensions, which allow simple arguments to lead to a proof by contradiction. We also sketch part of a possible proof in three dimensions.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1117–1126,
Berlin, Germany, August 7-12, 2016.

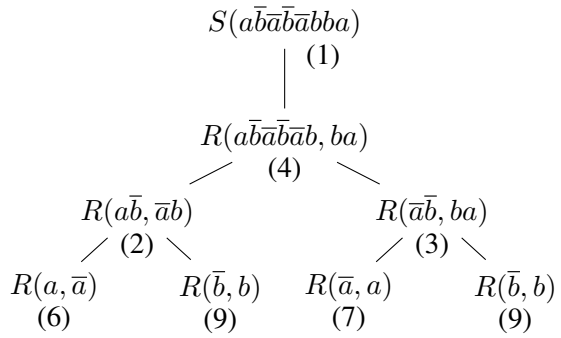


Figure 1: Derivation in G. The numbers indicate the rules that were used.

2 Initial problem

The MCFG G is defined as:

$$
\begin{aligned}
S(xy) &\leftarrow R(x,y) && (1)\\
R(xp,yq) &\leftarrow R(x,y)\ R(p,q) && (2)\\
R(xp,qy) &\leftarrow R(x,y)\ R(p,q) && (3)\\
R(xpy,q) &\leftarrow R(x,y)\ R(p,q) && (4)\\
R(p,xqy) &\leftarrow R(x,y)\ R(p,q) && (5)\\
R(a,\overline{a}) &\leftarrow && (6)\\
R(\overline{a},a) &\leftarrow && (7)\\
R(b,\overline{b}) &\leftarrow && (8)\\
R(\overline{b},b) &\leftarrow && (9)\\
R(\varepsilon,\varepsilon) &\leftarrow && (10)
\end{aligned}
$$

For the meaning of MCFGs in general, see Seki et al. (1991); for a closely related formalism, see Vijay-Shanker et al. (1987); see Kallmeyer (2010) for an overview of mildly context-sensitive grammar formalisms.

The reader unfamiliar with this literature is encouraged to interpret the rules of the grammar as logical implications, with S and R representing predicates. There is an implicit conjunction between the two occurrences of R in the right-hand side of each of the rules (2) — (5). The symbols x, y, p, q are string-valued variables, with implicit universal quantification that has scope over both left-hand side and right-hand side of a rule. The rules (6) — (10) act as axioms. The symbols a, $\overline{a}$, b, $\overline{b}$ are terminals, and ε denotes the empty string.

We can derive $S(x)$ for certain strings x, and $R(x,y)$ for certain strings x and y. Figure 1 presents an example of a derivation. The language generated by G is the set L of strings x such that $S(x)$ can be derived.

By induction on the depth of derivations, one can show that if $R(x,y)$, for strings x and y, then $xy \in \mathbf{O}_2$. Thereby, if $S(x)$ then $x \in \mathbf{O}_2$, which means $L \subseteq \mathbf{O}_2$. The task ahead is to prove that if $xy \in \mathbf{O}_2$, for some x and y, then $R(x,y)$. From this, $L = \mathbf{O}_2$ then follows.

Let $|x|$ denote the length of string x. For an inductive proof that $xy \in \mathbf{O}_2$ implies $R(x,y)$, the base cases are as follows. If $xy \in \mathbf{O}_2$ and $|x| \leq 1$ and $|y| \leq 1$, then trivially $R(x,y)$ by rules (6) — (10).

Furthermore, if we can prove that $xy \in \mathbf{O}_2$, $x \neq \varepsilon$ and $y \neq \varepsilon$ together imply $R(x,y)$, for $|xy| = m$, for some m, then we may also prove that $x'y' \in \mathbf{O}_2$ on its own implies $R(x',y')$ for $|x'y'| = m$. To see this, consider $m > 0$ and $z \in \mathbf{O}_2$ with $|z| = m$, and write it as $z = xy$ for some $x \neq \varepsilon$ and $y \neq \varepsilon$. If by assumption $R(x,y)$, then together with $R(\varepsilon,\varepsilon)$ and rule (4) or (5) we may derive $R(xy,\varepsilon)$ or $R(\varepsilon,xy)$, respectively. In the light of this, the inductive step merely needs to show that if for some x and y:

- $xy \in \mathbf{O}_2$, $|x| \geq 1$, $|y| \geq 1$ and $|xy| > 2$, and
- $pq \in \mathbf{O}_2$ and $|pq| < |xy|$ imply $R(p,q)$, for all p and q,

then also $R(x,y)$. One easy case is if $x \in \mathbf{O}_2$ (and thereby $y \in \mathbf{O}_2$) because then we can write $x = x_1x_2$ for some $x_1 \neq \varepsilon$ and $x_2 \neq \varepsilon$. The inductive hypothesis states that $R(x_1,x_2)$ and $R(\varepsilon,y)$, which imply $R(x,y)$ using rule (4).

A second easy case is if x or y has a proper prefix or proper suffix that is in $\mathbf{O}_2$. For example, assume there are $z_1 \neq \varepsilon$ and $z_2 \neq \varepsilon$ such that $x = z_1z_2$ and $z_1 \in \mathbf{O}_2$. Then we can use the inductive hypothesis on $R(z_1,\varepsilon)$ and $R(z_2,y)$, together with rule (2).

At this time, the reader may wish to read Figure 1 from the root downward. First, $a\overline{b}\overline{a}\overline{b}\overline{a}bba$ is divided into a pair of strings, namely $a\overline{b}\overline{a}\overline{b}\overline{a}b$ and ba. At each branching node in the derivation, a pair of strings is divided into four strings, which are grouped into two pairs of strings, using rules (2) — (5), read from left to right. Rules (2) and (3) divide each left-hand side argument into two parts. Rule (4) divides the first left-hand side argument into three parts, and rule (5) divides the second left-hand side argument into three parts.

What remains to show is that if $z_1z_2 \in \mathbf{O}_2$, $z_1 \notin \mathbf{O}_2$ and $|z_1z_2| > 2$, and no proper prefix or proper suffix of z_1 or of z_2 is in $\mathbf{O}_2$, then there is at least

one rule that allows us to divide z_1 and z_2 into four strings altogether, say x, y, p, q, of which at least three are non-empty, such that $xy \in \mathbf{O}_2$. This will then permit use of the inductive hypothesis on $R(x, y)$ and on $R(p, q)$.

We can in fact restrict our attention to $z_1'z_2' \in \mathbf{O}_2$, $|z_1'z_2'| > 2$, and no non-empty substring of z_1' or of z_2' is in $\mathbf{O}_2$, which can be justified as follows. Suppose we have z_1 and z_2 as in the previous paragraph, and suppose z_1' and z_2' result from z_1 and z_2 by exhaustively removing all non-empty substrings that are in $\mathbf{O}_2$; note that still $|z_1'z_2'| > 2$. If we can use a rule to divide z_1' and z_2' into x', y', p', q', of which at least three are non-empty, such that $x'y' \in \mathbf{O}_2$, then the same rule can be used to divide z_1 and z_2 into x, y, p, q with the required properties, which can be found from x', y', p', q' simply by reintroducing the removed substrings at corresponding positions.

3 Geometrical view

We may interpret a string x geometrically in two dimensions, as a path consisting of a series of line segments of length 1, starting in some point (i, j). Every symbol in x, from beginning to end, represents the next line segment in that path; an occurrence of a represents a line segment from the previous point (i, j) to the next point $(i + 1, j)$, $\overline{a}$ represents a line segment from (i, j) to $(i - 1, j)$, b represents a line segment from (i, j) to $(i, j+1)$, and $\overline{b}$ represents a line segment from (i, j) to $(i, j - 1)$. If $x \in \mathbf{O}_2$, then the path is *closed*, that is, the starting point and the ending point are the same. If we have two strings x and y such that $xy \in \mathbf{O}_2$ and $x \notin \mathbf{O}_2$, then this translates to two paths, connecting two distinct points, which together form a closed path. This is illustrated in Figure 2.

In the following, we assume a fixed choice of some x and y such that $xy \in \mathbf{O}_2$, $|xy| > 2$, and no non-empty substring of x or of y is in $\mathbf{O}_2$. If we follow the path of x starting in $P[0] = (0, 0)$, then the path ends in some point $P[1] = (i, j)$ such that i is the number of occurrences of a minus the number of occurrences of $\overline{a}$ and j is the number of occurrences of b minus the number of occurrences of $\overline{b}$. This path from $P[0]$ to $P[1]$ will be called $A[0]$. The path of y from $P[1]$ back to $P[0]$ will be called $B[1]$. We generalize this by defining for any integer k: $P[k]$ is the point $(k \cdot i, k \cdot j)$, $A[k]$ is the path of x from $P[k]$ to $P[k + 1]$ and $B[k]$ is the path of y from $P[k]$ to $P[k - 1]$. Where the starting points are irrelevant and only the shapes matter, we talk about paths A and B.

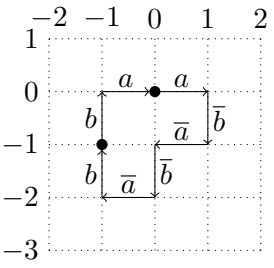


Figure 2: Two strings $x = a\overline{b}\overline{a}\overline{b}\overline{a}b$ and $y = ba$ together represent a closed path, consisting of a path from $(0, 0)$ to $(-1, -1)$ and a path from $(-1, -1)$ to $(0, 0)$.

Let C be a path, which can be either $A[k]$ or $B[k]$ for some k. We write $Q \in C$ to denote that Q is a point on C. Let $Q = (i, j) \in C$, not necessarily with i and j being integers. We define the path-distance $d_C(Q)$ of Q on C to be the length of the path along line segments of C to get from $P[k]$ to Q. In Figure 2, $(0, -1)$ has path-distance 3 on $A[0]$, as the path on $A[0]$ to reach $(0, -1)$ from $P[0] = (0, 0)$ consists of the line segments represented by the prefix $a\overline{b}\overline{a}$ of x. Similarly, $d_{A[0]}((0.5, -1)) = 2.5$.

Let C be a path as above and let points $Q_1, Q_2 \in C$ be such that $d_C(Q_1) \leq d_C(Q_2)$. We define the subpath $D = sub_C(Q_1, Q_2)$ to be such that $Q \in D$ if and only if $Q \in C$ and $d_C(Q_1) \leq Q \leq d_C(Q_2)$, and $d_D(Q) = d_C(Q) - d_C(Q_1)$ for every $Q \in D$. For two points Q_1 and Q_2, the line segment between Q_1 and Q_2 is denoted by $seg(Q_1, Q_2)$.

The task formulated at the end of Section 2 is accomplished if we can show that at least one of the following must hold:

- the angle in $P[0]$ between the beginning of $A[0]$ and that of $B[0]$ is 180° (Figure 3);
- there is a point $Q \notin \{P[0], P[1]\}$ such that $Q \in A[0]$ and $Q \in B[1]$ (Figure 4);
- there is a point $Q \neq P[1]$ such that $Q \in A[0]$, $Q \in A[1]$ and $d_{A[0]}(Q) > d_{A[1]}(Q)$ (Figure 5); or
- there is a point $Q \neq P[0]$ such that $Q \in B[0]$, $Q \in B[1]$ and $d_{B[1]}(Q) > d_{B[0]}(Q)$ (analogous to Figure 5).

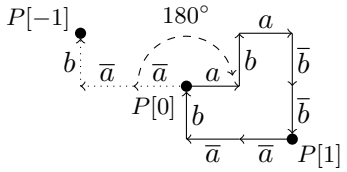


Figure 3: With $x = a\,b\,a\,\overline{b}\,\overline{b}$ and $y = \overline{a}\,\overline{a}\,b$, the beginning of path $A[0]$ and the beginning of (dotted) path $B[0]$ have an $180°$ angle in $P[0]$, which implies x and y start with complementing symbols (here a and $\overline{a}$; the other possibility is b and $\overline{b}$). By applying rule (2), two smaller closed paths result, one of which consists of these two complementing symbols.

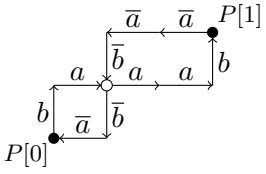


Figure 4: The paths $A[0]$ and $B[1]$ of $x = baaab$ and $y = \overline{a}\,\overline{a}\overline{b}\,\overline{b}\overline{a}$ have point $(1, 1)$ in common. Two smaller closed paths result by applying rule (3).

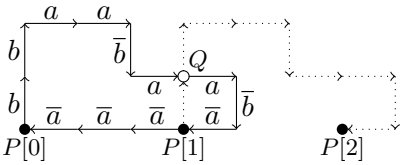


Figure 5: With $x = b\,b\,a\,a\,\overline{b}\,a\,a\,\overline{b}\,\overline{a}$ and $y = \overline{a}\,\overline{a}\,\overline{a}$, the path $A[0]$ and the (dotted) path $A[1]$ have point Q in common, with $d_{A[0]}(Q) = 6 > d_{A[1]}(Q) = 1$. By applying rule (4), two smaller closed paths result, one of which is formed by prefix b of length 1 and suffix $a\overline{b}\,\overline{a}$ of length $|x| - 6 = 3$ of x.

We will do this through a contradiction that results if we assume:

(i) the angle in $P[0]$ between the beginning of $A[0]$ and that of $B[0]$ is *not* $180°$;

(ii) $A[0] \cap B[1] = \{P[0], P[1]\}$;

(iii) there is no $Q \in (A[0] \cap A[1]) \setminus \{P[1]\}$ such that $d_{A[0]}(Q) > d_{A[1]}(Q)$; and

(iv) there is no $Q \in (B[0] \cap B[1]) \setminus \{P[0]\}$ such that $d_{B[1]}(Q) > d_{B[0]}(Q)$.

In the below, we will refer to these assumptions as the *four constraints*.

4 Continuous view

Whereas paths A and B were initially formed out of line segments of length 1 between points (i, j) with integers i and j, the proof becomes considerably easier if we allow i and j to be real numbers. The benefit lies in being able to make changes to the paths that preserve the four constraints, to obtain a convenient normal form for A and B. If we can prove a contradiction on the normal form, we will have shown that no A and B can exist that satisfy the four constraints.

We define, for each integer k, the line $\ell[k]$, which is perpendicular to the line through $P[k]$ and $P[k + 1]$, and lies exactly half-way between $P[k]$ and $P[k + 1]$. Much as before, we write $Q \in \ell[k]$ to denote that Q is a point on line $\ell[k]$. We will consistently draw points $\ldots, P[-1], P[0], P[1], \ldots$ in a straight line from left to right.

Let C be a path, which can be either $A[k']$ or $B[k']$, for some k', and let $Q \in C$. We write $\mathit{from\_right}_C(Q, \ell[k])$ to mean that path C is strictly to the right of $\ell[k]$ just before reaching Q, or formally, there is some $\delta > 0$ such that each $Q' \in C$ with $d_C(Q) - \delta < d_C(Q') < d_C(Q)$ lies strictly to the right of $\ell[k]$. The predicates $\mathit{from\_left}$, $\mathit{to\_right}$, $\mathit{to\_left}$ are similarly defined.

Let $Q_1, Q_2 \in C \cap \ell[k]$, for some k, such that $d_C(Q_1) \leq d_C(Q_2)$. We say that C has an *excursion from the right* between Q_1 and Q_2 at $\ell[k]$ if $\mathit{from\_right}_C(Q_1, \ell[k])$ and $\mathit{to\_right}_C(Q_2, \ell[k])$. This is illustrated in Figure 6: the path is strictly to the right of $\ell[k]$ just before reaching Q_1. From there on it may (but need not) cross over to the left of $\ell[k]$. Just after it reaches Q_2, it must again be strictly to the right of $\ell[k]$. The definition of *excursion from the left* is symmetric. Note that excursions may be nested; in Figure 6, $\mathit{sub}_C(Q_1, Q_2)$ has an excursion at $\ell[k]$ from the left below Q_2.

In Figure 6, the pair of points Q_1 and R_1 will be called a *crossing* of $\ell[k]$ from right to left, characterized by $Q_1, R_1 \in \ell[k]$, $\mathit{from\_right}_C(Q_1, \ell[k])$, $\mathit{to\_left}_C(R_1, \ell[k])$ and $\mathit{sub}_C(Q_1, R_1)$ being a line segment. The pair of points R_2 and Q_2 is a crossing of $\ell[k]$ from left to right, where the length of $\mathit{seg}(R_2, Q_2)$ happens to be 0. In much of the following we will simplify the discussion by assuming crossings consist of single points, as in the case of $R_2 = Q_2$. However, existence of crossings con-

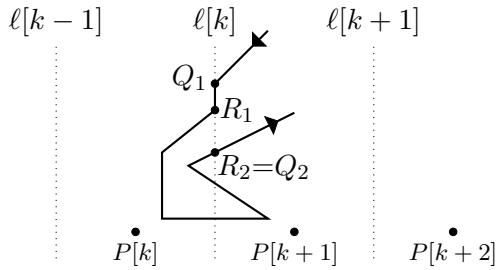


Figure 6: Excursion from the right at $\ell[k]$.

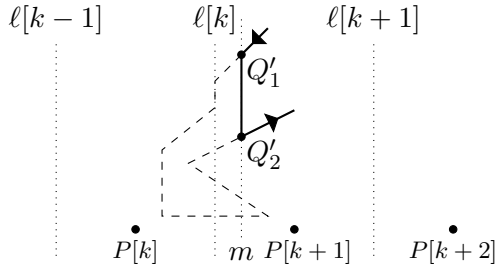


Figure 7: The excursion from Figure 6 truncated in Q'_1 and Q'_2 on line m.

sisting of line segments of non-zero length, as in the case of Q_1 and R_1, would not invalidate any of the arguments of the proof.

Excursions are the core obstacle that needs to be overcome for our proof. We can *truncate* an excursion at $\ell[k]$ by finding a suitable line m that is parallel to $\ell[k]$, some small distance away from it, between $\ell[k]$ and $P[k+1]$ for excursions from the right, and between $\ell[k]$ and $P[k]$ for excursions from the left. We further need to find points $Q'_1, Q'_2 \in C \cap m$, where $d_C(Q'_1) < d_C(Q_1)$ and $d_C(Q_2) < d_C(Q'_2)$. Because our coordinates no longer need to consist of integers, it is clear that m, Q'_1 and Q'_2 satisfying these requirements must exist.

The truncation consists in changing $sub_C(Q'_1, Q'_2)$ to become $seg(Q'_1, Q'_2)$, as illustrated by Figure 7. Note that if C is say $A[k']$, for some k', then changing the shape of C means changing the shape of $A[k'']$ for any other k'' as well; the difference between $A[k']$ and $A[k'']$ is only in the starting point $P[k']$ versus $P[k'']$.

At this time, we must allow for the possibility that for some excursions, no m, Q'_1 and Q'_2 can be found with which we can implement a truncation, if we also need to preserve the four constraints and preserve absence of self-intersections. There is a small number of possible causes. First,

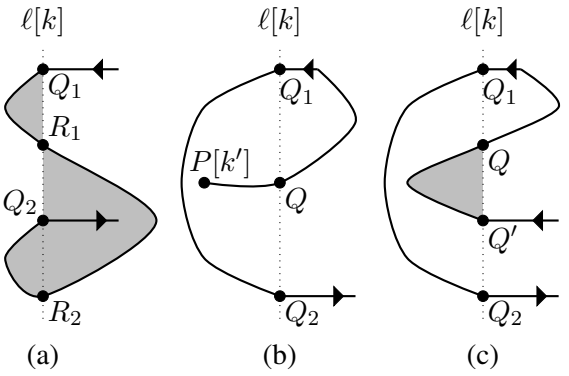


Figure 8: (a) Regions (shaded) of an excursion at $\ell[k]$; due to additional crossings in R_1 and R_2, three more excursions exist, each with a smaller area. (b) & (c) If truncation would introduce self-intersection, then either the excursion is filled, with some point $P[k']$ as in (b), or there is an excursion with smaller area, illustrated by shading in (c).

suppose that $C = A[k']$ and $B[k'+1]$ intersects with $seg(Q_1, Q_2)$. Then $B[k'+1]$ may intersect with $seg(Q'_1, Q'_2)$ for any choice of m, Q'_1 and Q'_2, and thereby no truncation is possible without violating constraint (ii). Similarly, a truncation may be blocked if $C = B[k'+1]$ and $A[k']$ intersects with $seg(Q_1, Q_2)$. Next, it could be that $C = A[k']$, while $d_{A[k']}(Q_1) > d_{A[k'+1]}(Q)$ holds for some $Q \in seg(Q_1, Q_2) \cap A[k'+1]$, or $d_{A[k'-1]}(Q) > d_{A[k']}(Q_2)$ holds for some $Q \in seg(Q_1, Q_2) \cap A[k'-1]$, either of which potentially blocks a truncation if constraint (iii) is to be preserved. Constraint (iv) has similar consequences. Furthermore, if we need to preserve absence of self-intersections, a truncation may be blocked if $d_C(Q) < d_C(Q_1)$ or $d_C(Q_2) < d_C(Q)$ for some $Q \in seg(Q_1, Q_2) \cap C$.

5 Normal form

The *regions* of an excursion of C between Q_1 and Q_2 at $\ell[k]$ are those that are enclosed by (subpaths of) $sub_C(Q_1, Q_2)$ and (subsegments of) $seg(Q_1, Q_2)$, as illustrated by Figure 8(a). The *area* of the excursion is the surface area of all regions together. We say an excursion is *filled* if any of its regions contains at least one point $P[k']$, for some integer k', otherwise it is said to be *unfilled*.

We say A and B are in *normal form* if no excursion can be truncated without violating the four constraints or introducing a self-intersection. Sup-

pose A and B are in normal form, while one or more excursions remain. Let us first consider the unfilled excursions. Among them choose one that has the smallest area. By assumption, one of the four constraints must be violated or a new self-intersection must be introduced, if we were to truncate that excursion. We will consider all relevant cases.

Each case will assume an unfilled excursion from the right (excursions from the left are symmetric) of a path C between Q_1 and Q_2 at $\ell[k]$. We may assume that $sub_C(Q_1, Q_2) \cap \ell[k] = \{Q_1, Q_2\}$, as additional crossings of $\ell[k]$ would mean that excursions exist with smaller areas (cf. Figure 8(a)), contrary to the assumptions. Now assume truncation is blocked due to $Q \in seg(Q_1, Q_2) \cap C$ such that $d_C(Q) < d_C(Q_1)$ (the case $d_C(Q_2) < d_C(Q)$ is symmetric), as we need to preserve absence of self-intersection. Suppose Q is the only such point, so that C crosses $seg(Q_1, Q_2)$ from left to right once without ever crossing it from right to left, until Q_1 is reached. Then C starts in the area of the excursion, or in other words, the excursion is filled, contrary to the assumptions (cf. Figure 8(b)). Now suppose there are points Q' and Q where C crosses $seg(Q_1, Q_2)$ from right to left and from left to right, respectively and $d_C(Q') < d_C(Q) < d_C(Q_1)$. If there are several choices, choose Q' and Q such that $sub_C(Q', Q) \cap \ell[k] = \{Q', Q\}$. This means the excursion between Q' and Q has an area smaller than the one between Q_1 and Q_2, contrary to the assumptions (cf. Figure 8(c)).

Note that excursions with zero area, that is, those that intersect with $\ell[k]$ without crossing over to the other side, can always be truncated. We can therefore further ignore non-crossing intersections.

Now suppose a truncation would violate constraint (ii), where $C = B[k'+1]$ and $D = A[k']$ crosses $seg(Q_1, Q_2)$. Then much as above, we may distinguish two cases. In the first, D has only one crossing of $seg(Q_1, Q_2)$ in some point Q, which means the excursion is filled with the starting or ending point of D, as in Figure 9(a). In the second, D has at least two consecutive crossings, say in Q and Q', from right to left and from left to right, respectively, which means the excursion between Q and Q' has smaller area than the one between Q_1 and Q_2, illustrated by shading in Figure 9(b). Both cases contradict the assumptions.

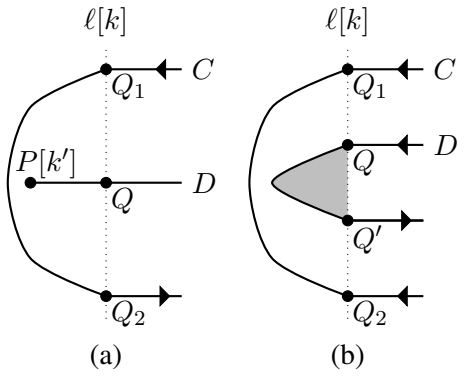


Figure 9: Truncating the excursion would introduce a violation of constraint (ii). The assumptions are contradicted in one of two ways.

For $C = A[k']$ and $D = B[k'+1]$, the reasoning is symmetric.

Next, suppose a truncation would violate constraint (iii), where $C = A[k']$ and $A[k'-1]$ crosses $seg(Q_1, Q_2)$ in Q, while $d_{A[k'-1]}(Q) > d_{A[k']}(Q_2)$. If the crossing in Q is from right to left, and there is an immediately next crossing in Q' from left to right, then we have the same situation as in Figure 9(b), involving an excursion with smaller area, contradicting the assumptions. If the crossing in Q is the only one, and it is from right to left, then we can use the fact that $sub_{A[k']}(Q_1, Q_2) \cap sub_{A[k'-1]}(Q, P[k']) = \emptyset$, as we assume the four constraints as yet hold. This means $P[k']$ must be contained in the area of the excursion, as illustrated in Figure 10(a), contradicting the assumption that the excursion is unfilled. If the crossing in Q is the only one, and it is from left to right, then we can use the fact that $sub_{A[k']}(Q_1, Q_2) \cap sub_{A[k'-1]}(Q'_2, Q) = \emptyset$, for the unique $Q'_2 \in A[k'-1] \cap \ell[k-1]$ such that $d_{A[k'-1]}(Q'_2) = d_{A[k']}(Q_2)$. This means the excursion contains Q'_2, which implies there is another unfilled excursion between points $R_1, R_2 \in A[k'] \cap \ell[k-1]$ with smaller area, as shaded in Figure 10(b), contrary to the assumptions.

Suppose a truncation would violate constraint (iii), where $C = A[k']$ and $A[k'+1]$ crosses $seg(Q_1, Q_2)$ in Q, while $d_{A[k']}(Q_1) > d_{A[k'+1]}(Q)$. The reasoning is now largely symmetric to the above, with the direction of the crossing reversed, except that the case analogous to Figure 10(b) is immediately excluded, as Q'_2 cannot be both to the left and to the right of $\ell[k]$. Constraint (iv) is symmetric to constraint (iii). All pos-

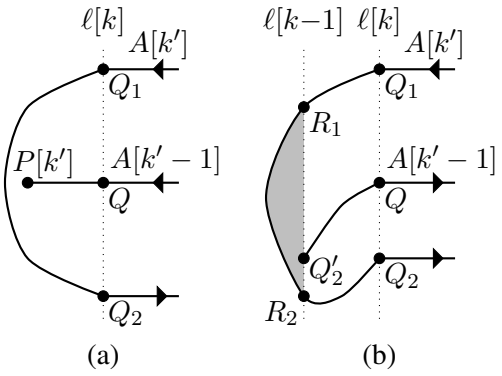


Figure 10: Truncating the excursion would introduce a violation of constraint (iii), where $d_{A[k'-1]}(Q) > d_{A[k']}(Q_2)$. The assumptions are contradicted in one of three ways, the first as in Figure 9(b), and the second and third as in (a) and (b) above.

sible cases have been shown to lead to contradictions, and therefore we conclude that there are no unfilled excursions if A and B are in normal form.

We now show that there cannot be any filled excursions either. For this, assume that $A[k']$ has a filled excursion between Q_1 and Q_2 at $\ell[k]$ from the right. This means $A[k'-1]$ has an identically shaped, filled excursion at $\ell[k-1]$ from the right, between corresponding points Q'_1 and Q'_2. Let us consider how path $A[k']$ proceeds after reaching Q_2. There are only three possibilities:

- it ends in $P[k+1]$, with $k' = k$, without further crossings of $\ell[k]$ or $\ell[k+1]$;
- it next crosses $\ell[k]$ leftward; or
- it next crosses $\ell[k+1]$ in some point Q_3.

The first of these can be excluded, in the light of $d_{A[k'-1]}(Q) \geq d_{A[k']}(Q_2)$ for each $Q \in sub_{A[k'-1]}(Q'_2, P[k'])$. Due to constraint (iii) therefore, this subpath of $A[k'-1]$ cannot intersect with the excursion of $A[k']$ to reach $P[k]$, and therefore $A[k']$ cannot reach $P[k+1]$. The second possibility is also excluded, as this would imply the existence of an unfilled excursion. For the remaining possibility, $Q_3 \in A[k'] \cap \ell[k+1]$ may be lower down than Q_2 (in the now familiar view of the points $P[0], P[1], \ldots$ being drawn from left to right along a horizontal line), or it may be higher up than Q_1. These two cases are drawn in Figures 11 and 12. The choice of Q_3 also determines a corresponding $Q'_3 \in A[k'-1] \cap \ell[k]$.

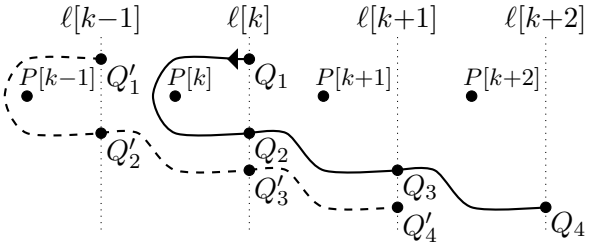


Figure 11: Continuing the (solid) path $A[k']$ after a filled excursion, restricted by the (dashed) path $A[k'-1]$, in the light of constraint (iii).

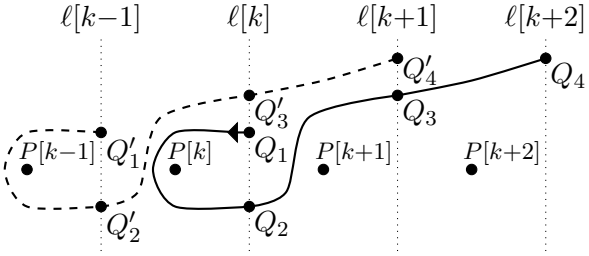


Figure 12: As in Figure 11 but Q_3 is chosen to be higher up than Q_1.

We now consider how $A[k']$ continues after Q_3 in the case of Figure 11. If it next crosses $\ell[k+1]$ leftward, this would imply the existence of an unfilled excursion. Further, $d_{A[k'-1]}(Q) \geq d_{A[k']}(Q_3)$ for each $Q \in sub_{A[k'-1]}(Q'_3, P[k'])$. Due to constraint (iii) therefore, this subpath of $A[k'-1]$ cannot intersect with $sub_{A[k']}(Q_2, Q_3)$, above which lies $P[k+1]$. Therefore, $A[k']$ must cross $\ell[k+2]$ in some Q_4, which is lower down than Q_3. This continues ad infinitum, and $A[k']$ will never reach its supposed end point $P[k'+1]$. The reasoning for Figure 12 is similar.

Filled excursions from the left are symmetric, but instead of investigating the path after Q_2, we must investigate the path *before* Q_1. The case of B is symmetric to that of A. We may now conclude no filled excursions exist.

6 The final contradiction

We have established that after A and B have been brought into normal form, there can be no remaining excursions. This means that $A[0]$ crosses $\ell[0]$ exactly once, in some point R_A, and $B[0]$ crosses $\ell[-1]$ exactly once, in some point L_B. Further, let L_A be the unique point where $A[-1]$ crosses $\ell[-1]$ and R_B the unique point where $B[1]$ crosses $\ell[0]$.

The region of the plane between $\ell[-1]$ and $\ell[0]$

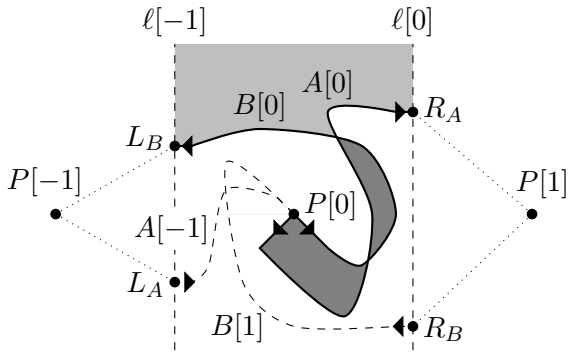


Figure 13: The region between $\ell[-1]$ and $\ell[0]$ is divided by $A[0]$ and $B[0]$ into a 'top' region (lightly shaded), a 'bottom' region (white), and areas enclosed by intersections of $A[0]$ and $B[0]$ (darkly shaded). Here $A[-1]$ and $B[1]$ are both in the 'bottom' region.

can now be partitioned into a 'top' region, a 'bottom' region, and zero or more enclosed regions. The 'top' region consists of those points that are reachable from any point between $\ell[-1]$ and $\ell[0]$ arbitrarily far above any point of $A[0]$ and $B[0]$, without intersecting with $A[0]$, $B[0]$, $\ell[-1]$ or $\ell[0]$. This is the lightly shaded region in Figure 13. The 'bottom' region is similarly defined, in terms of reachability from any point between $\ell[-1]$ and $\ell[0]$ arbitrarily far *below* any point of $A[0]$ and $B[0]$. The zero or more enclosed regions stem from possible intersections of $A[0]$ and $B[0]$; the two such enclosed regions in Figure 13 are darkly shaded. Note that the four constraints do *not* preclude intersections of $A[0]$ and $B[0]$.

However, constraint (ii) implies that, between $\ell[-1]$ and $\ell[0]$, $A[0]$ and $B[1]$ do not intersect other than in $P[0]$, and similarly, $A[-1]$ and $B[0]$ do not intersect other than in $P[0]$. Moreover, for any $Q \in sub_{A[-1]}(L_A, P[0])$ and any $Q' \in sub_{A[0]}(P[0], R_A)$ we have $d_{A[-1]}(Q) \geq d_{A[0]}(Q')$. By constraint (iii) this means $A[-1]$ and $A[0]$ do not intersect other than in $P[0]$. Similarly, $B[1]$ and $B[0]$ do not intersect other than in $P[0]$.

The angles in $P[0]$ between $A[0]$, $B[0]$, $A[-1]$ and $B[1]$ are multiples of $90°$. Because of constraint (i), which excludes a $180°$ angle between $A[0]$ and $B[0]$, it follows that either $sub_{A[-1]}(L_A, P[0])$ and $sub_{B[1]}(R_B, P[0])$ both lie entirely in the 'top' region, or both lie entirely in the 'bottom' region. The latter case is illustrated in Figure 13. In the former case, L_A and R_B are above L_B and R_A, respectively, and in the latter case L_A and R_B are below L_B and R_A. This is impossible, as L_A and R_A should be at the same height, these being corresponding points of $A[-1]$ and $A[0]$, which have the same shape, and similarly L_B and R_B should be at the same height.

This contradiction now leads back to the very beginning of our proof, and implies that the four constraints cannot all be true, and therefore that at least one rule is always applicable to allow use of the inductive hypothesis, and therefore that G generates $\mathbf{O}_2$.

7 Conclusions and outlook

We have presented a new proof that $\mathbf{O}_2$ is generated by a MCFG. It has at least superficial elements in common with the proof by Salvati (2015). Both proofs use essentially the same MCFG, both are geometric in nature, and both involve a continuous view of paths next to a discrete view. The major difference lies in the approach to tackling the myriad ways in which the paths can wind around each other and themselves. In the case of Salvati (2015), the key concept is that of the complex exponential function, which seems to restrict the proof technique to two-dimensional geometry. In our case, the key concepts are excursions and truncation thereof, and the identification of top and bottom regions.

At this time, no proof is within reach that generalizes the result to $\mathbf{O}_3$, i.e. the language of strings over an alphabet of six symbols, in which the number of a's equals the number of $\overline{a}$'s, the number of b's equals the number of $\overline{b}$'s, and the number of c's equals the number of $\overline{c}$'s; this language is rationally equivalent to MIX-4, which is defined analogously to MIX, but with four symbols. One may expect however that a proof would use three-dimensional geometry and generalize some of the arguments from this paper. Our aim here is to make this plausible, while emphasizing that an actual proof will require a novel framework at least as involved as that presented in the previous sections.

Omitting the start rule and the axioms, an obvious candidate MCFG to generate $\mathbf{O}_3$ would among others have the three rules:

$R(p_1q_1, p_2q_2, q_3p_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1q_1, q_2p_2, p_3q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(q_1p_1, p_2q_2, p_3q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$

as well as the six rules:

$R(p_1q_1p_2, p_3q_2, q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1q_1p_2, q_2, p_3q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1q_1, p_2q_2p_3, q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1q_1, q_2, p_2q_3p_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1, q_1p_2q_2, q_3p_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1, q_1p_2, q_2p_3q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$

Consider three strings x, y and z such that $xyz \in \mathbf{O}_3$. If we can use any of the above rules to divide these into six strings out of which we can select three, which concatenated together are a non-empty string in $\mathbf{O}_3$ shorter than xyz, then we can use the inductive hypothesis, much as in Section 2. For a proof by contradiction, therefore assume that no pair of prefixes of x and y and a suffix of z together form a non-empty string in $\mathbf{O}_3$ shorter than xyz, etc., in the light of the first three rules above, and assume that no 'short enough' prefix of x, a prefix of y and a 'short enough' suffix of x together form a non-empty string in $\mathbf{O}_3$, etc., in the light of the next six rules above.

For a geometric interpretation, consider the paths of x, y and z, leading from point $P_0 = (0, 0, 0)$ to points P_x, P_y and P_z, respectively. The concatenations of prefixes of x and y, and similarly those of x and z and those of y and z form three connecting surfaces, together forming one surface dividing the space around P_0 into an 'above' and a 'below'; cf. Figure 14. Our assumptions imply that the final parts of the paths of x, y and z from $-P_x$, $-P_y$ and $-P_z$, respectively, to P_0 should not intersect with this surface. In addition, no pair of strings from x, y and z should end on complementing symbols, i.e. a and $\overline{a}$, b and $\overline{b}$, or c and $\overline{c}$. This means that the three paths leading towards P_0 must all end in P_0 strictly 'above' or all strictly 'below' the surface.

This might lead to a contradiction, similar to that in Section 6, but only if one can ensure that none of the three paths to P_0 'sneak around' the surface. This is illustrated in Figure 15, where the path of z is 'entangled' with a copy of itself. It appears this can be achieved by adding three more rules, namely:

$R(p_1q_1p_2q_2, p_3, q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1, q_1p_2q_2p_3, q_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$
$R(p_1, q_1, q_2p_2q_3p_3) \leftarrow R(p_1, p_2, p_3)\ R(q_1, q_2, q_3)$

The physical interpretation of, say, the last rule seems to be that the path of z from $-P_z$ to P_0 can be iteratively shifted such that points other than its ending point coincide with P_0. At some stage

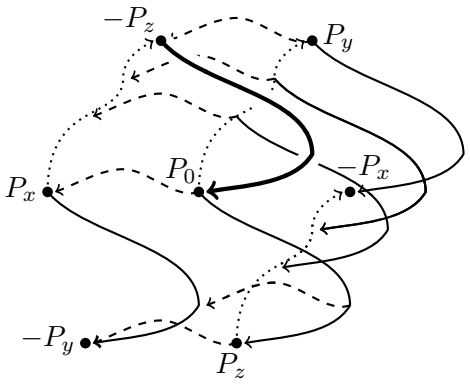


Figure 14: By taking prefixes of two strings from $\{x, y, z\}$ and concatenating them, we obtain a surface dividing the space around P_0 into 'above' and 'below'. Here the path of z from $-P_z$ to P_0 ends 'above', if our view is from above the surface.

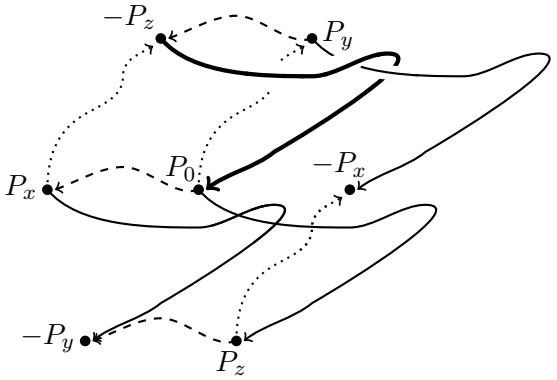


Figure 15: The path of z from $-P_z$ to P_0 is initially above the surface, but 'sneaks around' the path of z from P_y to $-P_x$, to end below.

the shifted path must intersect with the path of z from P_y to $-P_x$, before the entanglement of the two paths is broken.

The considerable challenges ahead involve finding a suitable definition of 'excursions' in three dimensions, and proving that these can be systematically truncated without violating appropriate constraints that preclude application of the above 12 rules.

Acknowledgements

This work came out of correspondence with Giorgio Satta. Gratefully acknowledged are also fruitful discussions with Sylvain Salvati, Vinodh Rajan, and Markus Pfeiffer. Much appreciated are anonymous referees and editors for their efforts and their courage to consider a theoretical paper for publication at this venue.

References

[Bach1981] E. Bach. 1981. Discontinuous constituents in generalized categorial grammars. In *Proceedings of the Eleventh Annual Meeting of the North Eastern Linguistic Society*, pages 1–12.

[Boullier1999] P. Boullier. 1999. Chinese numbers, MIX, scrambling, and Range Concatenation Grammars. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, pages 53–60, Bergen, Norway, June.

[Capelletti and Tamburini2009] M. Capelletti and F. Tamburini. 2009. Parsing with polymorphic categorial grammars. *Research in Computing Science*, 41(2009):87–98.

[Gilman2005] R.H. Gilman. 2005. Formal languages and their application to combinatorial group theory. *Contemporary Mathematics*, 378:1–36.

[Joshi et al.1991] A.K. Joshi, K. Vijay-Shanker, and D. Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In P. Sells, S.M. Shieber, and T. Wasow, editors, *Foundational Issues in Natural Language Processing*, chapter 2, pages 31–81. MIT Press.

[Joshi1985] A.K. Joshi. 1985. Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D.R. Dowty, L. Karttunen, and A.M. Zwicky, editors, *Natural language parsing: Psychological, computational, and theoretical perspectives*, pages 206–250. Cambridge University Press.

[Kallmeyer2010] Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*. Springer-Verlag.

[Kanazawa and Salvati2012] M. Kanazawa and S. Salvati. 2012. MIX is not a tree-adjoining language. In *50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 666–674, Jeju Island, Korea, July.

[Pullum1983] G.K. Pullum. 1983. Context-freeness and the computer processing of human languages. In *21st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 1–6, Cambridge, Massachusetts, July.

[Salvati2015] S. Salvati. 2015. MIX is a 2-MCFL and the word problem in $\mathbb{Z}^2$ is captured by the IO and the OI hierarchies. *Journal of Computer and System Sciences*, 81:1252–1277.

[Seki et al.1991] H. Seki, T. Matsumura, M. Fujii, and T. Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88:191–229.

[Sorokin2014] A. Sorokin. 2014. Pumping lemma and Ogden lemma for displacement context-free grammars. In *Developments in Language Theory, 18th International Conference*, volume 8633 of *Lecture Notes in Computer Science*, pages 154–165, Ekaterinburg, Russia. Springer-Verlag.

[Vijay-Shanker et al.1987] K. Vijay-Shanker, D.J. Weir, and A.K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pages 104–111, Stanford, California, USA, July.

Context-aware Argumentative Relation Mining

Huy V. Nguyen
Computer Science Department
University of Pittsburgh
Pittsburgh, PA 15260
hvn3@pitt.edu

Diane J. Litman
Computer Science Department and
Learning Research and Development Center
University of Pittsburgh
Pittsburgh, PA 15260
dlitman@pitt.edu

Abstract

Context is crucial for identifying argumentative relations in text, but many argument mining methods make little use of contextual features. This paper presents context-aware argumentative relation mining that uses features extracted from writing topics as well as from windows of context sentences. Experiments on student essays demonstrate that the proposed features improve predictive performance in two argumentative relation classification tasks.

1 Introduction

By supporting tasks such as automatically identifying *argument components*[1] (e.g., premises, claims) in text, and the *argumentative relations* (e.g., support, attack) between components, argument (argumentation) mining has been studied for applications in different research fields such as document summarization (Teufel and Moens, 2002), opinion mining (Boltužić and Šnajder, 2014), automated essay evaluation (Burstein et al., 2003), legal information systems (Palau and Moens, 2009), and policy modeling platforms (Florou et al., 2013).

Given a pair of argument components with one component as the *source* and the other as the *target*, *argumentative relation mining* involves determining whether a relation holds from the source to the target, and classifying the argumentative function of the relation (e.g., support vs. attack). Argumentative relation mining - beyond argument component mining - is perceived as an essential step towards more fully identifying the argumentative structure of a text (Peldszus and Stede, 2013; Sergeant, 2013; Stab et al., 2014). Consider the second paragraph shown in Figure 1. Only detecting the argument components (a claim in sentence 2 and two premises in sentences 3 and 4) does not give a complete picture of the argumentation. By looking for relations between these components, one can also see that the two premises together justify the claim. The argumentation structure of the text in Figure 1 is illustrated in Figure 2.

> **Essay 73**. Topic: Is image more powerful than the written word?
> ...(1)Hence, *I agree only to certain degree that in today's world, image serves as a more effective means of communication*$_{[MajorClaim]}$.
> (2)Firstly, **pictures can influence the way people think**$_{[Claim]}$. (3)For example, nowadays horrendous images are displayed on the cigarette boxes to illustrate the consequences of smoking$_{[Premise]}$. (4)As a result, statistics show a slight reduction in the number of smokers, indicating that they realize the effects of the negative habit$_{[Premise]}$...

Figure 1: Excerpt from a student persuasive essay (Stab and Gurevych, 2014a). Sentences are numbered and argument components are tagged.

Our current study proposes a novel approach for argumentative relation mining that makes use of contextual features extracted from surrounding sentences of *source* and *target* components as well as from topic information of the writings.

[1]There is no consensus yet on an annotation scheme for argument components, or on the minimal textual units to be annotated. We follow Peldszus and Stede (2013) and consider "*argument mining as the automatic discovery of an argumentative text portion, and the identification of the relevant components of the argument presented there.*" We also borrow their term "*argumentative discourse unit*" to refer to the textual units (e.g., text segment, sentence, clause) which are considered as argument components.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1127–1137,
Berlin, Germany, August 7-12, 2016.

Prior argumentative relation mining studies have often used features extracted from argument components to model different aspects of the relations between the components, e.g., relative distance, word pairs, semantic similarity, textual entailment (Cabrio and Villata, 2012; Stab and Gurevych, 2014b; Boltužić and Šnajder, 2014; Peldszus and Stede, 2015b). Features extracted from the text surrounding the components have been less explored, e.g., using words and their part-of-speech from adjacent sentences (Peldszus, 2014). The first hypothesis investigated in this paper is that the *discourse relations* of argument components with adjacent sentences (called *context windows* in this study, a formal definition is given in §5.3) can help characterize the argumentative relations that connect pairs of argument components. Reconsidering the example in Figure 1, without knowing the content "*horrendous images are displayed on the cigarette boxes*" in sentence 3, one cannot easily tell that "*reduction in the number of smokers*" in sentence 4 supports the "*pictures can influence*" claim in sentence 2. We expect that such content relatedness can be revealed from a discourse analysis, e.g., the appearance of a discourse connective "*As a result*".

While topic information in many writing genres (e.g., scientific publications, Wikipedia articles, student essays) has been used to create features for argument component mining (Teufel and Moens, 2002; Levy et al., 2014; Nguyen and Litman, 2015), topic-based features have been less explored for argumentative relation mining. The second hypothesis investigated in this paper is that features based on *topic context* also provide useful information for improving argumentative relation mining. In the excerpt below, knowing that '*online game*' and '*computer*' are topically related might help a model decide that the claim in sentence 1 supports the claim in sentence 2:

(1)**People who are addicted to games, especially online games, can eventually bear dangerous consequences**$_{[Claim]}$.

(2)Although it is undeniable that computer is a crucial part of human life$_{[Premise]}$, **it still has its bad side**$_{[MajorClaim]}$.[2]

Motivated by the discussion above, we propose *context-aware argumentative relation mining* – a novel approach that makes use of contextual features that are extracted by exploiting context sentence windows and writing topic to improve relation prediction. In particular, we derive features using *discourse relations* between argument components and windows of their surrounding sentences. We also derive features using an *argument and domain word lexicon* automatically created by post-processing an essay's topic model. Experimental results show that our proposed contextual features help significantly improve performance in two argumentative relation classification tasks.

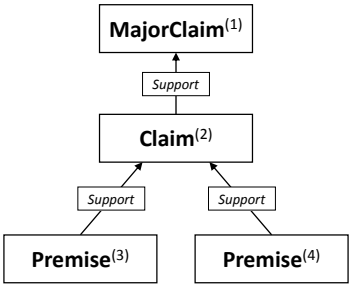


Figure 2: Structure of the argumentation in the excerpt. Relations are illustrated accordingly to the annotation provided in the corpus. Premises 3 and 4 were annotated for separate relations to Claim 2. Our visualization should not mislead that the two premises are linked or convergent.

2 Related Work

Unlike argument component identification where textual inputs are typically sentences or clauses (Moens et al., 2007; Stab and Gurevych, 2014b; Levy et al., 2014; Lippi and Torroni, 2015), textual inputs of argumentative relation mining vary from clauses (Stab and Gurevych, 2014b; Peldszus, 2014) to multiple-sentences (Biran and Rambow, 2011; Cabrio and Villata, 2012; Boltužić and Šnajder, 2014). Studying claim justification between user comments, Biran and Rambow (2011) proposed that the argumentation in justification of a claim can be characterized with discourse structure in the justification. They however only considered discourse markers but not discourse relations. Cabrio et al. (2013) conducted a corpus analysis and found certain similarity between Penn Discourse TreeBank relations (Prasad et al., 2008) and argumentation schemes (Walton et al., 2008). However they did not discuss how such similarity could be applied to argument mining.

Motivated by these findings, we propose to use features extracted from discourse relations be-

[2]In this excerpt, the Premise was annotated as an attack to the MajorClaim in sentence 2.

tween sentences for argumentative relation mining. Moreover, to enable discourse relation features when the textual inputs are only sentences/clauses, we group the inputs with their context sentences. Qazvinian and Radev (2010) used the term "context sentence" to refer to sentences surrounding a citation that contained information about the cited source but did not explicitly cite it. In our study, we only require that the context sentences of an argument component must be in the same paragraph and adjacent to the component.

Prior work in argumentative relation mining has used argument component labels to provide constraints during relation identification. For example, when an annotation scheme (e.g., (Peldszus and Stede, 2013; Stab and Gurevych, 2014a)) does not allow relations from claim to premise, no relations are inferred during relation mining for any argument component pair where the source is a claim and the target is a premise. In our work, we follow Stab and Gurevych (2014b) and use the predicted labels of argument components as features during argumentative relation mining. We, however, take advantage of an enhanced argument component model (Nguyen and Litman, 2016) to obtain more reliable argument component labels than in (Stab and Gurevych, 2014b).

Argument mining research has studied different data-driven approaches for separating organizational content (shell) from topical content to improve argument component identification, e.g., supervised sequence model (Madnani et al., 2012), unsupervised probabilistic topic models (Séaghdha and Teufel, 2014; Du et al., 2014). Nguyen and Litman (2015) post-processed LDA (Blei et al., 2003) output to extract a lexicon of argument and domain words from development data. Their semi-supervised approach exploits the topic context through essay titles to guide the extraction.

Finally, prior research has explored predicting different argumentative relationship labels between pairs of argument components, e.g., attachment (Peldszus and Stede, 2015a), support vs. non-support (Biran and Rambow, 2011; Cabrio and Villata, 2012; Stab and Gurevych, 2014b), {implicit, explicit}×{support, attack} (Boltužić and Šnajder, 2014), verifiability of support (Park and Cardie, 2014). Our experiments use two such argumentative relation classification tasks (Support vs. Non-support, Support vs. Attack) to evaluate the effectiveness of our proposed features.

3 Persuasive Essay Corpus

Stab and Gurevych (2014a) compiled the Persuasive Essay Corpus consisting of 90 student argumentative essays and made it publicly available.[3] Because the corpus has been utilized for different argument mining tasks (Stab and Gurevych, 2014b; Nguyen and Litman, 2015; Nguyen and Litman, 2016), we use this corpus to demonstrate our context-aware argumentative relation mining approach, and adapt the model developed by Stab and Gurevych (2014b) to serve as the baseline for evaluating our proposed approach.

Three experts identified possible argument components of three types within each sentence in the corpus (*MajorClaim* - writer's stance toward the writing topic, *Claim* - controversial statements that support or attack MajorClaim, and *Premise* - evidence used to underpin the validity of Claim), and also connected the argument components using two argumentative relations (*Support* and *Attack*). According to the annotation manual, each essay has exactly one MajorClaim. A sentence can have one or more argument components (*Argumentative* sentences). Sentences that do not contain any argument component are labeled *Non-argumentative*. Figure 1 shows an example essay with components annotated, and Figure 2 illustrates relations between those components. Argumentative relations are directed and can hold between a Premise and another Premise, a Premise and a (Major-) Claim, or a Claim and a MajorClaim. Except for the relation from Claim to MajorClaim, an argumentative relation does not cross paragraph boundaries. The three experts achieved inter-rater accuracy 0.88 for component labels and Krippendorff's α_U 0.72 for component boundaries. Given the annotated argument components, the three experts obtained Krippendorff's α 0.81 for relation labels. The number of relations are shown in Table 1.

4 Argumentative Relation Tasks

4.1 Task 1: Support vs. Non-support

Our first task follows (Stab and Gurevych, 2014b): given a pair of source and target argument components, identify whether the source argumentatively supports the target or not. Note that when a support relation does not hold, the source may attack or has no relation with the target compo-

[3] www.ukp.tu-darmstadt.de/data/argumentation-mining

Label	#instances
Within-paragraph constraint	
Support	989
Attack	103
No paragraph constraint	
Support	1312
Attack	161

Table 1: Data statistics of the corpus.

nent. For each of two argument components in the same paragraph[4], we form two pairs (i.e., reversing source and target). In total we obtain 6330 pairs, in which 989 (15.6%) have Support relation. Among 5341 Non-support pairs, 103 have Attack relation and 5238 are no-relation pairs.

Stab and Gurevych (2014b) split the corpus into an 80% training set and a 20% test set which have similar label distributions. We use this split to train and test our proposed models, and directly compare our models' performance to the reported performance in (Stab and Gurevych, 2014b).

4.2 Task 2: Support vs. Attack

To further evaluate the effectiveness of our approach, we conduct an additional task that classifies an argumentative relation as *Support* or *Attack*. For this task, we assume that the relation (i.e., attachment (Peldszus, 2014)) between two components is given, and aim at identifying the argumentative function of the relation. Because we remove the paragraph constraint in this task, we obtain more Support relations than in Task 1. As shown in Table 1, of the total 1473 relations, we have 1312 (89%) Support and 161 (11%) Attack relations. Because this task was not studied in (Stab and Gurevych, 2014b), we adapt Stab and Gurevych's model to use as the baseline.

5 Argumentative Relation Models

5.1 Baseline

We adapt (Stab and Gurevych, 2014b) to use as a baseline for evaluating our approach. Given a pair of argument components, we follow (Stab and Gurevych, 2014b) by first extracting 3 feature sets: structural (e.g., word counts, sentence position), lexical (e.g., word pairs, first words), and grammatical production rules (e.g., *S→NP,VP*).

[4]Allowing cross-paragraph relations exponentially increases the number of no-relation pairs, which makes the prediction data extremely skewed (Stab and Gurevych, 2014b).

Because a sentence may have more than one argument component, the relative component positions might provide useful information (Peldszus, 2014). Thus, we also include 8 new component position features: whether the source and target components are the whole sentences or the beginning/end components of the sentences; if the source is before or after the target component; and the absolute difference of their positions.

Stab and Gurevych (2014b) used a 55-discourse marker set to extract indicator features. We expand their discourse maker set by combining them with a 298-discourse marker set developed in (Biran and Rambow, 2011). We expect the expanded set of discourse markers will represent better possible discourse relations in the texts.

Stab and Gurevych (2014b) used predicted label of argument components as features for both training and testing their argumentation structure identification model.[5] As their predicted labels are not available to us, we adapt this feature set by using the argument component model in (Nguyen and Litman, 2016) which was shown to outperform the corresponding model of Stab and Gurevych.

For later presentation purposes, we name the set of all features from this section *except word pairs and production rules* as the **common features**. While word pairs and grammatical production rules were the most predictive features in (Stab and Gurevych, 2014b), we hypothesize that this large and sparse feature space may have negative impact on model robustness (Nguyen and Litman, 2015). Most of our proposed models replace word pairs and production rules with different combinations of new contextual features.

5.2 Topic-context Model

Our first proposed model (TOPIC) makes use of **Topic-context** features derived from a lexicon of argument and domain words for persuasive essays (Nguyen and Litman, 2015). Argument words (e.g., '*believe*', '*opinion*') signal the argumentative content and are commonly used across different topics. In contrast, domain words are specific terminologies commonly used within the topic (e.g., '*art*', '*education*'). The authors first use

[5]Stab and Gurevych (2014b) reported that including gold-standard labels of argument component in both training and testing phases yielded results close to human performance. Our preliminary experiment showed that including gold-standard argument component labels in training did not help when predicted labels were used in the test set.

topic prompts in development data of unannotated persuasive essays to semi-automatically collect argument and domain seed words. In particular, they used 10 argument seed words: *agree, disagree, reason, support, advantage, disadvantage, think, conclusion, result, opinion*. Domain seed words are those in the topic prompts but not argument seed words or stop words. The seeds words are then used to supervise an automated extraction of argument and domain words from output of LDA topic model (Blei et al., 2003) on the development data. The extracted lexicon consists of 263 (stemmed) argument words and 1806 (stemmed) domain words mapped to 36 LDA topics.[6] All argument words are from a single LDA topic while a domain word can map to multiple LDA topics (except the topic of argument words). Using the lexicon, we extract the following Topic-context features:

Argument word: from all word pairs extracted from the source and target components, we remove those that have at least one word not in the argument word list. Each argument word pair defines a boolean feature indicating its presence in the argument component pair. We also include each argument word of the source and target components as a boolean feature which is true if the word is present in the corresponding component. We count number of common argument words, the absolute difference in number of argument words between source and target components.

Domain word count: to measure the topic similarity between the source and target components, we calculate number of common domain words, number of pairs of two domain words that share an LDA topic, number of pairs that share no LDA topic, and the absolute difference in number of domain words between the two components.

Non-domain MainVerb-Subject dependency: we extract MainVerb-Subject dependency triples, e.g., *nsubj(belive, I)*, from the source and target components, and filter out triples that involve domain words. We model each extracted triple as a boolean feature which is true if the corresponding argument component has the triple.

Finally, we include the **common feature set**.

To illustrate the Topic-context features, consider the following source and target components. Argument words are in boldface, and domain words are in italic.

Essay 54. Topic: museum and art gallery will disappear soon?

Source: **more** and **more people can** *watch exhibitions* through *television* **or** *internet* at *home* **due to** *modern technology*$_{[Premise]}$

Target: **some people think** *museums* and *art galleries* **will** *disappear soon*$_{[Claim]}$

An argument word pair is **people-think**. There are 35 pairs of domain words. A pair of two domain words that share an LDA topic is *exhibitions-art*. A pair of two domain words that do not share any LDA topic is *internet-galleries*.

5.3 Window-context Model

Our second proposed model (WINDOW) extracts features from discourse relations and common words between context sentences in the *context windows* of the source and target components.

Definition. *Context window of an argument component is a text segment formed by neighboring sentences and the covering sentence of the component. The neighboring sentences are called context sentences, and must be in the same paragraph with the component.*

In this study, context windows are determined using *window-size* heuristics.[7] Given a window-size n, we form a context window by grouping the covering sentence with at most n adjacently preceding and n adjacently following sentences that must be in the same paragraph.

To minimize noise in feature space, we require that context windows of the source and target components must be mutually exclusive. Biran and Rambow (2011) observed that the relation between a source argument and a target argument is usually instantiated by some elaboration/justification provided in a support of the source argument. Therefore we prioritize the context window of source component when it overlaps with the target context window. Particularly, we keep overlapping context sentences in the source window, and remove them from the target window. For example, with window-size 1, context windows of the Claim in sentence 2 in Figure 1 and the Premise in sentence 4 overlap at sentence 3. When the Claim is set as source component, its

[6] An LDA topic is simply represented by a number, and should not be misunderstood with essay topics.

[7] Due to the paragraph constraint and window overlapping, window-size does not indicate the actual context window size. However, window-size tells what the maximum size a window can have.

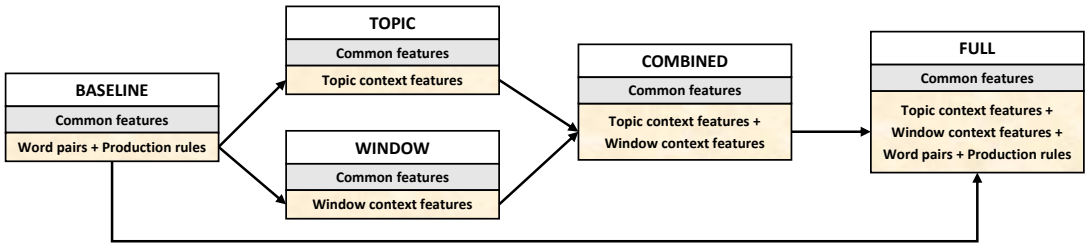


Figure 3: Features used in different models. Feature change across models are denoted by connectors.

context window includes sentences $\{2, 3\}$, and the Premise as a target has context window with only sentence 4. We extract three **Window-context** feature sets from the context windows:

Common word: as common word counts between adjacent sentences were shown useful for argument mining (Nguyen and Litman, 2016), we count common words between the covering sentence with preceding context sentences, and with following context sentences, for source and target components.

Discourse relation: for both source and target components, we extract discourse relations between context sentences, and within the covering sentence. We also extract discourse relations between each pair of source context sentence and target context sentence. Each relation defines a boolean feature. We extract both Penn Discourse Treebank (PDTB) relations (Prasad et al., 2008) and Rhetorical Structure Theory Discourse Treebank (RST-DTB) relations (Carlson et al., 2001) using publicly available discourse parsers (Ji and Eisenstein, 2014; Wang and Lan, 2015). Each PDTB relation has sense label defined in a 3-layered (class, type, subtype), e.g., *CONTINGENCY.Cause.result*. While there are only four semantic class labels at the class-level which may not cover well different aspects of argumentative relation, subtype-level output is not available given the discourse parser we use. Thus, we use relations at type-level as features. For RST-DTB relations, we use only relation labels, but ignore the nucleus and satellite labels of components as they do not provide more information given the component order in the pair. Because temporal relations were shown not helpful for argument mining tasks (Biran and Rambow, 2011; Stab and Gurevych, 2014b), we exclude them here.

Discourse marker: while the baseline model only considers discourse markers within the argument components, we define a boolean feature for each discourse marker classifying whether the marker is present before the covering sentence of the source and target components or not. This implementation aims to characterize the discourse of the preceding and following text segments of each argument component separately.

Finally, we include the **common feature set**.

5.4 Combined Model

While Window-context features are extracted from surrounding text of the argument components, which exploits the local context, the Topic-context features are an abstraction of topic-dependent information, e.g., domain words are defined within the context of topic domain (Nguyen and Litman, 2015), and thus make use of the global context of the topic domain. We believe that local and global context information represent complementary aspects of the relation between argument components. Thus, we expect to achieve the best performance by combining Window-context and Topic-context models.

5.5 Full Model

Finally, the FULL model includes all features in BASELINE and COMBINED models. That is, the FULL model is the COMBINED model plus word pairs and production rules. A summary of all models is shown in Figure 3.

6 Experiments

6.1 Task 1: Support vs. Non-support

Tuning Window-size Parameter

Because our WINDOW model uses a window-size parameter to form context windows of the source and target argument components, we investigate how the window-size of the context window impacts the prediction performance of the Window-context features. We set up a model with only Window-context features and determine the

	REPORTED	BASELINE	TOPIC	WINDOW	COMBINED	FULL
Accuracy	0.863	0.869	0.857	0.857	0.870	**0.877**
Kappa	–	0.445	0.407	0.449	**0.507***	0.481
Macro F1	0.722	0.722	0.703	0.724	**0.753***	0.739
Macro Precision	0.739	0.758	0.728	0.729	0.754	**0.777**
Macro Recall	0.705	0.699	0.685	0.720	**0.752***	0.715
F1:Support	0.519	0.519	0.488	0.533	**0.583***	0.550
F1:Non-support	0.920	0.925	0.917	0.916*	0.923	**0.929**

Table 2: Support vs. Non-support classification performances on test set. Best values are in bold. Values smaller than baseline are underlined. * indicates significantly different from the baseline ($p < 0.05$).

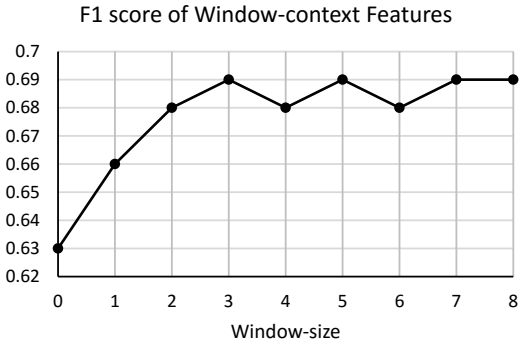


Figure 4: Performance of Window-context feature set by window-size.

window-size in range [0, 8][8] that yields the best F1 score in 10-fold cross validation. We use the training set as determined in (Stab and Gurevych, 2014b) to train/test[9] the models using LibLINEAR algorithm (Fan et al., 2008) without parameter or feature optimization. Cross-validations are conducted using Weka (Hall et al., 2009). We use Stanford parser (Klein and Manning, 2003) to perform text processing. As shown in Figure 4, while increasing the window-size from 2 to 3 improves performance (significantly), using window-sizes greater than 3 does not gain further improvement. We hypothesize that after a certain limit, larger context windows will produce more noise than helpful information for the prediction. Therefore, we set the window-size to 3 in all of our experiments involving Window-context model (all with a separate test set).

[8]Windows-size 0 means covering sentence is the only context sentence. We experimented with not using context sentence at all and obtained worse performance. Our data does not have context window with window-size 9 or larger.

[9]Note that via cross validation, in each fold some of our training set serves as a development set.

Performance on Test Set

We train all models using the training set and report their performances on the test set in Table 2. We also compare our baseline to the reported performance (REPORT) for Support vs. Non-support classification in (Stab and Gurevych, 2014b). The learning algorithm with parameters are kept the same as in the window-size tuning experiment. Given the skewed class distribution of this data, Accuracy and F1 of Non-support (the major class) are less important than Kappa, F1, and F1 of Support (the minor class). To conduct T-tests for performance significance, we split the test data into subsets by essays' ID, and record prediction performance for individual essays.

We first notice that the performances of our baseline model are better than (or equal to) REPORTED, except the Macro Recall. We reason that these performance disparities may be due to the differences in feature extractions between our implementation and Stab and Gurevych's, and also due to the minor set of new features (e.g., new predicted labels, expanded marker set, component position) that we added in our baseline.

Comparing proposed models with BASELINE, we see that WINDOW, COMBINED, and FULL models outperform BASELINE in important metrics: Kappa, F1, Recall, but TOPIC yields worse performances than BASELINE. However, the fact that COMBINED outperforms BASELINE, especially with significantly higher Kappa, F1, Recall, and F1:Support, has shown the value of Topic-context features. While Topic-context features alone are not effective, they help improve WINDOW model which supports our hypothesis that Topic-context and Window-context features are complementary aspects of context, and they together obtain better performance.

Comparing our proposed TOPIC, WINDOW,

	BASELINE	TOPIC	WINDOW	COMBINED	FULL
Accuracy	0.885	0.886	0.872	0.885	**0.887**
Kappa	0.245	0.305*	0.306*	**0.342***	0.274*
Macro F1	0.618	0.651*	0.652*	**0.670***	0.634*
Macro Precision	0.680	0.692	0.663	**0.697**	0.693
Macro Recall	0.595	0.628*	0.644*	**0.652***	0.609*
F1:Support	0.937	0.937	0.928*	0.936	**0.938**
F1:Attack	0.300	0.365*	0.376*	**0.404***	0.330*

Table 3: 5×10-fold cross validation performance of Support vs. Attack classification. * indicates significantly different from the baseline ($p < 0.01$).

COMBINED models with each other shows that COMBINED obtains the best performance while TOPIC performs the worst, which reveals that Topic-context feature set is less effective than Window-context set. While FULL model achieves the best Accuracy, Precision, and F1:Non-support, it has lower performance than COMBINED model in important metrics: Kappa, F1, F1:Support. We reason that the noise caused by word pairs and production rules even dominate the effectiveness of Topic-context and Window-context features, which degrades the overall performance.

Overall, by combining TOPIC and WINDOW models, we obtain the best performance. Most notably, we obtain the highest improvement in F1:Support, and have the best balance between Precision and Recall values among all models. These reveal that our contextual features not only dominate generic features like word pairs and production rules, but also are effective to predict minor positive class (i.e., Support).

6.2 Task 2: Support vs. Attack

To evaluate the robustness of our proposed models, we conduct an argumentative relation classification experiment that classifies a relation as Support or Attack. Because this task was not studied in (Stab and Gurevych, 2014b) and the training/test split for Support vs. Not task is not applicable here, we conduct 5×10-fold cross validation. We do not optimize the window-size parameter of the WINDOW model, and use the value 3 as set up before. Average prediction performance of all models are reported in Table 3.

Comparing our proposed models with the baseline shows that all of our proposed models significantly outperform the baseline in important metrics: Kappa, F1, F1:Attack. More notably than in the Support vs. Non-support classification, all of our proposed models predict the minor class (Attack) significantly more effectively than the baseline. The baseline achieves significantly higher F1:Support than WINDOW model. However, F1:Support of the baseline is in a tie with TOPIC, COMBINED, and FULL.

Comparing our proposed models, we see that TOPIC and WINDOW models reveal different behaviors. TOPIC model has significantly higher Precision and F1:Support, and significantly lower Recall and F1:Attack than WINDOW. Moreover, WINDOW model has slightly higher Kappa, F1, but significantly lower Accuracy. These comparisons indicate that Topic-context and Window-context features are equally effective but impact differently to the prediction. The different nature between these two feature sets is clearer than in the prior experiment, as now the classification involves classes that are more semantically different, i.e., Support vs. Attack. We recall that TOPIC model performs worse than WINDOW model in Support vs. Non-support task.

Our FULL model performs significantly worse than all of TOPIC, WINDOW, and COMBINED in Kappa, F1, Recall, and F1:Attack. Along with results from Support vs. Non-support task, this further suggests that word pairs and production rules are less effective and cannot be combined well with our contextual features.

Despite the fact that the Support vs. Attack task (Task 2) has smaller and more imbalanced data than the Support vs. Non-support (Task 1), our proposed contextual features seem to add even more value in Task 2 compared to Task 1. Using Kappa to roughly compare prediction performance across the two tasks, we observe a greater performance improvement from Baseline to Combined model in Task 2 than in Task 1. This is an evidence that our proposed context-aware features

work well even in a more imbalanced with smaller data classification task. The lower performance values of all models in Support vs. Attack than in Support vs. Non-support indirectly suggest that Support vs. Attack classification is a more difficult task. We hypothesize that the difference between support and attack exposes a deeper semantic relation than that between support and no-relation. We plan to extract textual text similarity and textual entailment features (Cabrio and Villata, 2012; Boltužić and Šnajder, 2014) to investigate this hypothesis in our future work.

7 Conclusions and Future Work

In this paper, we have presented *context-aware argumentative relation mining* that makes use of contextual features by exploiting information from topic context and context sentences. We have explored different ways to incorporate our proposed features with baseline features used in a prior study, and obtained insightful results about feature effectiveness. Experimental results show that Topic-context and Window-context features are both effective but impact predictive performance measures differently. In addition, predicting an argumentative relation will benefit most from combining these two set of features as they capture complementary aspects of context to better characterize the argumentation in justification.

The results obtained in this preliminary study are promising and encourage us to explore more directions to enable contextual features. Our next step will investigate uses of topic segmentation to identify context sentences and compare this linguistically-motivated approach to our current window-size heuristic. We plan to follow prior research on graph optimization to refine the argumentation structure and improve argumentative relation prediction. Also, we will apply our context-aware argumentative relation mining to different argument mining corpora to further evaluate its generality.

Acknowledgments

This research is supported by NSF Grant 1122504. We thank the reviewers for their helpful feedback. We also thank Christian Stab for providing us the data split for the first experiment.

References

Or Biran and Owen Rambow. 2011. Identifying Justifications in Written Dialogs by Classifying Text as Argumentative. *International Journal of Semantic Computing*, 5(4):363–381.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Filip Boltužić and Jan Šnajder. 2014. Back up your Stance: Recognizing Arguments in Online Discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58, Baltimore, Maryland, June. Association for Computational Linguistics.

Jill Burstein, Daniel Marcu, and Kevin Knight. 2003. Finding the WRITE Stuff: Automatic Identification of Discourse Structure in Student Essays. *IEEE Intelligent Systems*, 18(1):32–39, January.

Elena Cabrio and Serena Villata. 2012. Combining Textual Entailment and Argumentation Theory for Supporting Online Debates Interactions. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 208–212, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elena Cabrio, Sara Tonelli, and Serena Villata. 2013. From Discourse Analysis to Argumentation Schemes and Back: Relations and Differences. In *Computational Logic in Multi-Agent Systems*, volume 8143 of *Lecture Notes in Computer Science*, pages 1–17. Springer Berlin Heidelberg.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2001. Building a Discourse-tagged Corpus in the Framework of Rhetorical Structure Theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue - Volume 16*, SIGDIAL '01, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jianguang Du, Jing Jiang, Liu Yang, Dandan Song, and Lejian Liao. 2014. Shell Miner: Mining Organizational Phrases in Argumentative Texts in Social Media. In *Proceedings of the 2014 IEEE International Conference on Data Mining*, ICDM '14, pages 797–802, Washington, DC, USA. IEEE Computer Society.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *The Journal of Machine Learning Research*, 9:1871–1874, June.

Eirini Florou, Stasinos Konstantopoulos, Antonis Koukourikos, and Pythagoras Karampiperis. 2013. Argument extraction for supporting public policy formulation. In *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social*

Sciences, and Humanities, pages 49–54, Sofia, Bulgaria, August. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, November.

Yangfeng Ji and Jacob Eisenstein. 2014. Representation Learning for Text-level Discourse Parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13–24, Baltimore, Maryland, June. Association for Computational Linguistics.

Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

Ran Levy, Yonatan Bilu, Daniel Hershcovich, Ehud Aharoni, and Noam Slonim. 2014. Context Dependent Claim Detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1489–1500, Dublin, Ireland, August.

Marco Lippi and Paolo Torroni. 2015. Context-independent Claim Detection for Argument Mining. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 185–191, Buenos Aires, Argentina. AAAI Press.

Nitin Madnani, Michael Heilman, Joel Tetreault, and Martin Chodorow. 2012. Identifying High-Level Organizational Elements in Argumentative Discourse. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 20–28, Montrèal, Canada. Association for Computational Linguistics.

Marie-Francine Moens, Erik Boiy, Raquel Mochales Palau, and Chris Reed. 2007. Automatic Detection of Arguments in Legal Texts. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, ICAIL '07, pages 225–230, New York, NY, USA. ACM.

Huy Nguyen and Diane Litman. 2015. Extracting Argument and Domain Words for Identifying Argument Components in Texts. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 22–28, Denver, CO, June. Association for Computational Linguistics.

Huy Nguyen and Diane Litman. 2016. Improving argument mining in student essays by learning and exploiting argument indicators versus essay topics. In *Proceedings 29th International FLAIRS Conference*, Key Largo, FL, May.

Raquel Mochales Palau and Marie-Francine Moens. 2009. Argumentation Mining: The Detection, Classification and Structure of Arguments in Text. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, ICAIL '09, pages 98–107, New York, NY, USA. ACM.

Joonsuk Park and Claire Cardie. 2014. Identifying Appropriate Support for Propositions in Online User Comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2013. From Argument Diagrams to Argumentation Mining in Texts: A Survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31, January.

Andreas Peldszus and Manfred Stede. 2015a. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 938–948, Lisbon, Portugal, September. Association for Computational Linguistics.

Andreas Peldszus and Manfred Stede. 2015b. Towards Detecting Counter-considerations in Text. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 104–109, Denver, CO, June. Association for Computational Linguistics.

Andreas Peldszus. 2014. Towards segment-based recognition of argumentation structure in short texts. In *Proceedings of the First Workshop on Argumentation Mining*, pages 88–97, Baltimore, Maryland, June. Association for Computational Linguistics.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi, and Bonnie Webber. 2008. The Penn Discourse TreeBank 2.0. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-08)*, Marrakech, Morocco, May. European Language Resources Association (ELRA). ACL Anthology Identifier: L08-1093.

Vahed Qazvinian and Dragomir R. Radev. 2010. Identifying Non-explicit Citing Sentences for Citation-based Summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 555–564, Stroudsburg, PA, USA. Association for Computational Linguistics.

Diarmuid Ó Séaghdha and Simone Teufel. 2014. Unsupervised learning of rhetorical structure with untopic models. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING-14)*, Dublin, Ireland.

Alan Sergeant. 2013. Automatic argumentation extraction. In *The 10th European Semantic Web*

Conference, pages 656–660, Montpellier, France. Springer.

Christian Stab and Iryna Gurevych. 2014a. Annotating Argument Components and Relations in Persuasive Essays. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1501–1510, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Christian Stab and Iryna Gurevych. 2014b. Identifying Argumentative Discourse Structures in Persuasive Essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar. Association for Computational Linguistics.

Christian Stab, Christian Kirschner, Judith Eckle-Kohler, and Iryna Gurevych. 2014. Argumentation Mining in Persuasive Essays and Scientific Articles from the Discourse Structure Perspective. In Elena Cabrio, Serena Villata, and Adam Wyner, editors, *Proceedings of the Workshop on Frontiers and Connections between Argumentation Theory and Natural Language Processing*, pages 40–49, Bertinoro, Italy, July. CEUR-WS.

Simone Teufel and Marc Moens. 2002. Summarizing Scientific Articles: Experiments with Relevance and Rhetorical Status. *Computational Linguistics*, 28(4), December.

Douglas Walton, Christopher Reed, and Fabrizio Macagno. 2008. *Argumentation Schemes*. Cambridge University Press.

Jianxiang Wang and Man Lan. 2015. A Refined End-to-End Discourse Parser. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning - Shared Task*, pages 17–24, Beijing, China, July. Association for Computational Linguistics.

Leveraging Inflection Tables for Stemming and Lemmatization

Garrett Nicolai and **Grzegorz Kondrak**
Department of Computing Science
University of Alberta
{nicolai,gkondrak}@ualberta.ca

Abstract

We present several methods for stemming and lemmatization based on discriminative string transduction. We exploit the paradigmatic regularity of semi-structured inflection tables to identify stems in an unsupervised manner with over 85% accuracy. Experiments on English, Dutch and German show that our stemmers substantially outperform Snowball and Morfessor, and approach the accuracy of a supervised model. Furthermore, the generated stems are more consistent than those annotated by experts. Our direct lemmatization model is more accurate than Morfette and Lemming on most datasets. Finally, we test our methods on the data from the shared task on morphological reinflection.

1 Introduction

Many languages contain multiple inflected forms that correspond to the same dictionary word. Inflection is a grammatical procedure that has little impact on the meaning of the word. For example, the German words in Table 1 all refer to the action of giving. When working with these languages, it is often beneficial to establish a consistent representation across a set of inflections. This is the task that we address here.

There are two principal approaches to inflectional simplification: stemming and lemmatization. Stemming aims at removing inflectional affixes from a word form. It can be viewed as a kind of word segmentation, in which the boundaries of the stem are identified within the word; no attempt is made to restore stem changes that may occur as part of the inflection process. The goal of lemmatization is to map any inflected form to its unique *lemma*, which is typically the word form that represents a set of related inflections in a dictionary. Unlike stemming, lemmatization must always produce an actual word form.

Word form	Meaning	Tag	Stem
geben	"to give"	INF	geb
gibt	"gives"	3SIE	gib
gab	"gave"	1SIA	gab
gegeben	"given"	PP	geb

Table 1: Examples of German word-forms corresponding to the lemma *geben*.

In this paper, we present a discriminative string transduction approach to both stemming and lemmatization. Supervised stemmers require morphologically annotated corpora, which are expensive to build. We remove this constraint by extracting stems from semi-structured inflection tables, such as the one shown in Table 2, in an unsupervised manner. We design two transduction models that are trained on such stems, and evaluate them on unseen forms against a supervised model. We then extend our stemming models to perform the lemmatization task, and to incorporate an unannotated corpus. We evaluate them on several datasets.Our best system improves the state of the art for Dutch, German, and Spanish. Finally, we test our methods on the data from the shared task on morphological reinflection.

This paper is organized as follows. In Section 2, we present an overview of prior work on inflectional simplification. In Section 3, we describe our stemming methodology, followed by three types of evaluation experiments in Section 4. In Section 5, we describe our approach to lemmatization, followed by both intrinsic and extrinsic experiments in Section 6. Section 7 concludes the paper.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1138–1147,
Berlin, Germany, August 7-12, 2016.

2 Related Work

In this section, we review prior work on stemming and lemmatization.

2.1 Stemming and Segmentation

Stemming is a sub-task of the larger problem of morphological segmentation. Because of the scarcity of morphologically-annotated data, many segmentation algorithms are unsupervised or rule-based.

The Porter stemmer (Porter, 1980) and its derivatives, such as Snowball, apply hand-crafted context rules to strip affixes from a word. Creation of such rule-based programs requires significant effort and expert knowledge. We use structured inflection tables to create training data for a discriminative transducer.

Morfessor (Creutz and Lagus, 2002) and Linguistica (Goldsmith, 2001) are unsupervised word segmenters, which divide words into regularly occurring sub-sequences by applying the minimum description length (MDL) principle. While these methods are good at identifying common morphemes, they make no distinction between stems and affixes, and thus cannot be used for stemming. Morfessor Categories-MAP (Creutz and Lagus, 2004; Creutz and Lagus, 2005) distinguishes between stems and affixes, but not between derivational and inflectional affixes. We adapt a more recent version (Grönroos et al., 2014) to be used as an approximate stemmer.

Poon et al. (2009) abandons the generative model of Morfessor for a log-linear model that predicts segmentations in sequence. The discriminative approach allows for the incorporation of several priors that minimize over-segmentation. Their unsupervised model outperforms Morfessor, and they are also able to report semi- and fully-supervised results. We also approach the problem using a discriminative method, but by aligning structured inflection tables, we can take advantage of linguistic knowledge, without requiring costly annotation.

Ruokolainen et al. (2014) obtain further improvements by combining a structured perceptron CRF with letter successor variety (LSV), and the unsupervised features of Creutz and Lagus (2004). Their system is inherently supervised, while our stem annotations are derived in an unsupervised manner.

Cotterell et al. (2015) introduce Chipmunk, a fully-supervised system for labeled morphological segmentation. Extending the sequence-prediction models, Chipmunk makes use of data that is annotated not only for stem or affix, but also for inflectional role, effectively combining morphological segmentation and morphological analysis. While highly accurate, Chipmunk is limited in that it requires data that is fully-annotated for both segmentation and inflection. Our system has access to the morphological tags in inflection tables, but segmentation and tag alignment are performed in an unsupervised way.

	Singular			Plural
	1^{st}	2^{nd}	3^{rd}	1^{st}
Present	*doy*	*das*	*da*	*damos*
Imperfect	*daba*	*dabas*	*daba*	*dábamos*
Preterite	*di*	*diste*	*dio*	*dimos*
Future	*daré*	*darás*	*dará*	*daramos*

Table 2: A partial inflection table for the Spanish verb *dar* "to give".

2.2 Lemmatization

Unlike stemmers, which can be unsupervised, lemmatizers typically require annotated training data. In addition, some lemmatizers assume access to the morphological tag of the word, and/or the surrounding words in the text. Our focus is on context-free lemmatization, which could later be combined with a contextual disambiguation module.

Lemmatization is often part of the morphological analysis task, which aims at annotating each word-form with its lemma and morphological tag. Toutanova and Cherry (2009) learn a joint model for contextual lemmatization and part-of-speech prediction from a morphologically annotated lexicon. Their transduction model is tightly integrated with the POS information, which makes comparison difficult. However, in Section 6, we evaluate our approach against two other fully-supervised morphological analyzers: Morfette (Chrupała et al., 2008) and Lemming (Müller et al., 2015). Both of these systems perform lemmatization and morphological analysis in context, but can be trained to learn non-contextual models. Morfette requires morphological tags during training, while Lemming requires a morphological model constructed by its sister program, Marmot (Müller et al., 2013).

3 Stemming Methods

We approach stemming as a string transduction task. Stemming can be performed by inserting morpheme boundary markers between the stem and the affixes. For example, the German verb form *gegeben* is transduced into `ge+geb+en`, which induces the stem `geb`.

3.1 Character Alignment

The training of a transduction model requires a set of aligned pairs of source and target strings. The alignment involves every input and output character; the insertion and deletion operations are disallowed. Atomic character transformations are then extracted from the alignments.

We infer the alignment with a modified version of the M2M aligner of Jiampojamarn et al. (2007). The program applies the Expectation-Maximization algorithm with the objective to maximize the joint likelihood of its aligned source and target pairs. For our task, the source and target strings are nearly identical, except that the target includes stem-affix boundary markers. In order to account for every character in the target, which is usually longer than the source, we allow one-to-many alignment. This has the effect of tying the markers to the edge of a stem or affix. In order to encourage alignments between identical characters, we modify the aligner to generalize all identity transformations into a single match operation.

3.2 Supervised Transduction

Once we have aligned the source and target pairs, we proceed to train a *word-to-stem* transduction model for stemming unseen test instances. The word-to-stem model learns where to insert boundary markers. We refer to a model that is trained on annotated morphological segmentations as our supervised method.

We perform string transduction by adapting DIRECTL+, a tool originally designed for grapheme-to-phoneme conversion (Jiampojamarn et al., 2010). DIRECTL+ is a feature-rich, discriminative character transducer that searches for a model-optimal sequence of character transformation rules for its input. The core of the engine is a dynamic programming algorithm capable of transducing many consecutive characters in a single operation. Using a structured version of the MIRA algorithm (McDonald et al., 2005), training attempts to assign weights to each feature so that its linear model separates the gold-standard derivation from all others in its search space.

STEM|INF	**geb**|en	**setz**|en	**tu**|n
STEM|1SIA	gab|-	setz|te	tat|-
STEM|2SIE	gib|st	setz|t	tu|st
PP|STEM|PP	ge|geb|en	ge|setz|t	ge|ta|n

Table 3: Stemming of the training data based on the patterns of regularity in inflectional tables. Stemmas are shown in bold.

DIRECTL+ uses a number of feature templates to assess the quality of a rule: source context, target n-gram, and joint n-gram features. Context features conjoin the rule with indicators for all source character n-grams within a fixed window of where the rule is being applied. Target n-grams provide indicators on target character sequences, describing the shape of the target as it is being produced, and may also be conjoined with our source context features. Joint n-grams build indicators on rule sequences, combining source and target context, and memorizing frequently-used rule patterns.

Following Toutanova and Cherry (2009), we modify the out-of-the-box version of DIRECTL+ by implementing an abstract copy feature that indicates when a rule simply copies its source characters into the target, e.g. $b \rightarrow b$. The copy feature has the effect of biasing the transducer towards preserving the source characters during transduction.

3.3 Unsupervised Segmentation

In order to train a fully-supervised model for stemming, large lists of morphologically-segmented words are generally required. While such annotated corpora are rare, semi-structured, crowd-sourced inflection tables are available for many languages on websites such as Wiktionary (Table 2). In this section, we introduce an unsupervised method of inducing stems by leveraging paradigmatic regularity in inflection tables.

Sets of inflection tables often exhibit the same inflectional patterns, called paradigms, which are based on phonological, semantic, or morphological criteria (cf. Table 3). Each table consists of lists of word forms, including the lemma. The number of distinct stems, such as *'geb'* and *'gib'* for the verb *geben*, is typically very small, averaging slightly over two per German verb inflection table.

Source	g	i	b	t
Target	g	i	b	+t
Tags	STEM			3SIE
Joint	g	**e**	b	+3SIE

Table 4: Alignment of the various representations of the word *gibt*.

	Words	Noun	Verb	Adj
English	50,155	2	5	3
Dutch	101,667	2	9	3
German	96,038	8	27	48

Table 5: The number of words and distinct inflections for each language in the CELEX datasets.

The number of distinct affix forms corresponding to the same inflectional form across different lemmas is also small, averaging below three for German verbs. For example, the second person singular indicative present suffix is always either *-st*, *-est*, or *-t*.

We take advantage of this relative consistency to determine the boundaries between the stems and affixes of each word form in an unsupervised manner. We first associate each word form in the training data with an abstract tag sequence, which is typically composed of the `STEM` tag and a suffix tag representing a given inflection slot (Table 3). We then apply the unsupervised aligner to determine the most likely alignment between the character sequences and the tags, which are treated as indivisible units. The aligner simultaneously learns common representations for stems within a single inflection table, as well as common representations for each affix across multiple tables.

Some inflections, such as the German past participle (`PP` in Table 3) involve a *circumfix*, which can be analyzed as a prefix-suffix combination. Prior to the alignment, we associate all forms that belong to the inflection slots involving circumfixation with tag sequences composed of three tags. Occasionally, a word form will only have a suffix where one would normally expect a circumfix (e.g. *existiert*). In order to facilitate tag alignment in such cases, we prepend a dummy null character to each surface word form.

After the stem-affix boundaries have been identified, we proceed to train a *word-to-stem* transduction model as described in Section 3.2. We refer to this unsupervised approach as our basic method (cf. Figure 1).

3.4 Joint Stemming and Tagging

The method described in the previous section fails to make use of a key piece of information in the inflection table: the lemma. The stem of an inflected form is typically either identical or very similar to the stem of its lemma, or *stemma* (Table 3). Our joint method takes advantage of this similarity by transducing word-forms into stemmas with tags.

The format of the training data for the *word-to-stemma* model is different from the *word-to-stem* model. After the initial segmentation of the source word-forms into morphemes by the unsupervised aligner, as described in Section 3.3, the stems are replaced with the corresponding stemmas, and the affixes are replaced with the inflection tags. For example, the form *gibt* is paired with the sequence `geb+3SIE`, with the stem and stemma re-aligned at the character level as shown in Table 4.

Unlike the basic method, which simply inserts morpheme breaks into word-forms, the joint method uses the tags to identify the boundaries between stems and affixes. At test time, the input word-form is transduced into a stemma and tag sequence. The character string that has generated the tag is then stripped from the input word-form to obtain the stem. By making use of both the tags and the stemma, the *word-to-stemma* model jointly optimizes the stem and affix combination. We refer to this unsupervised approach as our joint method.

4 Stemming Experiments

Precise evaluation of stemming methods requires morphologically annotated lexicons, which are rare. Unlike lemmas, stems are abstract representations, rather than actual word forms. Unsurprisingly, annotators do not always agree on the segmentation of a word. In this section, we describe three experiments for evaluating stem extraction, intrinsic accuracy, and consistency.

We evaluate our methods against three systems that are based on very different principles. Snowball[1] is a rule-based program based on the methodology of the Porter Stemmer. Morfessor FlatCat (Grönroos et al., 2014) performs unsupervised morphological segmentation, and approximates stemming by distinguishing stems and af-

[1] http://snowball.tartarus.org

	EN	NL	DE
Our method	85.9	88.0	85.7
Snowball	48.2	58.8	49.5
Morfessor	61.4	71.4	61.4

Table 6: Unsupervised stemming accuracy of the CELEX training set.

fixes.[2] Chipmunk (Cotterell et al., 2015), is a fully-supervised system that represents the current state of the art.

4.1 Data

We perform an evaluation of stemming on English (EN), Dutch (NL), and German (DE) lexicons from CELEX (Baayen et al., 1995). The three languages vary in terms of morphological complexity (Table 5). We use the morphological boundary annotations for testing all stemming systems, as well as for training our supervised system.

For both unsupervised systems, we could build training sets from any inflection tables that contain unsegmented word-forms. However, in order to perform a precise comparison between the supervised and unsupervised systems, we extract the inflection tables from CELEX, disregarding the segmentation information. Each system is represented by a single stemming model that works on nouns, verbs, and adjectives. Due to differences in representation, the number of training instances vary slightly between models, but the number of words is constant (Table 5).

In order to demonstrate that our unsupervised methods require no segmentation information, we create additional German training sets using the inflection tables extracted from Wiktionary by Durrett and DeNero (2013). The sets contain 18,912 noun forms and 43,929 verb forms. We derive separate models for verbs and nouns in order to compare the difficulty of stemming different parts of speech.

The test sets for both CELEX and Wiktionary data come from CELEX, and consist of 5252, 6155, and 9817 unique forms for English, Dutch, and German, respectively. The German test set contains 2620 nouns, 3837 verbs, and 3360 adjectives.

Chipmunk[3] requires training data in which every morpheme of a word is annotated for morphological function. Since this information is not included in CELEX, we train and test Chipmunk, as well as a version of our supervised model, on the data created by Cotterell et al. (2015), which is much smaller. The English and German segmentation datasets contain 1161 and 1266 training instances, and 816 and 952 test instances, respectively.

	EN	NL	DE
Supervised	98.5	96.0	91.2
Basic	82.3	89.1	80.9
Joint	94.6	93.2	86.0
Snowball	50.0	58.4	48.2
Morfessor	65.2	60.9	51.8

Table 7: Stemming accuracy of systems trained and tested on CELEX datasets.

4.2 Stem Extraction Evaluation

First, we evaluate our unsupervised segmentation approach, which serves as the basis for our basic and joint models, on the union of the training and development parts of the CELEX dataset. We are interested how often the stems induced by the method described in Section 3.3 match the stem annotations in the CELEX database.

The results are presented in Table 6. Our method is substantially more accurate than either Snowball or Morfessor. Snowball, despite being called a stemming algorithm, often eliminates derivational affixes; e.g. `able` in *unbearable*. Morfessor makes similar mistakes, although less often. Our method tends to prefer longer stems and shorter affixes. For example, it stems *verwandtestem*, as `verwandte`, while CELEX has `verwandt`.

4.3 Intrinsic Evaluation

The results of the intrinsic evaluation of the stemming accuracy on unseen forms in Tables 7-9 demonstrate the quality of our three models. The joint model performs better than the basic model, and approaches the accuracy of the supervised model. On the CELEX data, our unsupervised joint model substantially outperforms Snowball and Morfessor on all three languages (Table 7).[4]

[2] Morfessor is applied to the union of the training and test data.

[3] http://cistern.cis.lmu.de/chipmunk

[4] The decrease in Morfessor accuracy between Tables 6 and 7 can be attributed to a different POS distribution between training and testing.

	Noun	Verb
Basic	76.8	90.3
Joint	85.2	91.1
Snowball	55.5	39.8
Morfessor	61.9	34.9

Table 8: German stemming accuracy of systems trained on Wiktionary data, and tested on the CELEX data.

	EN	DE
Supervised	94.7	85.1
Chipmunk	94.9	87.4

Table 9: Stemming accuracy of systems trained and tested on the Chipmunk data.

These results are further confirmed on the German Wiktionary data (Table 8). Our supervised model performs almost as well as Chipmunk on its dataset (Table 9).

A major advantage of the joint model over the basic model is its tag awareness (cf. Table 4). Although the tags are not always correctly recovered on the test data, they often allow the model to select the right analysis. For example, the basic model erroneously segments the German form *erklärte* as `erklärt+e` because `+e` is a common verbal, adjectival and nominal suffix. The joint model, recognizing `er` as a verbal derivational prefix, predicts a verbal inflection tag (`+1SIA`), and the correct segmentation `erklär+te`. Verbal stems are unlikely to end in `ärt`, and `+te`, unlike `+e`, can only be a verbal suffix.

4.4 Consistency Evaluation

When stemming is used for inflectional simplification, it should ideally produce the same stem for all word-forms that correspond to a given lemma. In many cases, this is not an attainable goal because of internal stem changes (cf. Table 1). However, most inflected words follow regular paradigms, which involve no stem changes. For example, all forms of the Spanish verb *cantar* contain the substring `cant`, which is considered the common stem. We quantify the extent to which the various systems approximate this goal by calculating the average number of unique generated stems per inflection table in the CELEX test sets.[5]

	EN	NL	DE
Gold	1.10	1.17	1.30
Supervised	1.13	1.64	1.50
Basic	1.06	1.21	1.25
Joint	1.09	1.08	1.20
Snowball	1.03	1.45	2.02
Morfessor	1.11	1.68	3.27

Table 10: Average number of stems per lemma.

The results are presented in Table 10. The stems-per-table average tends to reflect the morphological complexity of a language. All systems achieve excellent consistency on English, but the Dutch and German results paint a different picture. The supervised system falls somewhat short of emulating the gold segmentations, which may be due to the confusion between different parts of speech. In terms of consistency, the stems generated by our unsupervised methods are superior to those of Snowball and Morfessor, and even to the gold stems. We attribute this surprising result to the fact that the EM-based alignment of the training data favors consistency in both stems and affixes, although this may not always result in the correct segmentation.

5 Lemmatization Methods

In this section, we present three supervised lemmatization methods, two of which incorporate the unsupervised stemming models described in Section 3. The different approaches are presented schematically in Figure 1, using the example of the German past participle *gedacht*.

5.1 Stem-based Lemmatization

Our stem-based lemmatization method is an extension of our basic stemming method. We compose the *word-to-stem* transduction model from Section 3 with a *stem-to-lemma* model that converts stems into lemmas. The latter is trained on character-aligned pairs of stems and lemmas, where stems are extracted from the inflection tables via the unsupervised method described in Section 3.3.

[5] Chipmunk is excluded from the consistency evaluation because its dataset is not composed of complete inflection tables.

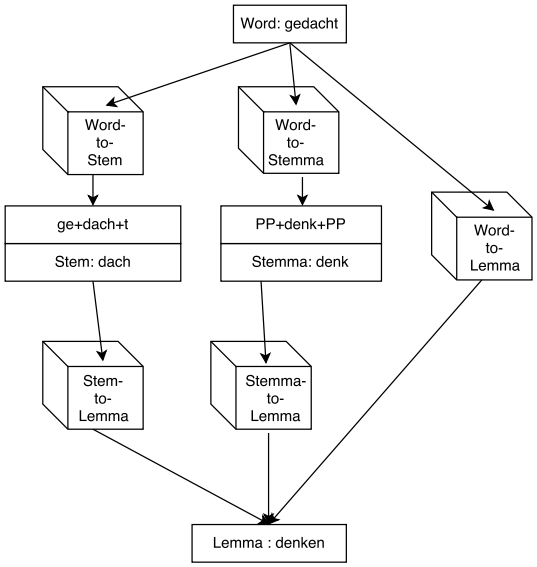


Figure 1: Three lemmatization methods.

5.2 Stemma-based Lemmatization

Our stemma-based lemmatization method is an extension of our joint stemming method. We compose the *word-to-stemma* transduction model described in Section 3.4 with a *stemma-to-lemma* model that converts stems into lemmas. The latter is trained on character-aligned pairs of stemmas and lemmas, where stemmas are extracted via the method described in Section 3.4. Typically, the model simply appends a lemmatic affix to the stemma, as all stem changes are handled by the *word-to-stemma* model.

5.3 Direct Lemmatization

Our final lemmatization method is a *word-to-lemma* transduction model that directly transforms word-forms into lemmas and tags. The model is trained on word-forms paired with their lemmas and inflectional tags, which are easily obtained from the inflection tables. A potential advantage of this method lies in removing the possibility of error propagation that is inherent in pipeline approaches. However, it involves a more complex transduction model that must simultaneously apply both stem changes, and transform inflectional affixes into lemmatic ones.

5.4 Re-ranking

Intuitively, lemmatization accuracy could be improved by leveraging large, unannotated corpora. After generating n-best lists of possible lemmas, we re-rank them using the method of Joachims (2002) implemented with the Liblinear SVM tool (Fan et al., 2008). We employ four features of the prediction:

1. normalized score from DIRECTL+,
2. rank in the n-best list
3. presence in the corpus,
4. normalized likelihood from a 4-gram character language model derived from the corpus.

6 Lemmatization Experiments

Unlike stemming, lemmatization is a completely consistent process: all word-forms within an inflection table correspond to the same lemma. In this section, we describe intrinsic and extrinsic experiments to evaluate the quality of the lemmas generated by our systems, and compare the results against the current state of the art.

6.1 Data

As in our stemming experiments, we extract complete English, Dutch, and German inflection tables from CELEX. We use the same data splits as in Section 4.1. We also evaluate our methods on Spanish verb inflection tables extracted from Wiktionary by Durrett and DeNero (2013), using the original data splits. Spanish is a Romance language, with a rich verbal morphology comprising 57 inflections for each lemma.

A different type of dataset comes from the CoNLL-2009 Shared Task (Hajič et al., 2009). Unlike the CELEX and Wiktionary datasets, they are extracted from an annotated text, and thus contain few complete inflection tables, with many lemmas represented by a small number of word-forms. We extract all appropriate parts-of-speech from the test section of the corpus for English, German, and Spanish. This results in a test set of 5165 unique forms for English, 6572 for German, and 2668 for Spanish.

For re-ranking, we make use of a word list constructed from the first one million lines of the appropriate Wikipedia dump.[6] A character language model is constructed using the CMU Statistical Language Modeling Toolkit.[7] 20% of the development set is reserved for the purpose of training a re-ranking model. For Lemming and Morfette, we provide a lexicon generated from the corpus.

Spanish marks unpredictable stress by marking a stressed vowel with an acute accent (e.g. *cantó*

[6] All dumps are from November 2, 2015.

[7] http://www.speech.cs.cmu.edu

	Wiki	CELEX			CoNLL		
	ES	EN	NL	DE	EN	DE	ES
Stem-based	97.1	89.1	82.3	76.3	90.2	71.1	83.2
Stemma-based	94.5	96.4	85.2	85.8	**92.5**	75.9	91.2
Direct	**98.8**	96.4	**89.5**	**88.7**	**92.5**	**80.1**	**91.5**
Morfette	98.0	96.0	80.2	81.3	**92.5**	73.5	**91.5**
Lemming	98.6	**96.7**	86.6	88.2	**92.5**	77.9	90.4

Table 11: Lemmatization results without the use of a corpus.

vs. *canto*). In order to facilitate generalization, we perform a lossless pre-processing step that replaces all accented vowels with their unaccented equivalent followed by a special stress symbol (e.g. `canto'`). For consistency, this modification is applied to the data for each system.

6.2 Intrinsic Evaluation

We evaluate lemmatization using word accuracy. In cases where a surface word-form without a morphological tag may correspond to multiple lemmas, we judge the prediction as correct if it matches any of the lemmas. For example, both the noun *Schrei* and the verb *schreien* are considered to be correct lemmas for the German word *schreien*.[8]

The results without the use of a corpus are shown in Table 11. Thanks to its tag awareness, the stemma-based method is more accurate than the stem-based method, except on the verb-only Spanish Wiktionary dataset. However, our best method is the direct *word-to-lemma* model, which outperforms both Morfette and Lemming on most datasets.

We interpret the results as the evidence for the effectiveness of our discriminative string transduction approach. The direct model is superior to the stemma-based model because it avoids any information loss that may occur during an intermediate stemming step. However, it is still able to take advantage of the tag that it generates together with the target lemma. For example, Lemming incorrectly lemmatizes the German noun form *Verdienste* "earnings" as `verdien` because `+ste` is a superlative adjective suffix. Our direct model, however, considers `dien` to be an unlikely ending for an adjective, and instead produces the correct lemma *Verdienst*.

The results with the use of a corpus are shown in Table 12. We omit the results on Spanish Wiktionary and on both English datasets, which are almost identical to those in Table 11. We observe that both the stemma-based and direct methods achieve a substantial error rate reduction on the Dutch and German datasets, while Lemming improvements are minimal.[9] The Spanish CoNLL results are different: only the stem-based and stemma-based methods benefit noticeably from re-ranking.

	CELEX		CoNLL	
	NL	DE	DE	ES
Stem-based	82.3	76.9	71.9	90.6
Stemma-based	87.3	88.4	79.0	**93.3**
Direct	**92.4**	**90.0**	**81.3**	91.9
Lemming	86.9	88.5	77.9	90.6

Table 12: Lemmatization results boosted with a raw corpus.

Error analysis indicates that the re-ranker is able to filter non-existent lemmas, such as `wint` for *Winter*, and `endstadie` for *Endstadien*, instead of *Endstadium*. In general, the degree of improvement seems to depend on the set of randomly selected instances in the held-out set used for training the re-ranker. If a base model achieves a very high accuracy on the held-out set, the re-ranker tends to avoid correcting the predictions on the test set.

6.3 Extrinsic Evaluation

We perform our final evaluation experiment on the German dataset[10] from the SIGMORPHON shared task on morphological reinflection (Cot-

[8] The capitalization of German nouns is ignored.

[9] We were unable to obtain any corpus improvement with Morfette.

[10] http://sigmorphon.org/sharedtask

	Task 1	Task 3
Baseline	89.4	81.5
Chipmunk	82.0	88.3
Stem-based	86.9	89.3
Stemma-based	84.0	89.5
Lemma-based	n/a	90.7
Source-Target	94.8	88.2

Table 13: Accuracy on the German dataset from the shared task on morphological reinflection.

terell et al., 2016).[11] The task of inflection generation (Task 1) is to produce a word-form given a lemma and an abstract inflectional tag. The task of unlabeled reinflection (Task 3) takes as input an unannotated inflected form instead of a lemma.

We evaluate four different methods that combine the models introduced in this paper. For Task 1, the stem-based method composes a *lemma-to-stem* and a *stem-to-word* models; the stemma-based method is similar, but pivots on stemmas instead; and the source-target method is a *lemma-to-word* model. For Task 3, a *word-to-lemma* model is added in front of both the stem-based and stemma-based methods; the lemma-based method composes a *word-to-lemma* and a *lemma-to-word* models; and the source-target method is a *word-to-word* model. In addition, we compare with a method that is similar to our *stem-based* method, but pivots on Chipmunk-generated stems instead. As a baseline, we run the transduction method provided by the task organizers.

The results are shown in Table 13. On Task 1, none of the stemming approaches is competitive with a direct *lemma-to-word* model. This is not surprising. First, the lemmatic suffixes provide information regarding part-of-speech. Second, the stemmers fail to take into account the fact that the source word-forms are lemmas. For example, the German word *überhitzend* "overheated" can either be an adjective, or the present participle of the verb *überhitzen*; if the word is a lemma, it is obviously the former.

The *lemma-based* method is the best performing one on Task 3. One advantage that it has over the *word-to-word* model lies in the ability to reduce the potentially quadratic number of transduction operations between various related word-forms to a linear number of transduction operations between the word-forms and their lemmas, and vice-versa.

[11] We use the development sets for this evaluation because the target sides of the test sets have not been publicly released.

7 Conclusion

We have presented novel methods that leverage readily available inflection tables to produce high-quality stems and lemmas. In the future, we plan to expand our method to predict morphological analyses, as well as to incorporate other information such as parts-of-speech.

Acknowledgments

This research was supported by the Natural Sciences and Engineering Research Council of Canada, and the Alberta Innovates Technology Futures.

References

Harald R. Baayen, Richard Piepenbrock, and Leon Gulikers. 1995. *The CELEX Lexical Database. Release 2 (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, Pennsylvania.

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. 2008. Learning morphology with Morfette. In *LREC*.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015. Labeled morphological segmentation with semi-markov models. *CoNLL 2015*, page 164.

Ryan Cotterell, Christo Kirov, John Sylak-Glassman, David Yarowsky, Jason Eisner, and Mans Hulden. 2016. The SIGMORPHON 2016 shared task—morphological reinflection. In *SIGMORPHON*.

Mathias Creutz and Krista Lagus. 2002. Unsupervised discovery of morphemes. In *Proceedings of the ACL-02 workshop on Morphological and phonological learning-Volume 6*, pages 21–30.

Mathias Creutz and Krista Lagus. 2004. Induction of a simple morphology for highly-inflecting languages. In *Proceedings of the 7th Meeting of the ACL Special Interest Group in Computational Phonology: Current Themes in Computational Phonology and Morphology*, pages 43–51.

Mathias Creutz and Krista Lagus. 2005. Inducing the morphological lexicon of a natural language from unannotated text. In *Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning (AKRR05)*, volume 1(106-113), pages 51–59.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *HLT-NAACL*, pages 1185–1195.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

John Goldsmith. 2001. Unsupervised learning of the morphology of a natural language. *Computational linguistics*, 27(2):153–198.

Stig-Arne Grönroos, Sami Virpioja, Peter Smit, and Mikko Kurimo. 2014. Morfessor FlatCat: An HMM-based method for unsupervised and semi-supervised learning of morphology. In *COLING*, pages 1177–1185.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *CoNLL*, pages 1–18.

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *NAACL-HLT*, pages 372–379.

Sittichai Jiampojamarn, Colin Cherry, and Grzegorz Kondrak. 2010. Integrating joint n-gram features into a discriminative training network. In *NAACL-HLT*.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *ACL*.

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *EMNLP*, pages 322–332.

Thomas Müller, Ryan Cotterell, and Alexander Fraser. 2015. Joint lemmatization and morphological tagging with LEMMING. In *EMNLP*.

Hoifung Poon, Colin Cherry, and Kristina Toutanova. 2009. Unsupervised morphological segmentation with log-linear models. In *NAACL-HLT*, pages 209–217.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

Teemu Ruokolainen, Oskar Kohonen, Sami Virpioja, and Mikko Kurimo. 2014. Painless semi-supervised morphological segmentation using conditional random fields. *EACL*, page 84.

Kristina Toutanova and Colin Cherry. 2009. A global model for joint lemmatization and part-of-speech prediction. In *ACL*, pages 486–494.

Scaling a Natural Language Generation System

Jonathan Pfeil
Department of EECS
Case Western Reserve University
Cleveland, OH, USA
jonathan.pfeil@case.edu

Soumya Ray
Department of EECS
Case Western Reserve University
Cleveland, OH, USA
sray@case.edu

Abstract

A key goal in natural language generation (NLG) is to enable fast generation even with large vocabularies, grammars and worlds. In this work, we build upon a recently proposed NLG system, Sentence Tree Realization with UCT (STRUCT). We describe four enhancements to this system: (i) pruning the grammar based on the world and the communicative goal, (ii) intelligently caching and pruning the combinatorial space of semantic bindings, (iii) reusing the lookahead search tree at different search depths, and (iv) learning and using a search control heuristic. We evaluate the resulting system on three datasets of increasing size and complexity, the largest of which has a vocabulary of about 10K words, a grammar of about 32K lexicalized trees and a world with about 11K entities and 23K relations between them. Our results show that the system has a median generation time of 8.5s and finds the best sentence on average within 25s. These results are based on a sequential, interpreted implementation and are significantly better than the state of the art for planning-based NLG systems.

1 Introduction and Related Work

We consider the restricted natural language generation (NLG) problem (Reiter and Dale, 1997): given a grammar, lexicon, world and a communicative goal, output a valid sentence that satisfies this goal. Though restricted, this problem is still challenging when the NLG system has to deal with the large probabilistic grammars of natural language, large knowledge bases representing realistic worlds with many entities and relations between them, and complex communicative goals.

Prior work has approach NLG from two directions. One strategy is over-generation and ranking, in which an intermediate structure generates many candidate sentences which are then ranked according to how well they match the goal. This includes systems built on chart parsers (Shieber, 1988; Kay, 1996; White and Baldridge, 2003), systems that use forest architectures such as HALogen/Nitrogen, (Langkilde-Geary, 2002), systems that use tree conditional random fields (Lu et al., 2009), and newer systems that use recurrent neural networks (Wen et al., 2015b; Wen et al., 2015a). Another strategy formalizes NLG as a goal-directed planning problem to be solved using an automated planner. This plan is then semantically enriched, followed by *surface realization* to turn it into natural language. This is often viewed as a *pipeline* generation process (Reiter and Dale, 1997).

An alternative to pipeline generation is *integrated generation*, in which the sentence planning and surface realization tasks happen simultaneously (Reiter and Dale, 1997). CRISP (Koller and Stone, 2007) and PCRISP (Bauer and Koller, 2010) are two such systems. These generators encode semantic components and grammar actions in PDDL (Fox and Long, 2003), the input format for many off-the-shelf planners such as Graphplan (Blum and Furst, 1997). During the planning process a semantically annotated parse is generated alongside the sentence, preventing ungrammatical sentences and structures that cannot be realized. PCRISP builds upon the CRISP system by incorporating grammar probabilities as costs in an off-the-shelf metric planner (Bauer and Koller, 2010). Our work builds upon the Sentence Tree Realization with UCT (STRUCT) system (McKinley and Ray, 2014), described further in the next section. STRUCT performs integrated generation by for-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1148–1157,
Berlin, Germany, August 7-12, 2016.

malizing the generation problem as planning in a Markov decision process (MDP), and using a probabilistic planner to solve it.

Results reported in previous work (McKinley and Ray, 2014) show that STRUCT is able to correctly generate sentences for a variety of communicative goals. Further, the system scaled better with grammar size (in terms of vocabulary) than CRISP. Nonetheless, these experiments were performed with toy grammars and worlds with artificial communicative goals written to test specific experimental variables in isolation. In this work, we consider the question: can we enable STRUCT to scale to realistic generation tasks? For example, we would like STRUCT to be able to generate any sentence from the Wall Street Journal (WSJ) corpus (Marcus et al., 1993). We describe four enhancements to the STRUCT system: (i) pruning the grammar based on the world and the communicative goal, (ii) intelligently caching and pruning the combinatorial space of semantic bindings, (iii) reusing the lookahead search tree at different search depths, and (iv) learning and using a search control heuristic. We call this enhanced version Scalable-STRUCT (S-STRUCT). In our experiments, we evaluate S-STRUCT on three datasets of increasing size and complexity derived from the WSJ corpus. Our results show that even with vocabularies, grammars and worlds containing tens of thousands of constituents, S-STRUCT has a median generation time of 8.5s and finds the best sentence on average within 25s, which is significantly better than the state of the art for planning-based NLG systems.

2 Background: LTAG and STRUCT

STRUCT uses an MDP (Puterman, 1994) to formalize the NLG process. The states of the MDP are semantically-annotated partial sentences. The actions of the MDP are defined by the rules of the grammar. STRUCT uses a probabilistic lexicalized tree adjoining grammar (PLTAG).

Tree Adjoining Grammars (TAGs) (Figure 1) consist of two sets of trees: initial trees and auxiliary (adjoining) trees. An initial tree can be applied to an existing sentence tree by replacing a leaf node whose label matches the initial tree's root label in an action called "substitution". Auxiliary trees have a special "foot" node whose label matches the label of its root, and uses this to encode recursive language structures. Given an existing sentence tree, an auxiliary tree can be applied in a three-step process called "adjunction". First, an adjunction site is selected from the sentence tree; that is, any node whose label matches that of the auxiliary tree's root and foot. Then, the subtree rooted by the adjunction site is removed from the sentence tree and substituted into the foot node of the auxiliary tree. Finally, the modified auxiliary tree is substituted back into the original adjunction location. LTAG is a variation of TAG in which each tree is associated with a lexical item known as an anchor (Joshi and Schabes, 1997). Semantics can be added to an LTAG by annotating each tree with compositional lambda semantics that are unified via β-reduction (Jurafsky and Martin, 2000). A PLTAG associates probabilities with every tree in the LTAG and includes probabilities for starting a derivation, probabilities for substituting into a specific node, and probabilities for adjoining at a node, or not adjoining.

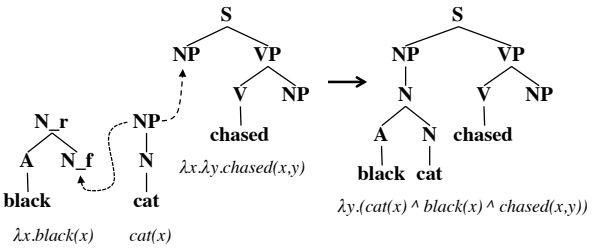


Figure 1: LTAG examples: initial tree (chased), substitution (cat), and adjunction (black)

The STRUCT reward function is a measure of progress towards the communicative goal as measured by the overlap with the semantics of a partial sentence. It gives positive reward to subgoals fulfilled and gives negative reward for unbound entities, unmet semantic constraints, sentence length, and ambiguous entities. Therefore, the best sentence for a given goal is the shortest unambiguous sentence which fulfills the communicative goal and all semantic constraints. The transition function of the STRUCT MDP assigns the total probability of selecting and applying an action in a state to transition to the next, given by the action's probability in the grammar. The final component of the MDP is the discount factor, which is set to 1. This is because with lexicalized actions, the state does not loop, and the algorithm may need to generate long sentences to match the communicative goal.

STRUCT uses a modified version of the probabilistic planner UCT (Kocsis and Szepesvári, 2006), which can generate near-optimal plans

with a time complexity independent of the state space size. UCT's online planning happens in two steps: for each action available, a lookahead search tree is constructed to estimate the action's utility. Then, the best available action is taken and the procedure is repeated. If there are any unexplored actions, UCT will choose one according to an "open action policy" which samples PLTAGs without replacement. If no unexplored actions remain, an action a is chosen in state s according to the "tree policy" which maximizes Equation 1.

$$P(s,a) = Q(s,a) + c\sqrt{\frac{lnN(s)}{N(s,a)}} \qquad (1)$$

Here $Q(s,a)$ is the estimated value of a, computed as the sum of expected future rewards after (s,a). $N(s,a)$ and $N(s)$ are the visit counts for s and (s,a) respectively. c is a constant term controlling the exploration/exploitation trade off. After an action is chosen, the policy is rolled out to depth D by repeatedly sampling actions from the PLTAG, thereby creating the lookahead tree.

UCT was originally used in an adversarial environment, so it selects actions leading to the best average reward; however, language generation is not adversarial, so STRUCT chooses actions leading to the best overall reward instead.

Algorithm 1 S-STRUCT Algorithm

Require: Grammar R, World W, Goal G, num trials N, lookahead depth D, timeout T

1: $^{\dagger}R \leftarrow pruneGrammar(R)$
2: $state \leftarrow$ empty state
3: $uctTree \leftarrow$ new search tree at $state$
4: **while** $state$ not terminal **and** time $<$ T **do**
5: $\quad ^{\dagger}uctTree \leftarrow getAction(uctTree, N, D)$
6: $\quad state \leftarrow uctTree.state$
7: **end while**
8: **return** $extractBestSentence(uctTree)$

The modified STRUCT algorithm presented in this paper, which we call Scalable-STRUCT (S-STRUCT), is shown in Algorithm 1. If the changes described in the next section (lines marked with †) are removed, we recover the original STRUCT system.

3 Scaling the STRUCT system

In this section, we describe five enhancements to STRUCT that will allow it to scale to real world NLG tasks. Although the implementation details of these are specific to STRUCT, all but one (reuse of the UCT search tree) could theoretically be applied to any planning-based NLG system.

Algorithm 2 getAction (Algorithm 1, line 5)

Require: Search Tree $uctTree$, num trials N, lookahead depth D, grammar R

1: **for** N **do**
2: $\quad node \leftarrow uctTree$
3: $\quad$ **if** $node.state$ has unexplored actions **then**
4: $\quad\quad ^{\dagger}action \leftarrow$ pick with open action policy
5: $\quad$ **else**
6: $\quad\quad ^{\dagger}action \leftarrow$ pick with tree policy
7: $\quad$ **end if**
8: $\quad ^{\dagger}node \leftarrow applyAction(node, action)$
9: $\quad depth \leftarrow 1$
10: $\quad$ **while** $depth < D$ **do**
11: $\quad\quad action \leftarrow$ sample PLTAG from R
12: $\quad\quad ^{\dagger}node \leftarrow applyAction(node, action)$
13: $\quad\quad reward \leftarrow calcReward(node.state)$
14: $\quad\quad$ propagate $reward$ up $uctTree$
15: $\quad\quad depth \leftarrow depth + 1$
16: $\quad$ **end while**
17: **end for**
18: $uctTree \leftarrow$ best child of $uctTree$
19: **return** $uctTree$

3.1 Grammar Pruning

It is clear that for a given communicative goal, only a small percentage of the lexicalized trees in the grammar will be helpful in generating a sentence. Since these trees correspond to actions, if we prune the grammar suitably, we reduce the number of actions our planner has to consider.

Algorithm 3 pruneGrammar (Algorithm 1, line 1)

Require: Grammar R, World W, Goal G

1: $G' \leftarrow \emptyset$
2: **for** $e \in G$.entities **do**
3: $\quad G' \leftarrow G' \cup referringExpression(e, W)$
4: **end for**
5: $R' \leftarrow \emptyset$
6: **for** $tree \in R$ **do**
7: $\quad$ **if** $tree$ fulfills semantic constraints **or** $tree$.relations $\subseteq G'$.relations **then**
8: $\quad\quad R' \leftarrow R' \cup \{tree\}$
9: $\quad$ **end if**
10: **end for**
11: **return** R'

There are four cases in which an action is rele-

vant. First, the action could directly contribute to the goal semantics. Second, the action could satisfy a semantic constraint, such as mandatory determiner adjunction which would turn "cat" into "the cat" in Figure 1. Third, the action allows for additional beneficial actions later in the generation. An auxiliary tree anchored by "that", which introduces a relative clause, would not add any semantic content itself. However, it would add substitution locations that would let us go from "the cat" to "the cat that chased the rabbit" later in the generation process. Finally, the action could disambiguate entities in the communicative goal. In the most conservative approach, we cannot discard actions that introduce a relation sharing an entity with a goal entity (through any number of other relations), as it may be used in a referring expression (Jurafsky and Martin, 2000). However, we can optimize this by ensuring that we can find at least *one*, instead of *all*, referring expressions.

This grammar pruning is "lossless" in that, after pruning, the full communicative goal can still be reached, all semantic constraints can be met, and all entities can be disambiguated. However it is possible that the solution found will be longer than necessary. This can happen if we use two separate descriptors to disambiguate two entities where one would have sufficed. For example, we could generate the sentence "the black dog chased the red cat" where saying "the large dog chased the cat" would have sufficed (if "black", "red", and "large" were only included for disambiguation purposes).

We implement the pruning logic in the $pruneGrammar$ algorithm shown in Algorithm 3. First, an expanded goal G' is constructed by explicitly solving for a referring expression for each goal entity and adding it to the original goal. The algorithm is based on prior work (Bohnet and Dale, 2005) and uses an alternating greedy search, which chooses the relation that eliminates the most distractors, and a depth-first search to describe the entities. Then, we loop through the trees in the grammar and only keep those that can fulfill semantic constraints or can contribute to the goal. This includes trees introducing relative clauses.

3.2 Handling Semantic Bindings

As a part of the reward calculation in Algorithm 4, we must generate the valid bindings between the entities in the partial sentence and the entities in the world (line 2). We must have at least one valid binding, as this indicates that our partial sentence is factual (with respect to the world); however, more than one binding means that the sentence is ambiguous, so a penalty is applied. Unfortunately, computing the valid bindings is a combinatorial problem. If there are N world entities and K partial sentence entities, there are $\binom{N}{K}$ bindings between them that we must check for validity. This quickly becomes infeasible as the world size grows.

Algorithm 4 calcReward (Algorithm 2, line 13)

Require: Partial Sentence S, World W, Goal G

1: $score \leftarrow 0$
2: $^{\dagger}B \leftarrow getValidBindings(S, W)$
3: **if** $|B| > 0$ **then**
4: $\quad ^{\dagger}m \leftarrow getValidBinding(S, G)$
5: $\quad S \leftarrow$ apply m to S
6: $\quad score \mathrel{+}= C_1 \, |G.\text{relations} \cap S.\text{relations}|$
7: $\quad score \mathrel{-}= C_2 \, |G.\text{conds} - S.\text{conds}|$
8: $\quad score \mathrel{-}= C_3 \, |G.\text{entities} \ominus S.\text{entities}|$
9: $\quad score \mathrel{-}= C_4 \, |S.\text{sentence}|$
10: $\quad score \mathrel{/}= C_5 \, |B|$
11: **end if**
12: **return** $score$

Algorithm 5 getValidBindings (Alg. 4, line 2)

Require: Partial Sentence S, World W

1: $validBindings \leftarrow \emptyset$
2: $queue \leftarrow prevBindings$ **if exists else** $[\emptyset]$
3: **while** $|queue| > 0$ **do**
4: $\quad b \leftarrow queue$.pop()
5: $\quad S' \leftarrow$ apply binding b to S
6: $\quad$ **if** S', W consistent **and** S'.entities all bound **then**
7: $\quad\quad validBindings$.append(b)
8: $\quad$ **else if** S', W consistent **then**
9: $\quad\quad freeS \leftarrow$ unbound S'.entities
10: $\quad\quad freeW \leftarrow W$.entities not in b
11: $\quad\quad$ **for** $e_s, e_w \in freeS \times freeW$ **do**
12: $\quad\quad\quad queue$.push($b \cup \{e_s \rightarrow e_w\}$)
13: $\quad\quad$ **end for**
14: $\quad$ **end if**
15: **end while**
16: **return** $validBindings$

Instead of trying every binding, we use the procedure shown in Algorithm 5 to greatly reduce the number of bindings we must check. Starting with an initially empty binding, we repeatedly add a single $\{sentenceEntity \rightarrow worldEntity\}$ pair (line 12). If a binding contains all partial sentence

entities and the semantics are consistent with the world, the binding is valid (lines 6-7). If at any point, a binding yields partial sentence semantics that are *inconsistent* with the world, we no longer need to consider any bindings which it is a subset of (when condition on line 8 is false, no children expanded). The benefit of this bottom-up approach is that when an inconsistency is caused by adding a mapping of partial sentence entity e_1 and world entity e_2, *all* of the $\binom{N-1}{K-1}$ bindings containing $\{e_1 \rightarrow e_2\}$ are ruled out as well. This procedure is especially effective in worlds/goals with low ambiguity (such as real-world text).

We further note that many of the binding checks are repeated between action selections. Because our sentence semantics are conjunctive, entity specifications only get *more* specific with additional relations; therefore, bindings that were invalidated earlier in the search procedure can never again become valid. Thus, we can cache and reuse valid bindings from the previous partial sentence (line 2). For domains with very large worlds (where most relations have no bearing on the communicative goal), most of the possible bindings will be ruled out with the first few action applications, resulting in large computational savings.

3.3 Reusing the Search Tree

The STRUCT algorithm constructs a lookahead tree of depth D via policy rollout to estimate the value of each action. This tree is then discarded and the procedure repeated at the next state. But it may be that at the next state, many of the *useful* actions will already have been visited by prior iterations of the algorithm. For a lookahead depth D, some actions will have already been explored up to depth $D-1$.

For example if we have generated the partial sentence "the cat chased the rabbit" and S-STRUCT looks ahead to find that a greater reward is possible by introducing the relative clause "the rabbit that ate", when we transition to "the rabbit that", we do not need to re-explore "ate" and can directly try actions that result in "that ate grass", "that ate carrots", etc. Note that if there are still unexplored actions at an earlier depth, these will still be explored as well (action rollouts such as "that drank water" in this example).

Reusing the search tree is especially effective given that the tree policy causes us to favor areas of the search space with high value. Therefore, when we transition to the state with highest value, it is likely that many useful actions have already been explored. Reusing the search tree is reflected in Algorithms 1-2 by passing $uctTree$ back and forth to/from $getAction$ instead of starting a new search tree at each step. In $applyAction$, when a state/action already in the tree is chosen, S-STRUCT transitions to the next state without having to recompute the state or its reward.

3.4 Learning and Using Search Control

During the search procedure, a large number of actions are explored but relatively few of them are helpful. Ideally, we would know which actions would lead to valuable states without actually having to expand and evaluate the resultant states, which is an expensive operation. From prior knowledge, we know that if we have a partial sentence of "the sky is", we should try actions resulting in "the sky is blue" before those resulting in "the sky is yellow". This prior knowledge can be estimated through *learned* heuristics from previous runs of the planner (Yoon et al., 2008). To do this, a set of previously completed plans can be treated as a training set: for each (state, action) pair considered, a feature vector $\Phi(s,a)$ is emitted, along with either the distance to the goal state or a binary indicator of whether or not the state is on the path to the goal. A perceptron (or similar model) $H(s,a)$ is trained on the $(\Phi(s,a), target)$ pairs. $H(s,a)$ can be incorporated into the planning process to help guide future searches.

We apply this idea to our S-STRUCT system by tracking the (state, action) pairs visited in previous runs of the STRUCT system where STRUCT obtained at least 90% of the reward of the known best sentence and emit a feature vector for each, containing: global tree frequency, tree probability (as defined in Section 4.1), and the word correlation of the action's anchor with the two words on either side of the action location. We define the global tree frequency as the number of times the tree appeared in the corpus normalized by the number of trees in the corpus; this is different than the tree probability as it does not take any context into account (such as the parent tree and substitution location). Upon search completion, the feature vectors are annotated with a binary indicator label of whether or not the (state, action) pair was on the path to the best sentence. This training set is then used to train a perceptron $H(s,a)$.

Table 1: Summary statistics for test data sets

Test Set	Goals / Sentences	Vocab Size	Lex Trees / Actions	World Entities	World Relations	Avg. Goal Entities	Avg. Goal Relations	Max Depth
Small	50	130	395	77	135	1.54	2.70	0
Medium	500	1165	3734	741	1418	1.48	2.83	1
Large	5000	9872	31966	10998	23097	2.20	4.62	6

We use $H(s, a)$ to inform both the open action policy (Algorithm 2, line 4) and the tree policy (Algorithm 2, line 6). In the open action policy , we choose open actions according to their heuristic values, instead of just their tree probabilities. In the tree policy, we incorporate $H(s, a)$ into the reward estimation by using Equation 2 in place of Equation 1 in Algorithm 2 (Chaslot et al., 2008a):

$$P(s,a) = Q(s,a) + \lambda H(s,a) + c\sqrt{\frac{lnN(s)}{N(s,a)}}. \quad (2)$$

Here, $H(s, a)$ is a value prediction from prior knowledge and λ is a parameter controlling the trade-off between prior knowledge and estimated value on this goal.

4 Empirical Evaluation

In this section, we evaluate three hypotheses: (1) S-STRUCT can handle real-world datasets, as they scale in terms of (a) grammar size, (b) world size, (c) entities/relations in the goal, (d) lookahead required to generate sentences, (2) S-STRUCT scales better than STRUCT to such datasets and (3) Each of the enhancements above provides a positive contribution to STRUCT's scalability in isolation.

4.1 Datasets

We collected data in the form of grammars, worlds and goals for our experiments, starting from the WSJ corpus of the Penn TreeBank (Marcus et al., 1993). We parsed this with an LTAG parser to generate the best parse and derivation tree (Sarkar, 2000; XTAG Research Group, 2001). The parser generated valid parses for 18,159 of the WSJ sentences. To pick the *best* parse for a given sentence, we choose the parse which minimizes the PARSEVAL bracket-crossing metric against the gold-standard (Abney et al., 1991). This ensures that the major structures of the parse tree are retained. We then pick the 31 most frequently occurring XTAG trees (giving us 74% coverage of the parsed sentences) and annotate them with compositional semantics. The final result of this process was a corpus of semantically annotated WSJ sentences along with their parse and derivation trees [1].

To show the scalability of the improved STRUCT system, we extracted 3 datasets of increasing size and complexity from the semantically annotated WSJ corpus. We nominally refer to these datasets as Small, Medium, and Large. Summary statistics of the data sets are shown in Table 1. For each test set, we take the grammar to be all possible lexicalizations of the unlexicalized trees given the anchors of the test set. We set the world as the union of all communicative goals in the test set. The PLTAG probabilities are derived from the entire parseable portion of the WSJ. Due to the data sparsity issues (Bauer and Koller, 2010), we use unlexicalized probabilities.

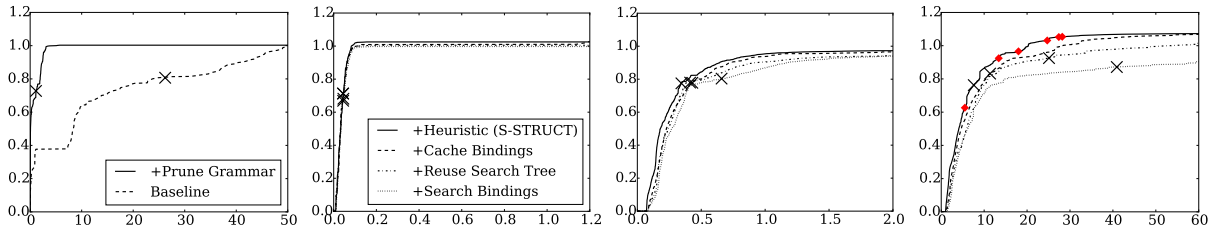
The reward function constants $\boldsymbol{C}$ were set to $[500, 100, 10, 10, 1]$. In the tree policy, c was set to 0.5. These are as in the original STRUCT system. λ was chosen as 100 after evaluating $\{0, 10, 100, 1000, 10000\}$ on a tuning set.

In addition to test sets, we extract an independent training set using 100 goals to learn the heuristic $H(s, a)$. We train a separate perceptron for each test set and incorporate this into the S-STRUCT algorithm as described in Section 3.4.

4.2 Results

For these experiments, S-STRUCT was implemented in Python 3.4. The experiments were run on a single core of a Intel(R) Xeon(R) CPU E5-2450 v2 processor clocked at 2.50GHz with access to 8GB of RAM. The times reported are from the start of the generation process instead of the start of the program execution to reduce variation caused by interpreter startup, input parsing, etc. In

[1] Not all of the covered trees were able to recursively derive their semantics, despite every constituent tree being semantically annotated. This is because β-reduction of the λ-semantics is not associative in many cases where the syntactic composition is associative, causing errors during semantic unification. Due to this and other issues, the number of usable parse trees/sentences was about 7500.



(a) Small Baseline (b) Small (c) Medium (d) Large

Figure 2: Avg. Best Normalized Reward (y-axis) vs. Time in Seconds (x-axis) for (a) Small Baseline, (b) Small, (c) Medium, (d) Large. Time when first grammatical sentence available marked as $\times$. Experiments are cumulative (a trial contains all improvements below it in the legend).

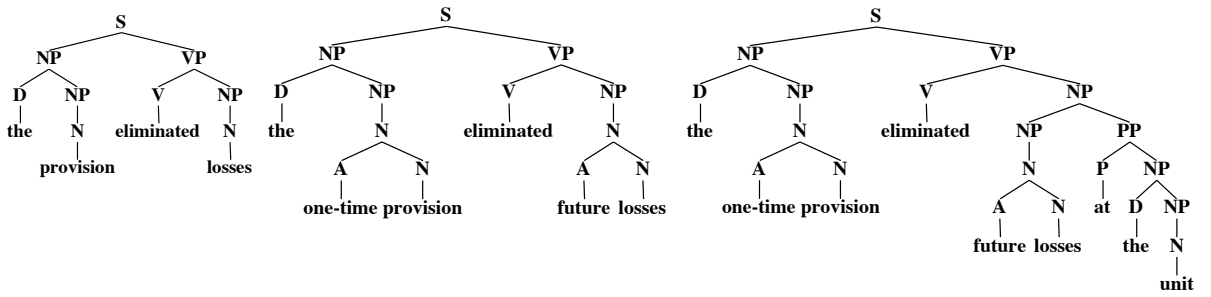


Figure 3: Best sentence available during S-STRUCT generation at 5.5 (s), 18.0 (s), and 28.2 (s)

all experiments, we normalize the reward of a sentence by the reward of the actual parse tree, which we take to be the gold standard. Note that this means that in some cases, S-STRUCT can produce solutions with *better* than this value, e.g. if there are multiple ways to achieve the semantic goal.

To investigate the first two hypotheses that S-STRUCT can handle the scale of real-world datasets and scales better than STRUCT, we plot the average best reward of all goals in the test set over time in Figure 2. The results show the cumulative effect of the enhancements; working up through the legend, each line represents "switching on" another option and includes the effects of all improvements listed below it. The addition of the heuristic represents the entire S-STRUCT system. On each line, $\times$ marks the time at which the first grammatically correct sentence was available.

The Baseline shown in Figure 2a is the original STRUCT system proposed in (McKinley and Ray, 2014). Due to the large number of actions that must be considered, the Baseline experiment's average first sentence is not available until 26.20 seconds, even on the Small dataset. In previous work, the experiments for both STRUCT and CRISP were on toy examples, with grammars having 6 unlexicalized trees and typically < 100 lexicalized trees (McKinley and Ray, 2014; Koller and Stone, 2007). In these experiments, STRUCT was shown to perform better than or as well as CRISP. Even in our smallest domain, however, the baseline STRUCT system is impractically slow. Further, prior work on PCRISP used a grammar that was extracted from the WSJ Penn TreeBank, however it was restricted to the 416 sentences in Section 0 with $<$16 words. With PCRISP's extracted grammar, the most successful realization experiment yielded a sentence in only 62% of the trials, the remainder having timed out after five minutes (Bauer and Koller, 2010). Thus it is clear that these systems do not scale to real NLG tasks.

Adding the grammar pruning to the Baseline allows S-STRUCT to find the first grammatically correct sentence in 1.3 seconds, even if the reward is still sub-optimal. For data sets larger than Small, the Baseline and Prune Grammar experiments could not be completed, as they still enumerated all semantic bindings. For even the medium world, a sentence with 4 entities would have to consider 1.2×10^{10} bindings. Therefore, the cumulative experiments start with Prune Grammar and Search Bindings turned on.

Figures 2b, 2c and 2d show the results for each enhancement above on the corresponding dataset. We observe that the improved binding search further improves performance on the Small task. The Small test set does not require any lookahead, so it is expected that there would be no benefit to

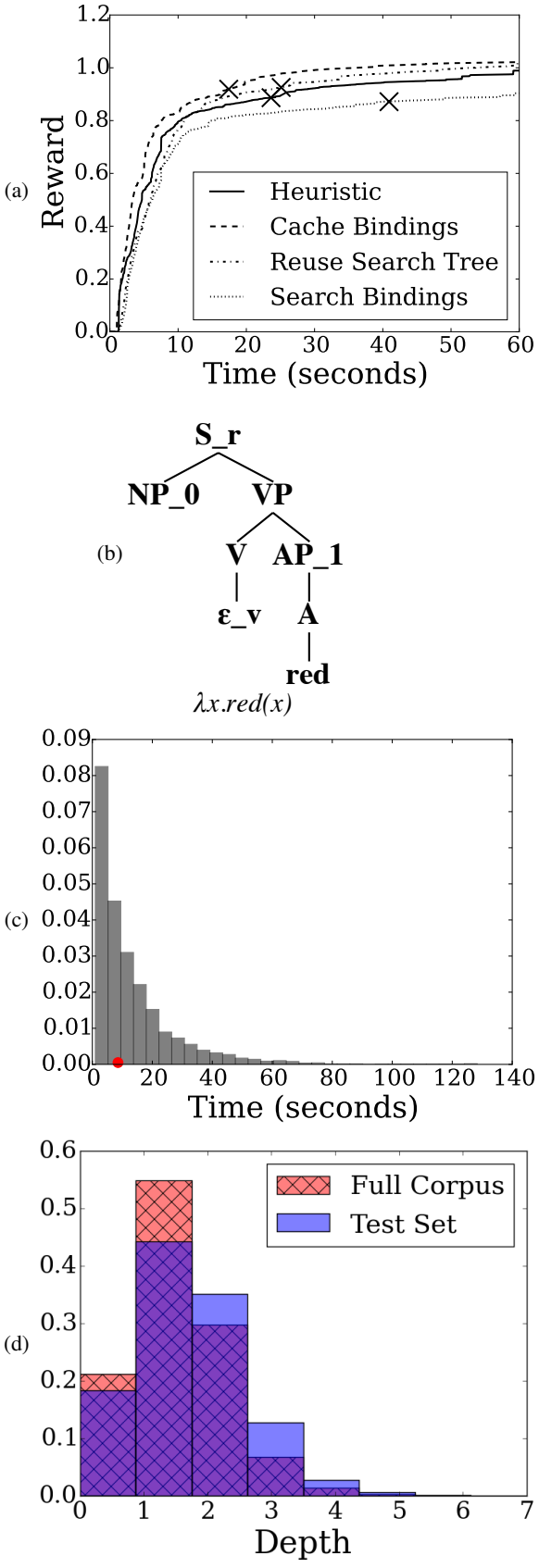


Figure 4: (a) Large Non-Cumulative Experiment (b) $\alpha nx0Ax1$ XTAG tree (c) Time to 90% Reward (d) Lookahead Required.

reusing the search tree, and little to no benefit from caching bindings or using a heuristic. In the Small domain, S-STRUCT is able to generate sentences very quickly; the first sentence is available by 44ms and the best sentence is available by 100ms. In the medium and large domains, the "Reuse Search Tree", "Cache Bindings", and "Heuristic" changes *do* improve upon the use of only "Search Bindings". The Medium domain is still extremely fast, with the first sentence available in 344ms and the best sentence available around 1s. The large domain slows down due to the larger lookahead required, the larger grammar, and the huge number of bindings that have to be considered. Even with this, S-STRUCT can generate a first sentence in 7.5s and the best sentence in 25s. In Figure 4c, we show a histogram of the generation time to 90% of the best reward. The median time is 8.55s (● symbol).

Additionally, histograms of the lookahead required for guaranteed optimal generation are shown for the entire parsable WSJ and our Large world in Figure 4d. The complexity of the entire WSJ does not exceed our Large world, thus we argue that our results are representative of S-STRUCT's performance on real-world tasks.

To investigate the third hypothesis that each improvement contributes positively to the scalability, the noncumulative impact of each improvement is shown in Figure 4a. All experiments still must have Prune Grammar and Search Bindings turned on in order to terminate. Therefore, we take this as a baseline to show that the other changes provide additional benefits. Looking at Figure 4a, we see that each of the changes improves the reward curve and the time to generate the first sentence.

4.3 Discussion, Limitations and Future Work

As an example of sentences available at a given time in the process, we annotate the Large Cumulative Heuristic Experiment with ♦ symbols for a specific trial of the Large dataset. Figure 3 shows the best sentence that was available at three different times. The first grammatically correct sentence was available 5.5 seconds into the generation process, reading "The provision eliminated losses". This sentence captured the major idea of the communicative goal, but missed some critical details. As the search procedure continued, S-STRUCT explored adjunction actions. By 18 seconds, additional semantic content was added to expand upon

the details of the provision and losses. S-STRUCT settled on the best sentence it could find at 28.2 seconds, able to match the entire communicative goal with the sentence "The one-time provision eliminated future losses at the unit".

In domains with large lookaheads required, reusing the Search Tree has a large effect on both the best reward at a given time and on the time to generate the first sentence. This is because S-STRUCT has already explored some actions from depth 1 to $D-1$. Additionally, in domains with a large world, the Cache Binding improvement is significant. The learned heuristic, which achieves the best reward and the shortest time to a complete sentence, tries to make S-STRUCT choose better actions at each step instead of allowing STRUCT to explore actions faster; this means that there is less overlap between the improvement of the heuristic and other strategies, allowing the *total* improvement to be higher.

One strength of the heuristic is in helping S-STRUCT to avoid "bad" actions. For example, the XTAG tree $\alpha nx0Ax1$ shown in Figure 4b is an initial tree lexicalized by an adjective. This tree would be used to say something like "The dog is red." S-STRUCT may choose this as an initial action to fulfill a subgoal; however, if the goal was to say that a red dog chased a cat, S-STRUCT will be shoehorned into a substantially worse goal down the line, when it can no longer use an initial tree that adds the "chase" semantics. Although the rollout process helps, some sentences can share the same reward up to the lookahead and only diverge later. The heuristic can help by biasing the search against such troublesome scenarios.

All of the results discussed above are without parallelization and other engineering optimizations (such as writing S-STRUCT in C), as it would make for an unfair comparison with the original system. The core UCT procedure used by STRUCT and S-STRUCT could easily be parallelized, as the sampling shown in Algorithm 2 can be done independently. This has been done in other domains in which UCT is used (Computer Go), to achieve a speedup factor of 14.9 using 16 processor threads (Chaslot et al., 2008b). Therefore, we believe these optimizations would result in a constant factor speedup.

Currently, the STRUCT and S-STRUCT systems only focuses on the domain of single sentence generation, rather than discourse-level planning. Additionally, neither system handles non-semantic feature unification, such as constraints on number, tense, or gender. While these represent practical concerns for a production system, we argue that their presence will not affect the system's scalability, as there is already feature unification happening in the λ-semantics. In fact, we believe that additional features could *improve* the scalability, as many available actions will be ruled out at each state.

5 Conclusion

In this paper we have presented S-STRUCT, which enhances the STRUCT system to enable better scaling to real generation tasks. We show via experiments that this system can scale to large worlds and generate complete sentences in real-world datasets with a median time of 8.5s. To our knowledge, these results and the scale of these NLG experiments (in terms of grammar size, world size, and lookahead complexity) represents the state-of-the-art for planning-based NLG systems. We conjecture that the parallelization of S-STRUCT could achieve the response times necessary for real-time applications such as dialog. S-STRUCT is available through Github upon request.

References

S. Abney, S. Flickenger, C. Gdaniec, C. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, M. Liberman, M. Marcus, S. Roukos, B. Santorini, and T. Strzalkowski. 1991. Procedure for quantitatively comparing the syntactic coverage of english grammars. In E. Black, editor, *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, pages 306–311, Stroudsburg, PA, USA. Association for Computational Linguistics.

D. Bauer and A. Koller. 2010. Sentence generation as planning with probabilistic LTAG. *Proceedings of the 10th International Workshop on Tree Adjoining Grammar and Related Formalisms, New Haven, CT*.

A.L. Blum and M.L. Furst. 1997. Fast planning through planning graph analysis. *Artificial intelligence*, 90(1):281–300.

Bernd Bohnet and Robert Dale. 2005. Viewing referring expression generation as search. In *International Joint Conference on Artificial Intelligence*, pages 1004–1009.

Guillaume M. JB Chaslot, Mark H.M. Winands, H. Jaap van Den Herik, Jos W.H.M. Uiterwijk, and Bruno Bouzy. 2008a. Progressive strategies for

monte-carlo tree search. *New Mathematics and Natural Computation*, 4(03):343–357.

Guillaume M. JB Chaslot, Mark H.M. Winands, and H Jaap van Den Herik. 2008b. Parallel monte-carlo tree search. In *Computers and Games*, pages 60–71. Springer.

M. Fox and D. Long. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124.

Aravind K Joshi and Yves Schabes. 1997. Tree-adjoining grammars. In *Handbook of formal languages*, pages 69–123. Springer.

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, pages 200–204, Stroudsburg, PA, USA. Association for Computational Linguistics.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg. Springer-Verlag.

Alexander Koller and Matthew Stone. 2007. Sentence generation as a planning problem. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 336–343, Prague, Czech Republic, June. Association for Computational Linguistics.

I. Langkilde-Geary. 2002. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the 12th International Natural Language Generation Workshop*, pages 17–24. Citeseer.

W. Lu, H.T. Ng, and W.S. Lee. 2009. Natural language generation with tree conditional random fields. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1*, pages 400–409. Association for Computational Linguistics.

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

Nathan McKinley and Soumya Ray. 2014. A decision-theoretic approach to natural language generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 552–561, Baltimore, Maryland, June. Association for Computational Linguistics.

M.L. Puterman. 1994. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Ehud Reiter and Robert Dale. 1997. Building applied natural language generation systems. *Natural Language Engineering*, 3(1):57–87.

Anoop Sarkar. 2000. Practical experiments in parsing using tree adjoining grammars. In *Proceedings of TAG*, volume 5, pages 25–27.

Stuart M. Shieber. 1988. A uniform architecture for parsing and generation. In *Proceedings of the 12th conference on Computational linguistics - Volume 2*, COLING '88, pages 614–619, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Dongho Kim, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 275–284, Prague, Czech Republic, September. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, September. Association for Computational Linguistics.

M. White and J. Baldridge. 2003. Adapting chart realization to CCG. In *Proceedings of the 9th European Workshop on Natural Language Generation*, pages 119–126.

XTAG Research Group. 2001. A lexicalized tree adjoining grammar for english. Technical Report IRCS-01-03, IRCS, University of Pennsylvania.

Sungwook Yoon, Alan Fern, and Robert Givan. 2008. Learning control knowledge for forward search planning. *The Journal of Machine Learning Research*, 9:683–718.

ALTO: Active Learning with Topic Overviews for Speeding Label Induction and Document Labeling

Forough Poursabzi-Sangdeh
Computer Science
University of Colorado
forough.poursabzisangdeh@colorado.edu

Jordan Boyd-Graber
Computer Science
University of Colorado
Jordan.Boyd.Graber@colorado.edu

Leah Findlater
iSchool and UMIACS
University of Maryland
leahkf@umd.edu

Kevin Seppi
Computer Science
Brigham Young University
kseppi@cs.byu.edu

Abstract

Effective text classification requires experts to annotate data with labels; these training data are time-consuming and expensive to obtain. If you know what labels you want, active learning can reduce the number of labeled documents needed. However, establishing the label set remains difficult. Annotators often lack the global knowledge needed to induce a label set. We introduce ALTO: Active Learning with Topic Overviews, an interactive system to help humans annotate documents: topic models provide a global overview of what labels to create and active learning directs them to the right documents to label. Our forty-annotator user study shows that while active learning alone is best in extremely resource limited conditions, topic models (even by themselves) lead to better label sets, and ALTO's combination is best overall.

1 Introduction

Many fields depend on texts labeled by human experts; computational linguistics uses such annotation to determine word senses and sentiment (Kelly and Stone, 1975; Kim and Hovy, 2004); while social science uses "coding" to scale up and systemetize content analysis (Budge, 2001; Klingemann et al., 2006).

Classification takes these labeled data as a training set and labels new data automatically. Creating a broadly applicable and consistent label set that generalizes well is time-consuming and difficult, requiring expensive annotators to examine large swaths of the data. Effective NLP systems must measure (Hwa, 2004; Osborne and Baldridge, 2004; Ngai and Yarowsky, 2000) and reduce annotation cost (Tomanek et al., 2007). Annotation is hard because it requires both *global* and *local* knowledge of the entire dataset. Global knowledge is required to create the set of labels, and local knowledge is required to annotate the most useful examples to serve as a training set for an automatic classifier. The former's cost is often hidden in multiple rounds of refining annotation guidelines.

We create a single interface—ALTO (Active Learning with Topic Overviews)—to address both global and local challenges using two machine learning tools: *topic models* and *active learning* (we review both in Section 2). Topic models address the need for annotators to have a *global overview* of the data, exposing the broad themes of the corpus so annotators know what labels to create. Active learning *selects* documents that help the classifier understand the differences between labels and directs the user's attention *locally* to them. We provide users four experimental conditions to compare the usefulness of a topic model or a simple list of documents, with or without active learning suggestions (Section 3). We then describe our data and evaluation metrics (Section 4).

Through both synthetic experiments (Section 5) and a user study (Section 6) with forty participants, we evaluate ALTO and its constituent components by comparing results from the four conditions introduced above. We first examine user strategies for organizing documents, user satisfaction, and user efficiency. Finally, we evaluate the overall effectiveness of the label set in a post study crowdsourced task.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1158–1169,
Berlin, Germany, August 7-12, 2016.

Topic words	Document Title
metropolitan, carrier, rail, freight, passenger, driver, airport, traffic, transit, vehicles	A bill to improve the safety of motorcoaches, and for other purposes.
violence, sexual, criminal, assault, offense, victims, domestic, crime, abuse, trafficking	A bill to provide criminal penalties for stalking.
agricultural, farm, agriculture, rural, producer, dairy, crop, producers, commodity, nutrition	To amend the Federal Crop Insurance Act to extend certain supplemental agricultural disaster assistance programs through fiscal year 2017, and for other purposes.

Table 1: Given a dataset—in this case, the US congressional bills dataset—topics are automatically discovered sorted lists of terms that summarize segments of a document collection. Topics also are associated with documents. These topics give users a sense of documents' main themes and help users create high-quality labels.

2 Topic Overviews and Active Learning

ALTO,[1] a framework for assigning labels to documents that uses both global and local knowledge to help users create and assign document labels, has two main components: topic *overview* and active learning *selection*. We explain how ALTO uses topic models and active learning to aid label induction and document labeling.

Topic Models Topic models (Blei et al., 2003) automatically induce structure from a text corpus. Given a corpus and a constant K for the number of topics, topic models output (i) a distribution over words for each topic k ($\phi_{k,w}$) and (ii) a distribution over topics for each document ($\theta_{d,k}$). Each topic's most probable words and associated documents can help a user understand what the collection is about. Table 1 shows examples of topics and their highest associated documents from our corpus of US congressional bills.

Our hypothesis is that showing documents grouped by topics will be more effective than having the user wade through an undifferentiated list of random documents and *mentally sort the major themes themselves*.

Active Learning Active learning (Settles, 2012) directs users' attention to the examples that would be most useful to label when training a classifier. When user time is scarce, active learning builds a more effective training set than random labeling: uncertainty sampling (Lewis and Gale, 1994) or query by committee (Seung et al., 1992) direct users to the most useful documents to label.

In contrast to topic models, active learning provides local information: *this document* is the one you should pay attention to. Our hypothesis is that active learning directing users to documents most beneficial to label will not only be more effective than randomly selecting documents but will also *complement* the global information provided by topic models. Section 3.3 describes our approaches for directing user's local attention.

3 Study Conditions

Our goal is to characterize how local and global knowledge can aid users in annotating a dataset. This section describes our four experimental conditions and outlines the user's process for labeling documents.

3.1 Study Design

The study uses a 2×2 between-subjects design, with factors of document collection *overview* (two levels: topic model or list) and document *selection* (two levels: active or random). The four conditions, with the TA condition representing ALTO, are:

1. Topic model overview, active selection (TA)
2. Topic model overview, random selection (TR)
3. List overview, active selection (LA)
4. List overview, random selection (LR)

3.2 Document Collection Overview

The topic and list overviews offer different overall structure but the same basic elements for users to create, modify, and apply labels (Section 3.4). The topic overview (Figure 1a) builds on Hu et al. (2014): for each topic, the top twenty words are shown alongside twenty document titles. Topic words (w) are sized based on their probability $\phi_{k,w}$ in the topic k and the documents with the highest probability of that topic ($\theta_{d,k}$) are shown. The list overview, in contrast, presents documents as a simple, randomly ordered list of titles (Figure 1b). We display the same number of documents ($20K$, where K is the total number of topics) in both the topic model and list overviews, but the list overview provides no topic information.

[1]Code available at https://github.com/Foroughp/ALTO-ACL-2016

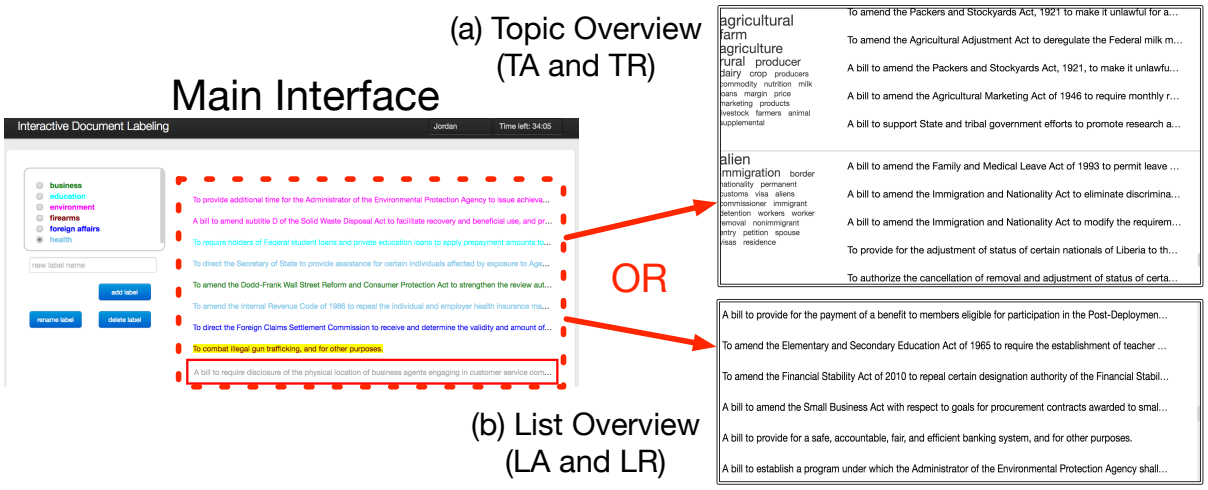


Figure 1: Our annotation system. Initially, the user sees lists of documents organized in either a list format or grouped into topics (only two topics are shown here; users can scroll to additional documents). The user can click on a document to label it.

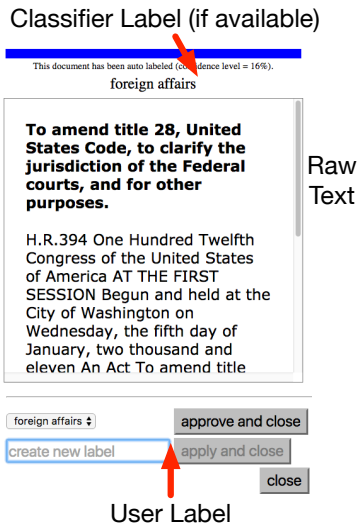


Figure 2: After clicking on a document from the list or topic overview, the user inspects the text and provides a label. If the classifier has a guess at the label, the user can confirm the guess.

3.3 Document Selection

We use a *preference function* U to direct users' attention to specific documents. To provide consistency across the four conditions, each condition will highlight the document that scores the highest for the condition's preference function. For the random selection conditions, TR and LR, document selection is random, within a topic or globally. We expect this to be less useful than active learning. The document preference functions are:

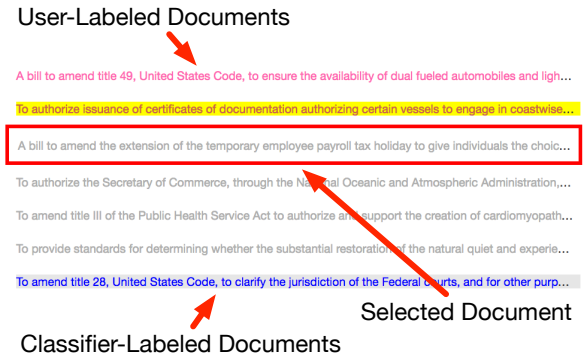


Figure 3: After the user has labeled some documents, the system can automatically label other documents and select which documents would be most helpful to annotate next. In the random selection setting, random documents are selected.

LA: LA uses traditional uncertainty sampling:

$$U_d^{\text{LA}} = \mathbb{H}_C\left[Y_d\right], \tag{1}$$

where $\mathbb{H}_C\left[y_d\right] = -\sum_i P(y_i|d) log P(y_i|d)$ is the classifier entropy. Entropy measures how confused (uncertain) classifier C is about its prediction of a document d's label y. Intuitively, it prefers documents the classifier suggests many labels instead of a single, confident prediction.

LR: LR's approach is the same as LA's except we replace $\mathbb{H}_C\left[y_d\right]$ with a uniform random number:

$$U_d^{\text{LR}} \sim \text{unif}(0, 1). \tag{2}$$

In contrast to LA, which suggests the most uncertain document, LR suggests a random document.

TA: Dasgupta and Hsu (2008) argue that clustering should inform active learning criteria, balancing coverage against classifier accuracy. We adapt their method to flat topic models—in contrast to their hierarchical cluster trees—by creating a composite measure of document uncertainty within a topic:

$$U_d^{\text{TA}} = \mathbb{H}_C\left[y_d\right]\theta_{d,k}, \tag{3}$$

where k is the prominent topic for document d. U_d^{TA} prefers documents that are *representative* of a topic (i.e., have a high value of $\theta_{d,k}$ for that topic) and are informative for the classifier.

TR: TR's approach is the same as TA's except we replace $\mathbb{H}_C\left[Y_d\right]$ with a uniformly random number:

$$U_d^{\text{TR}} = \text{unif}(0,1)\theta_{d,k}. \tag{4}$$

Similar to TA, U_d^{TR} prefers documents that are representative of a topic, but not any particular document in the topic. Incorporating the random component encourages covering different documents in diverse topics.

In LA and LR, the preference function directly chooses a document and directs the user to it. On the other hand, U_d^{TA} and U_d^{TR} are topic dependent. TA emphasizes documents that are both informative to the classifier and representative of a topic; if a document is not representative, the surrounding context of a topic will be less useful. Therefore, the factor $\theta_{d,k}$ appears in both. Thus, they require that a topic be chosen first and then the document with maximum preference, U, within that topic can be chosen. In TR, the topic is chosen randomly. In TA, the topic is chosen by

$$k^* = \arg\max_k(\text{median}_d(\mathbb{H}_C\left[y_d\right]\theta_{d,k}). \tag{5}$$

That is the topic with the maximum median U. Median encodes how "confusing" a topic is.[2] In other words, topic k^* is the topic that its documents confuse the classifier most.

3.4 User Labeling Process

The user's labeling process is the same in all four conditions. The *overview* (topic or list) allows users to examine individual documents (Figure 1). Clicking on a document opens a dialog box (Figure 2) with the text of the document and three options:

1. Create and assign a new label to the document.
2. Choose an existing label for the document.
3. Skip the document.

Once the user has labeled two documents with different labels, the displayed documents are replaced based on the preference function (Section 3.3), every time the user labels (or updates labels for) a document. In TA and TR, each topic's documents are replaced with the twenty highest ranked documents. In LA and LR, all documents are updated with the top $20K$ ranked documents.[3]

The system also suggests one document to consider by auto-scrolling to it and drawing a red box around its title (Figure 3). The user may ignore that document and click on any other document. After the user labels ten documents, the classifier runs and assigns labels to other documents.[4] For classifier-labeled documents, the user can either approve the label or assign a different label. The process continues until the user is satisfied or a time runs out (forty minutes in our user study, Section 6). We use time to control for the varying difficulty of assigning document labels: active learning will select more difficult documents to annotate, but they may be more useful; time is a more fair basis of comparison in real-world tasks.

4 Data and Evaluation Metrics

In this section, we describe our data, the machine learning techniques to learn classifiers from examples, and the evaluation metrics to know whether the final labeling of the complete documents collection was successful.

4.1 Datasets

Data Our experiments require corpora to compare user labels with gold standard labels. We experiment with two corpora: 20Newsgroups (Lang, 2007) and US congressional bills from GovTrack.[5]

For US congressional bills, GovTrack provides bill information such as the title and text, while the Congressional Bills Project (Adler and Wilkerson, 2006) provides labels and sub-labels for the bills. Examples of labels are agriculture and health, while sub-labels include agricultural trade and comprehensive health care reform. The twenty

[2] Outliers skew other measures (e.g., max or mean).

[3] In all conditions, the number of displayed unlabeled documents is adjusted based on the number of manually labeled documents. i.e. if the user has labeled n documents in topic k, n manually labeled documents followed by top $20 - n$ uncertain documents will be shown in topic k.

[4] To reduce user confusion, for each existing label, only the top 100 documents get a label assigned in the UI.

[5] `https://www.govtrack.us/`

top-level labels have been developed by consensus over many years by a team of top political scientists to create a reliable, robust dataset. We use the 112th Congress; after filtering,[6] this dataset has 5558 documents. We use this dataset in both the synthetic experiments (Section 5) and the user study (Section 6).

The 20 Newsgroups corpus has 19,997 documents grouped in twenty news groups that are further grouped into six more general topics. Examples are talk.politics.guns and sci.electronics, which belong to the general topics of politics and science. We use this dataset in synthetic experiments (Section 5).

4.2 Machine Learning Techniques

Topic Modeling To choose the number of topics (K), we calculate average topic coherence (Lau et al., 2014) on US Congressional Bills, between ten and forty topics and choose $K = 19$, as it has the maximum coherence score. For consistency, we use the same number of topics ($K = 19$) for 20 Newsgroups corpus. After filtering words based on TF-IDF, we use Mallet (McCallum, 2002) with default options to learn topics.

Features and Classification A logistic regression predicts labels for documents and provides the classification uncertainty for active learning. To make classification and active learning updates efficient, we use incremental learning (Carpenter, 2008, LingPipe). We update classification parameters using stochastic gradient descent, restarting with the previously learned parameters as new labeled documents become available.[7] We use cross validation, using argmax topics as surrogate labels, to set the parameters for learning the classifier.[8]

The features for classification include topic probabilities, unigrams, and the fraction of labeled documents in each document's prominent topic. The intuition behind adding this last feature is to allow active learning to suggest documents in a diverse range of topics if it finds this feature a useful indicator of uncertainty.[9]

4.3 Evaluation Metrics

Our goal is to create a system that allows users to quickly induce a high-quality label set. We compare the user-created label sets against the data's gold label sets. Comparing different clusterings is a difficult task, so we use three clustering evaluation metrics: purity (Zhao and Karypis, 2001), rand index (Rand, 1971, RI), and normalized mutual information (Strehl and Ghosh, 2003, NMI).[10]

Purity The documents labeled with a good user label should only have one (or a few) gold labels associated with them: this is measured by cluster purity. Given each user cluster, it measures what fraction of the documents in a user cluster belong to the most frequent gold label in that cluster:

$$\text{purity}(\mathbf{\Omega}, \mathbf{G}) = \frac{1}{N} \sum_l \max_j |\Omega_l \cap G_j|, \quad (6)$$

where L is the number of labels user creates, $\mathbf{\Omega} = \{\Omega_1, \Omega_2, \ldots, \Omega_L\}$ is the user clustering of documents, $\mathbf{G} = \{G_1, G_2, \ldots, G_J\}$ is gold clustering of documents, and N is the total number of documents. The user Ω_l and gold G_j labels are interpreted as sets containing all documents assigned to that label.

Rand index (RI) RI is a *pair counting* measure, where cluster evaluation is considered as a series of decisions. If two documents have the same gold label and the same user label (TP) or if they do not have the same gold label and are not assigned the same user label (TN), the decision is right. Otherwise, it is wrong (FP, FN). RI measures the percentage of decisions that are right:

$$\text{RI} = \frac{TP + TN}{TP + FP + TN + FN}. \quad (7)$$

Normalized mutual information (NMI) NMI is an *information theoretic* measure that measures the amount of information one gets about the gold clusters by knowing what the user clusters are:

$$\text{NMI}(\mathbf{\Omega}, \mathbf{G}) = \frac{2\mathbb{I}(\mathbf{\Omega}, \mathbf{G})}{\mathbb{H}_{\mathbf{\Omega}} + \mathbb{H}_{\mathbf{G}}}, \quad (8)$$

[6] We remove bills that have less than fifty words, no assigned gold label, duplicate titles, or have the gold label GOVERNMENT OPERATIONS or SOCIAL WELFARE, which are broad and difficult for users to label.

[7] Exceptions are when a new label is added, a document's label is deleted, or a label is deleted. In those cases, we train the classifier from scratch. Also, for final results in Section 6, we train a classifier from scratch.

[8] We use `blockSize=`$\frac{1}{\#\text{examples}}$ `minEpochs`=100, `learningRate`=0.1, `minImprovement`=0.01, `maxEpochs`=1000, and `rollingAverageSize`=5. The regression is unregularized.

[9] However, final classifier's coefficients suggested that this feature did not have a large effect.

[10] We avoided using adjusted rand index (Hubert and Arabie, 1985), because it can yield negative values, which is not consistent with purity and NMI. We also computed variation of information (Meilă, 2003) and normalized information distance (Vitányi et al., 2009) and observed consistent trends. We omit these results for the sake of space.

where $\mathbf{\Omega}$ and $\mathbf{G}$ are user and gold clusters, $\mathbb{H}$ is the entropy and $\mathbb{I}$ is mutual information (Bouma, 2009).

While purity, RI, and NMI are all normalized within $[0, 1]$ (higher is better), they measure different things. Purity measures the intersection between two clusterings, it is sensitive to the number of clusters, and it is not symmetric.

On the other hand, RI and NMI are less sensitive to the number of clusters and are symmetric. RI measures pairwise agreement in contrast to purity's emphasis on intersection. Moreover, NMI measures shared information between two clusterings.

None of these metrics are perfect: purity can be exploited by putting each document in its own label, RI does not distinguish separating similar documents with distinct labels from giving dissimilar documents the same label, and NMI's ability to compare different numbers of clusters means that it sometimes gives high scores for clusterings by chance. Given the diverse nature of these metrics, if a labeling does well in all three of them, we can be relatively confident that it is not a degenerate solution that games the system.

5 Synthetic Experiments

Before running a user study, we test our hypothesis that topic model overviews and active learning selection improve final cluster quality compared to standard baselines: list overview and random selection. We simulate the four conditions on Congressional Bills and 20 Newsgroups.

Since we believe annotators create more specific labels compared to the gold labels, we use sub-labels as simulated user labels and labels as gold labels (we give examples of labels and sub-labels in Section 4.1). We start with two randomly selected documents that have different sub-labels, assign the corresponding sub-labels, then add more labels based on each condition's preference function (Section 3.3). We follow the condition's preference function and incrementally add labels until 100 documents have been labeled (100 documents are representative of what a human can label in about an hour). Given these labels, we compute purity, RI, and NMI over time. This procedure is repeated fifteen times (to account for the randomness of initial document selections and the preference functions with randomness).[11]

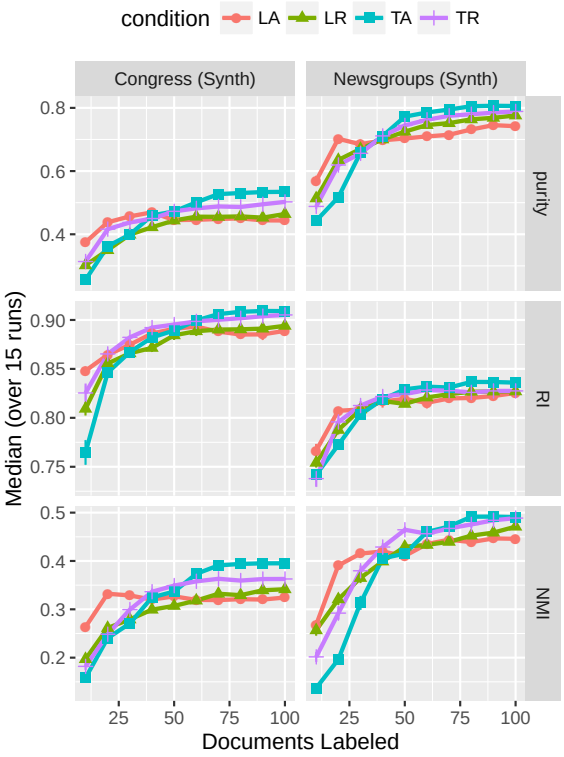


Figure 4: Synthetic results on US Congressional Bills and 20 Newsgroups data sets. Topic models help guide annotation attention to diverse segments of the data.

Synthetic results validate our hypothesis that topic overview and active learning selection can help label a corpus more efficiently (Figure 4). LA shows early gains, but tends to falter eventually compared to both topic overview and topic overview combined with active learning selection (TR and TA).

However, these experiments do not validate ALTO. Not all documents require the same time or effort to label, and active learning focuses on the hardest examples, which may confuse users. Thus, we need to evaluate how effectively actual users annotate a collection's documents.

6 User Study

Following the synthetic experiments, we conduct a user study with forty participants to evaluate ALTO (TA condition) against three alternatives that lack topic overview (LA), active learning selection (TR), or both (LR) (Sections 6.1 and 6.2). Then, we conduct a crowdsourced study to compare the overall

[11] Synthetic experiment data available at http://github.com/Pinafore/publications/tree/master/2016_acl_doclabel/data/synthetic_exp

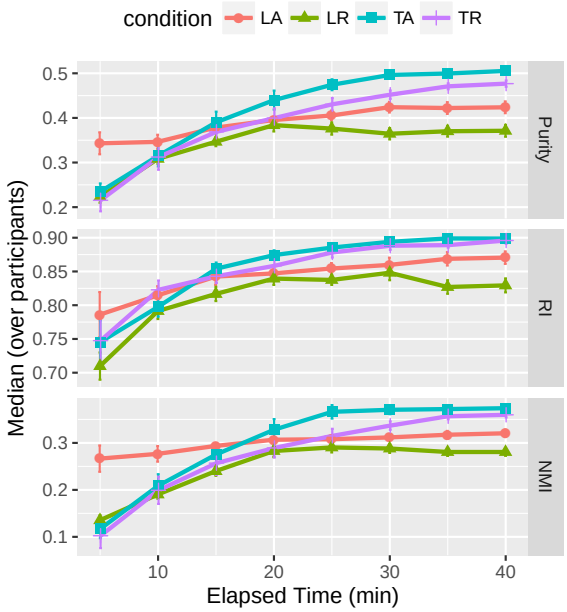


Figure 5: User study results on US Congressional Bills dataset. Active learning selection helps initially, but the combination of active learning selection and topic model overview has highest quality labels by the end of the task.

effectiveness of the label set generated by the participants in the four conditions (Section 6.3).

6.1 Method

We use the freelance marketplace Upwork to recruit online participants.[12] We require participants to have more than 90% job success on Upwork, English fluency, and US residency. Participants are randomly assigned to one of the four conditions and we recruited ten participants per condition.

Participants completed a demographic questionnaire, viewed a video of task instructions, and then interacted with the system and labeled documents until satisfied with the labels or forty minutes had elapsed.[13] The session ended with a survey, where participants rated mental, physical, and temporal demand, and performance, effort, and frustration on 20-point scales, using questions adapted from the NASA Task Load Index (Hart and Staveland, 1988, TLX). The survey also included 7-point scales for ease of coming up with labels, usefulness and satisfaction with the system, and—for TR and TA—topic information helpfulness. Each participant was paid fifteen dollars.[14]

For statistical analysis, we primarily use 2×2 (*overview* $\times$ *selection*) ANOVAs with Aligned Rank Transform (Wobbrock et al., 2011, ART), which is a non-parametric alternative to a standard ANOVA that is appropriate when data are not expected to meet the normality assumption of ANOVA.

	F		p	
	Overview	Selection	Overview	Selection
final purity	81.03	7.18	$< .001$	.011
final RI	39.89	6.28	$< .001$	.017
final NMI	70.92	9.87	$< .001$	.003

df(1,36) for all reported results

Table 2: Results from 2×2 ANOVA with ART analyses on the final purity, RI, and NMI metrics. Only main effects for the factors of *overview* and *selection* are shown; no interaction effects were statistically significant. Topics and active learning both had significant effects on quality scores.

6.2 Document Cluster Evaluation

We analyze the data by dividing the forty-minute labeling task into five minute intervals. If a participant stops before the time limit, we consider their final dataset to stay the same for any remaining intervals. Figure 5 shows the measures across study conditions, with similar trends for all three measures.

Topic model overview and active learning both significantly improve final dataset measures. The topic overview and active selection conditions significantly outperform the list overview and random selection, respectively, on the final label quality metrics. Table 2 shows the results of separate 2×2 ANOVAs with ART with each of final purity, RI, and NMI scores. There are significant main effects of *overview* and *selection* on all three metrics; no interaction effects were significant.

TR outperforms LA. Topic models by themselves outperform traditional active learning strategies (Figure 5). LA performs better than LR; while active learning was useful, it was not as useful as the topic model overview (TR and TA).

LA provides an initial benefit. Average purity, NMI and RI were highest with LA for the earliest labeling time intervals. Thus, when time is very

[12] http://Upwork.com

[13] Forty minutes of activity, excluding system time to classify and update documents. Participants nearly exhausted the time: 39.3 average minutes in TA, 38.8 in TR, 40.0 in LA, and 35.9 in LR.

[14] User study data available at http://github.com/Pinafore/publications/tree/master/2016_acl_doclabel/data/user_exp

	$M \pm SD\,[median]$		
	purity	RI	NMI
TA	$0.31 \pm 0.08\,[0.32]$	$0.80 \pm 0.05\,[0.80]$	$0.19 \pm 0.08\,[0.21]$
TR	$0.32 \pm 0.09\,[0.31]$	$0.82 \pm 0.04\,[0.82]$	$0.21 \pm 0.09\,[0.20]$
LA	$0.35 \pm 0.05\,[0.35]$	$0.82 \pm 0.04\,[0.81]$	$0.27 \pm 0.05\,[0.28]$
LR	$0.31 \pm 0.04\,[0.31]$	$0.79 \pm 0.04\,[0.79]$	$0.19 \pm 0.03\,[0.19]$

Table 3: Mean, standard deviation, and median purity, RI, and NMI after ten minutes. NMI in particular shows the benefit of LA over other conditions at early time intervals.

limited, using traditional active learning (LA) is preferable to topic overviews; users need time to explore the topics and a subset of documents within them. Table 3 shows the metrics after ten minutes. Separate 2×2 ANOVAs with ART on the means of purity, NMI and RI revealed a significant interaction effect between *overview* and *selection* on mean NMI ($F(1, 36) = 5.58$, p = .024), confirming the early performance trends seen in Figure 5 at least for NMI. No other main or interaction effects were significant, likely due to low statistical power.

Subjective ratings. Table 4 shows the average scores given for the six NASA-TLX questions in different conditions. Separate 2×2 ANOVA with ART for each of the measures revealed only one significant result: participants who used the topic model overview find the task to be significantly less frustrating ($M = 4.2$ and $median = 2$) than those who used the list overview ($M = 7.3$ and $median = 6.5$) on a scale from 1 (low frustration) to 20 (high frustration) ($F(1, 36) = 4.43$, $p = .042$), confirming that the topic overview helps users organize their thoughts and experience less stress during labeling.

Participants in the TA and TR conditions rate topic information to be useful in completing the task ($M = 5.0$ and $median = 5$) on a scale from 1 (not useful at all) to 7 (very useful). Overall, users are positive about their experience with the system. Participants in all conditions rate overall satisfaction with the interface positively ($M = 5.8$ and $median = 6$) on a scale from 1 (not satisfied at all) to 7 (very satisfied).

Discussion. One can argue that using topic overviews for labeling could have a negative effect: users may ignore the document content and focus on topics for labeling. We tried to avoid this issue by making it clear in the instructions that they need to focus on document content and use topics as a guidance. On average, the participants in TR create 1.96 labels per topic and the participants in TA created 2.26 labels per topic. This suggests that participants are going beyond what they see in topics for labeling, at least in the TA condition.

6.3 Label Evaluation Results

Section 6.2 compares clusters of documents in different conditions against the gold clusters but does not evaluate the quality of the labels themselves. Since one of the main contributions of ALTO is to accelerate inducing a high quality label set, we use crowdsourcing to assess how the final induced label sets compare in different conditions.

For completeness, we also compare labels against a fully automatic labeling method (Aletras and Stevenson, 2014) that does not require human intervention. We assign *automatic* labels to documents based on their most prominent topic.

We ask users on a crowdsourcing platform to *vote* for the "best" and "worst" label that describes the content of a US congressional bill (we use Crowdflower restricted to US contributors).

Five users label each document and we use the aggregated results generated by Crowdflower. The user gets $0.20 for each task.

We randomly choose 200 documents from our dataset (Section 4.1). For each chosen document, we randomly choose a participant from all four conditions (TA, TR, LA, LR). The labels assigned in different conditions and the automatic label of the document's prominent topic construct the candidate labels for the document.[15] Identical labels are merged into one label to avoid showing duplicate labels to users. If a merged label gets a "best" or "worst" vote, we split that vote across all the identical instances.[16] Figure 6 shows the average number of "best" and "worst" votes for each condition and the automatic method. ALTO (TA) receives the most "best" votes and the fewest "worst" votes. LR receives the most worst votes. The automatic labels, interestingly, appear to do at least as well as the list view labels, with a similar number of best votes and fewer worst votes. This indicates that automatic labels have reasonable quality compared to at least some manually generated labels. However, when users are provided with a topic model overview—

[15] Some participants had typos in the labels. We corrected all the typos using pyEnchant (`http://pythonhosted.org/pyenchant/`) spellchecker. If the corrected label was still wrong, we corrected it manually.

[16] Evaluation data available at `http://github.com/Pinafore/publications/tree/master/2016_acl_doclabel/data/label_eval`

Condition	Mental Demand	Physical Demand	Temporal Demand	Performance	Effort	Frustration
	$M \pm SD\,[median]$					
TA	$9.8 \pm 5.6\,[10]$	$2.9 \pm 3.4\,[2]$	$9 \pm 7.8\,[7]$	$5.5 \pm 5.8\,[1.5]$	$9.4 \pm 6.3\,[10]$	$4.5 \pm 5.5\,[1.5]$
TR	$10.6 \pm 4.5\,[11]$	$2.4 \pm 2.8\,[1]$	$7.4 \pm 4.1\,[9]$	$8.8 \pm 6.1\,[7.5]$	$9.8 \pm 3.7\,[10]$	$3.9 \pm 3.0\,[3.5]$
LA	$9.1 \pm 5.5\,[10]$	$1.7 \pm 1.3\,[1]$	$10.2 \pm 4.8\,[11]$	$8.6 \pm 5.3\,[10]$	$10.7 \pm 6.2\,[12.5]$	$6.7 \pm 5.1\,[5.5]$
LR	$9.8 \pm 6.1\,[10]$	$3.3 \pm 2.9\,[2]$	$9.3 \pm 5.7\,[10]$	$9.4 \pm 5.6\,[10]$	$9.4 \pm 6.2\,[10]$	$7.9 \pm 5.4\,[8]$

Table 4: Mean, standard deviation, and median results from NASA-TLX post-survey. All questions are scaled 1 (low)–20 (high), except performance, which is scaled 1 (good)–20 (poor). Users found topic model overview conditions, TR and TA, to be significantly less frustrating than the list overview conditions.

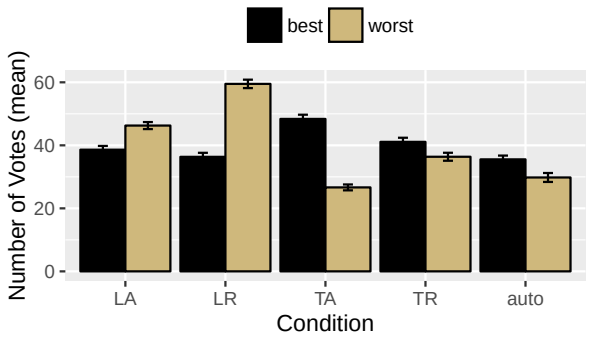


Figure 6: Best and worst votes for document labels. Error bars are standard error from bootstrap sample. ALTO (TA) gets the most best votes and the fewest worst votes.

with or without active learning selection—they can generate label sets that improve upon automatic labels and labels assigned without the topic model overview.

7 Related Work

Text classification—a ubiquitous machine learning tool for automatically labeling text (Zhang, 2010)—is a well-trodden area of NLP research. The difficulty is often creating the training data (Hwa, 2004; Osborne and Baldridge, 2004); coding theory is an entire subfield of social science devoted to creating, formulating, and applying labels to text data (Saldana, 2012; Musialek et al., 2016). Crowdsourcing (Snow et al., 2008) and active learning (Settles, 2012), can decrease the cost of annotation but only *after* a label set exists.

ALTO's corpus overviews aid text understanding, building on traditional interfaces for gaining both local and global information (Hearst and Pedersen, 1996). More elaborate interfaces (Eisenstein et al., 2012; Chaney and Blei, 2012; Roberts et al., 2014) provide richer information given a fixed topic model. Alternatively, because topic models are imperfect (Boyd-Graber et al., 2014), refining underlying topic models may also improve users' understanding of a corpus (Choo et al., 2013; Hoque and Carenini, 2015).

Summarizing document collections through discovered topics can happen through raw topics labeled manually by users (Talley et al., 2011), automatically (Lau et al., 2011), or by learning a mapping from labels to topics (Ramage et al., 2009). When there is not a direct correspondence between topics and labels, classifiers learn a mapping (Blei and McAuliffe, 2007; Zhu et al., 2009; Nguyen et al., 2015). Because we want topics to be consistent between users, we use a classifier with static topics in ALTO. Combining our interface with dynamic topics could improve overall labeling, perhaps at the cost of introducing confusion as topics change during the labeling process.

8 Conclusion and Future Work

We introduce ALTO, an interactive framework that combines both active learning *selection*s with topic model *overview*s to both help users induce a label set and assign labels to documents. We show that users can more effectively and efficiently induce a label set and create training data using ALTO in comparison with other conditions, which lack either topic *overview* or active *selection*.

We can further improve ALTO to help users gain better and faster understanding of text corpora. Our current system limits users to view only $20K$ documents at a time and allows for one label assignment per document. Moreover, the topics are static and do not adapt to better reflect users' labels. Users should have better support for browsing documents and assigning multiple labels.

Finally, with slight changes to what the system considers a document, we believe ALTO can be extended to NLP applications other than classification, such as named entity recognition or semantic role labeling, to reduce the annotation effort.

Acknowledgments

We thank the anonymous reviewers, David Mimno, Edward Scott Adler, Philip Resnik, and Burr Settles for their insightful comments. We also thank Nikolaos Aletras for providing the automatic topic labeling code. Boyd-Graber and Poursabzi-Sangdeh's contribution is supported by NSF Grant NCSE-1422492; Findlater, Seppi, and Boyd-Graber's contribution is supported by collaborative NSF Grant IIS-1409287 (UMD) and IIS-1409739 (BYU). Any opinions, findings, results, or recommendations expressed here are of the authors and do not necessarily reflect the view of the sponsor.

References

E Scott Adler and John Wilkerson. 2006. Congressional bills project. *NSF*, 880066:00880061.

Nikolaos Aletras and Mark Stevenson. 2014. Labelling topics using unsupervised graph-based methods. In *Proceedings of the Association for Computational Linguistics*, pages 631–636.

Pranav Anand, Joseph King, Jordan L Boyd-Graber, Earl Wagner, Craig H Martell, Douglas W Oard, and Philip Resnik. 2011. Believe me-we can do this! annotating persuasive acts in blog text. In *Computational Models of Natural Argument*.

David M. Blei and Jon D. McAuliffe. 2007. Supervised topic models. In *Proceedings of Advances in Neural Information Processing Systems*.

David M. Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3.

Gerlof Bouma. 2009. Normalized (pointwise) mutual information in collocation extraction. In *The Biennial GSCL Conference*, pages 31–40.

Jordan Boyd-Graber, David Mimno, and David Newman. 2014. Care and feeding of topic models: Problems, diagnostics, and improvements. *Handbook of Mixed Membership Models and Their Applications; CRC Press: Boca Raton, FL, USA*.

Ian Budge. 2001. *Mapping policy preferences: estimates for parties, electors, and governments, 1945-1998*, volume 1. Oxford University Press.

Bob Carpenter. 2008. Lingpipe 4.1.0. `http://alias-i.com/lingpipe`.

Allison Chaney and David Blei. 2012. Visualizing topic models. In *International AAAI Conference on Weblogs and Social Media*.

Jaegul Choo, Changhyun Lee, Chandan K. Reddy, and Haesun Park. 2013. UTOPIAN: User-driven topic modeling based on interactive nonnegative matrix factorization. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):1992–2001.

Sanjoy Dasgupta and Daniel Hsu. 2008. Hierarchical sampling for active learning. In *Proceedings of the International Conference of Machine Learning*.

Jacob Eisenstein, Duen Horng Chau, Aniket Kittur, and Eric Xing. 2012. TopicViz: interactive topic exploration in document collections. In *International Conference on Human Factors in Computing Systems*.

Sandra G Hart and Lowell E Staveland. 1988. Development of nasa-tlx (task load index): Results of empirical and theoretical research. *Advances in psychology*, 52:139–183.

M.A. Hearst and J.O. Pedersen. 1996. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*.

Enamul Hoque and Giuseppe Carenini. 2015. Convisit: Interactive topic modeling for exploring asynchronous online conversations. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15.

Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. 2014. Interactive topic modeling. *Machine learning*, 95(3):423–469.

Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2(1):193–218.

Rebecca Hwa. 2004. Sample selection for statistical parsing. *Computational linguistics*, 30(3):253–276.

Mohit Iyyer, Peter Enns, Jordan L Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the Association for Computational Linguistics*.

Edward F Kelly and Philip J Stone. 1975. *Computer recognition of English word senses*, volume 13. North-Holland.

Soo-Min Kim and Eduard Hovy. 2004. Determining the sentiment of opinions. In *Proceedings of the Association for Computational Linguistics*, page 1367. Association for Computational Linguistics.

Hans-Dieter Klingemann, Andrea Volkens, Judith Bara, Ian Budge, Michael D McDonald, et al. 2006. *Mapping policy preferences II: estimates for parties, electors, and governments in Eastern Europe, European Union, and OECD 1990-2003*. Oxford University Press Oxford.

Ken Lang. 2007. 20 newsgroups data set. http://www.ai.mit.edu/people/jrennie/20Newsgroups/.

Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic models. In *Proceedings of the Association for Computational Linguistics*, pages 1536–1545.

Jey Han Lau, David Newman, and Timothy Baldwin. 2014. Machine reading tea leaves: Automatically evaluating topic coherence and topic model quality. In *Proceedings of the European Chapter of the Association for Computational Linguistics*.

David D Lewis and William A Gale. 1994. A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet.

Marina Meilă. 2003. Comparing clusterings by the variation of information. In *Learning theory and kernel machines*, pages 173–187. Springer.

Chris Musialek, Philip Resnik, and S. Andrew Stavisky. 2016. Using text analytic techniques to create efficiencies in analyzing qualitative data: A comparison between traditional content analysis and a topic modeling approach. In *American Association for Public Opinion Research*.

Grace Ngai and David Yarowsky. 2000. Rule writing or annotation: Cost-efficient resource usage for base noun phrase chunking. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics.

Thang Nguyen, Jordan Boyd-Graber, Jeff Lund, Kevin Seppi, and Eric Ringger. 2015. Is your anchor going up or down? Fast and accurate supervised topic models. In *Conference of the North American Chapter of the Association for Computational Linguistics*.

Sonya Nikolova, Jordan Boyd-Graber, and Christiane Fellbaum. 2011. Collecting semantic similarity ratings to connect concepts in assistive communication tools. In *Modeling, Learning, and Processing of Text Technological Data Structures*, pages 81–93. Springer.

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In *Conference of the North American Chapter of the Association for Computational Linguistics*, pages 89–96. Citeseer.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of Empirical Methods in Natural Language Processing*.

William M Rand. 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Margaret E Roberts, Brandon M Stewart, Dustin Tingley, Christopher Lucas, Jetson Leder-Luis, Shana Kushner Gadarian, Bethany Albertson, and David G Rand. 2014. Structural topic models for open-ended survey responses. *American Journal of Political Science*, 58(4):1064–1082.

J. Saldana. 2012. *The Coding Manual for Qualitative Researchers*. SAGE Publications.

Burr Settles. 2012. Active learning (synthesis lectures on artificial intelligence and machine learning). *Long Island, NY: Morgan & Clay Pool*.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294. ACM.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Ng. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of Empirical Methods in Natural Language Processing*.

Alexander Strehl and Joydeep Ghosh. 2003. Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *The Journal of Machine Learning Research*, 3:583–617.

Edmund M. Talley, David Newman, David Mimno, Bruce W. Herr, Hanna M. Wallach, Gully A. P. C. Burns, A. G. Miriam Leenders, and Andrew McCallum. 2011. Database of NIH grants using machine-learned categories and graphical clustering. *Nature Methods*, 8(6):443–444, May.

Katrin Tomanek, Joachim Wermter, and Udo Hahn. 2007. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 486–495.

Paul MB Vitányi, Frank J Balbach, Rudi L Cilibrasi, and Ming Li. 2009. Normalized information distance. In *Information theory and statistical learning*, pages 45–82. Springer.

Jacob O Wobbrock, Leah Findlater, Darren Gergle, and James J Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 143–146. ACM.

Tong Zhang. 2010. Fundamental statistical techniques. In Nitin Indurkhya and Fred J. Damerau, editors, *Handbook of Natural Language Processing*. Chapman & Hall/CRC, 2nd edition.

Ying Zhao and George Karypis. 2001. Criterion functions for document clustering: Experiments and analysis. Technical report, University of Minnesota.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2009. MedLDA: maximum margin supervised topic models for regression and classification. In *Proceedings of the International Conference of Machine Learning*.

Predicting the Rise and Fall of Scientific Topics from Trends in their Rhetorical Framing

Vinodkumar Prabhakaran
Stanford University
vinod@cs.stanford.edu

William L. Hamilton
Stanford University
wleif@stanford.edu

Dan McFarland
Stanford University
dmcfarla@stanford.edu

Dan Jurafsky
Stanford University
jurafsky@stanford.edu

Abstract

Computationally modeling the evolution of science by tracking how scientific topics rise and fall over time has important implications for research funding and public policy. However, little is known about the mechanisms underlying topic growth and decline. We investigate the role of rhetorical framing: whether the rhetorical role or function that authors ascribe to topics (as methods, as goals, as results, etc.) relates to the historical trajectory of the topics. We train topic models and a rhetorical function classifier to map topic models onto their rhetorical roles in 2.4 million abstracts from the Web of Science from 1991-2010. We find that a topic's rhetorical function is highly predictive of its eventual growth or decline. For example, topics that are rhetorically described as results tend to be in decline, while topics that function as methods tend to be in early phases of growth.

1 Introduction

One of the most compelling research questions in the computational analysis of scientific literature is whether the vast collections of scientific text hold important clues about the dynamics involved in the evolution of science; clues that may help predict the rise and fall of scientific ideas, methods and even fields. Being able to predict scientific trends in advance could potentially revolutionize the way science is done, for instance, by enabling funding agencies to optimize allocation of resources towards promising research areas.

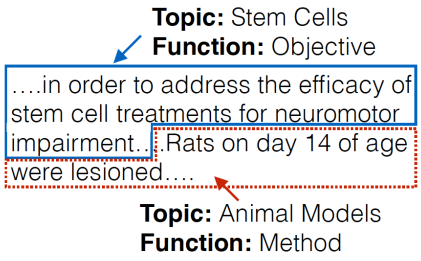


Figure 1: Example abstract snippet. The abstract rhetorically frames the *stem cells* topic as the OBJECTIVE of the research, while the *animal models* topic functions as the research METHOD.

Prior studies have often tracked scientific trends by applying topic modeling (Blei et al., 2003) based techniques to large corpora of scientific texts (Griffiths and Steyvers, 2004; Blei and Lafferty, 2006; Hall et al., 2008). They capture scientific ideas, methods, and fields in terms of *topics*, modeled as distributions over collection of words. These approaches usually adopt a decontextualized view of text and its usage, associating topics to documents based solely on word occurrences, disregarding where or how the words were employed. In reality, however, scientific abstracts often follow narrative structures (Crookes, 1986; Latour, 1987) that signal the specific *rhetorical roles* that different topics play within the research (Figure 1). The rhetorical role of a topic is the purpose or role it plays in the paper: as its background (scientific context), its objective/goal, the data employed, the design or method used (mode of inference), the results (what is found) or the conclusions (what they mean).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1170–1180,
Berlin, Germany, August 7-12, 2016.

RATIONALE: Neonatal ibotenic acid lesion of the ventral hippocampus was proposed as a relevant animal model of schizophrenia reflecting positive as well as negative symptoms of this disease. Before and after reaching maturity, specific alterations in the animals' social behaviour were found. **OBJECTIVE:** In this study, social behaviour of ventral hippocampal lesioned rats was analysed. For comparison, rats lesioned either in the ventral hippocampus or the dorsal hippocampus at the age of 8 weeks were tested. **METHODS:** Rats on day 7 of age were lesioned with ibotenic acid in the ventral hippocampus and social behaviour was tested at the age of 13 weeks. For comparison, adult 8-week-old rats were lesioned either in the ventral or the dorsal hippocampus. Their social behaviour was tested at the age of 18 weeks. **RESULTS:** It was found that neonatal lesion resulted in significantly decreased time spent in social interaction and an enhanced level of aggressive behaviour. This shift is not due to anxiety because we could not find differences between control rats and lesioned rats in the elevated plus-maze. Lesion in the ventral and dorsal hippocampus, respectively, in 8-week-old rats did not affect social behaviour. **CONCLUSIONS:** The results of our study indicate that ibotenic acid-induced hippocampal damage per se is not related to the shift in social behaviour. We favour the hypothesis that these changes are due to lesion-induced impairments in neurodevelopmental processes at an early stage of ontogenesis.

Figure 2: An example of a self-annotated abstract.
Source: `http://www.ncbi.nlm.nih.gov/pubmed/10435405`.

Rhetorical functions that topics take part in could hold important clues about the stage or development of an intellectual movement they stand to represent. For example, a topic that shifts over time from being employed as a method to being mentioned as background may signal an increase in its maturity and perhaps a corresponding decrease in its popularity among new research.

In this paper, we introduce a new algorithm to determine the rhetorical functions of topics associated with an abstract. There is much work on annotating and automatically parsing the rhetorical functions or narrative structure of scientific writing (e.g., Teufel, 2000; Chung, 2009; Gupta and Manning, 2011; de Waard and Maat, 2012). We derive insights from this prior work, but since we desire to apply our analysis to a broad range of domains, we build our narrative structure model based on over 83,000 self-labeled abstracts extracted from a variety of domains in the Web of Science corpus. Figure 2 shows an example of an abstract in which the authors have labeled the different narrative sections explicitly and identified the rhetorical functions. We use our narrative structure model to assign rhetorical function labels to scientific topics and show that these labels offer important clues indicating whether topics will eventually grow or decline.

Contributions: The three main contributions of our paper are: 1) we introduce the notion of the rhetorical scholarly functions of scientific *topics*, extending previous work which tended to focus on the rhetorical functions of individual *sentences*. We present an algorithm to assign rhetorical function labels to a topic as used in an individual paper; 2) we derive a new narrative scheme for scientific abstracts from over 83,000 abstracts that are labeled with narrative structures by their authors themselves, and present a tagger trained on this data that can parse unseen abstracts with 87% accuracy; 3) we show that the rhetorical function distribution of a topic reflects its temporal trajectory, and that it is predictive of whether the topic will eventually rise or fall in popularity.

2 Related Work

Our work builds upon a wealth of previous literature in both topic modeling and scientific discourse analysis, which we discuss in this section. We also discuss how our work relates to prior work on analyzing scientific trends.

2.1 Topic Modeling

Topic modeling has a long history of applications to scientific literature, including studies of temporal scientific trends (Griffiths and Steyvers, 2004; Steyvers et al., 2004; Wang and McCallum, 2006), article recommendation (Wang and Blei, 2011), and impact prediction (Yogatama et al., 2011). For example, Hall et al. (2008) and Anderson et al. (2012) show how tracking topic popularities over time can produce a 'computational history' of a particular scientific field (in their case ACL, where they tracked the rise of statistical NLP, among other dramatic changes).

Technical advancements in these areas usually correspond to modifications or extensions of the topic modeling (i.e., LDA) framework itself, such as by incorporating citation (Nallapati et al., 2008) or co-authorship information (Mei et al., 2008) directly into the topic model; Nallapati et al. (2011) employ such an extension to estimate the temporal "lead" or "lag" of different scientific information outlets. We contribute to this line of work by showing how we can build off of the standard LDA

framework—by overlaying rhetorical roles—and how this allows us to not only detect the growth and decline of scientific topics but also to predict these trends based upon the rhetorical roles being employed. Since our framework is structured as a pipeline (Figure 3) and works with the output of a topic modeling system, it is compatible with the vast majority of these extended topic models.

2.2 Scientific Discourse Analysis

Scientific discourse analysis is an active area of research with many different proposed schema of analysis — Argument Zones (Teufel, 2000), Information Structure (Guo et al., 2010), Core Scientific Concepts (Liakata, 2010), Research Aspects (Gupta and Manning, 2011), Discourse Segments (de Waard and Maat, 2012), Relation Structures (Tateisi et al., 2014), and Rhetorical Roles (Chung, 2009) to name a few. Most studies in this area focus on improving automatic discourse parsing of scientific text, while some works also focus on the linguistic patterns and psychological effects of scientific argumentation (e.g., de Waard and Maat, 2012). A wide range of techniques have been used in prior work to parse scientific abstracts, from fully supervised techniques (Chung, 2009; Guo et al., 2010) to semi-supervised (Guo et al., 2011c; Guo et al., 2013) and unsupervised techniques (Kiela et al., 2015).

Scientific discourse parsing has also been applied to other downstream tasks within the biomedical domain, such as information retrieval from randomized controlled trials in evidence based medicine (Chung, 2009; Kim et al., 2011; Verbeke et al., 2012), cancer risk assessment (Guo et al., 2011b), summarization (Teufel and Moens, 2002; Contractor et al., 2012), and question answering (Guo et al., 2013). Our work also falls in this category in the sense that our goal is to apply the rhetorical function parser to better understand the link between rhetoric and the historical trajectory of scientific ideas.

2.3 Scientific Trends Analysis

There is also a large body of literature in bibliometrics and scientometrics on tracking scientific trends using various citation patterns. Researchers have attempted to detect emerging research fronts using topological measures of citation networks (Shibata et al., 2008) as well as co-citation clusters (Small, 2006; Shibata et al., 2009). Unlike this line of work, our focus is not on citation patterns, but on how scientific trends are reflected in the texts of scientific publications.

Prior studies have also analyzed text to detect scientific trends. Mane and Börner (2004) and Guo et al. (2011a) use word burst detection (Kleinberg, 2003) to map new and emerging scientific fields, while Small (2011) examined sentiments expressed in the text surrounding citations, showing *uncertainty* in interdisciplinary citations contrasted with *utility* in within-discipline citations. In contrast to this previous work, we analyze the rhetorical function of automatically extracted topics from abstract text, without access to the citation context in full text.

3 Corpus

We use the Thomson Reuters Web of Science Core Collection, which contains scientific abstracts from over 8,500 of leading scientific and technical journals across 150 disciplines. We limit our study to the subset of abstracts from 1991 to 2010, which forms the majority of articles. This subset (denoted WoS hereafter) contains over 25 million articles from around 250 fields.

4 Rhetorical Functions of Scientific Topics

We use the term *rhetorical function* to identify the purpose or role a scientific topic plays within a research paper. This function qualifies the association between a topic and a paper.

A topic could represent the general domain of the research or its main objective/goal. It could also correspond to the data used, the way the research is designed, or the methods used. A topic may serve one or more of these roles within the same paper. The same topic may also serve different roles in different papers. We are interested in finding the different rhetorical functions by which topics are associated with the papers in our corpus, as a tool for understanding the growth and decline of topics over time. Our focus is thus on the rhetorical functions that topics play across papers, in order to understand the 'rhetorical structure of science' (Latour, 1987), although these are cued by specific rhetorical structures in individual sentences of individual papers. (Our work on the function of *topics* thus differs somewhat from previous research focusing on the rhetorical role of individual *sentences* or segments in the structure of the paper.)

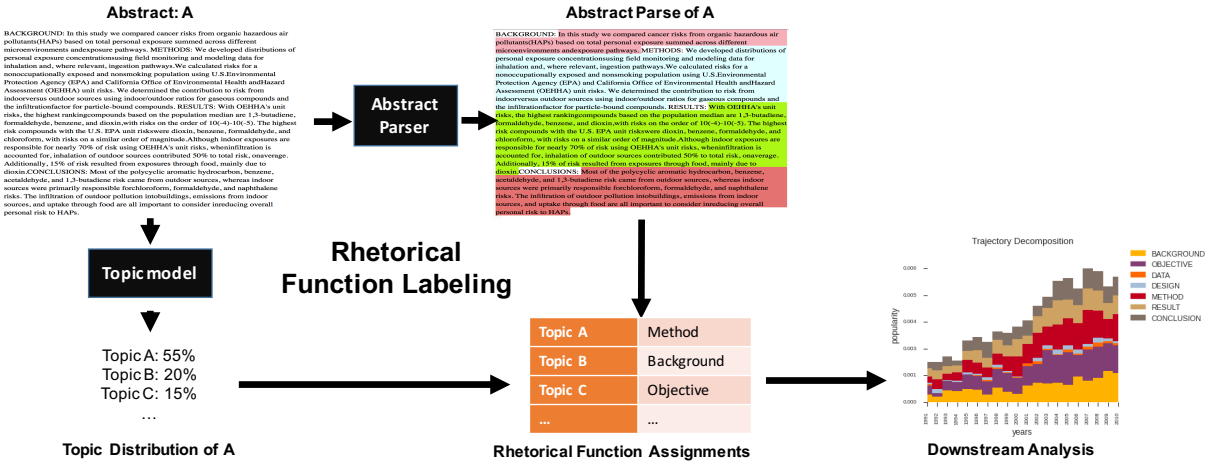


Figure 3: Rhetorical Function Labeling
The topic model (step a) assigns topic distributions to the abstract text (bottom left) and the abstract parser (step b) divides the text into discourse segments (top right). Rhetorical function labeling (step c) combines these two analyses to assign rhetorical functions to topics (bottom middle). These labels enrich the analysis of trends in topic popularity over decades (bottom right).

We follow a three-step process to assign rhetorical function labels to a large corpus of scientific abstracts. Figure 3 presents the pictorial representation of the procedure we follow — 1) obtain topic probabilities for each abstract, 2) parse the narrative structure of the abstracts in order to arrive at segments of text with different discourse intentions, and 3) superimpose the topic assignments and the abstract parse to arrive at the rhetorical function labels that capture how the topics are associated with the work presented in the abstract. We describe each of these steps in detail below.

a. Topic modeling: The core of our approach relies on the popular latent Dirichlet allocation (LDA) (Blei et al., 2003) algorithm. It probabilistically assigns both words and abstracts to different topics, in an unsupervised manner. Let $A = \{a_1, a_2, ...a_{|A|}\}$ be the set of abstracts within the field we are studying, and let $T = \{t_1, t_2, ...t_{|T|}\}$ be the set of different topics in the field. A topic model trained on A assigns $\theta_{a,t}$, the probability of topic t occurring in abstract a for all $a \in A$ and $t \in T$. The topic model also provides the $\phi_{t,w}$, the probability of word w occurring in topic t.

b. Abstract parsing: An abstract parser divides the abstract text into a sequence of discourse segments, with each segment assigned a specific label denoting its rhetorical purpose. Let $S(a) = (s_1, s_2, ...s_{|S(a)|})$ be the sequence of segments identified by the abstract parser, and let L denote the set of labels in the abstract parsing framework. The abstract parser assigns a label $l(s_i) \in L$, for each $s_i \in S(a)$.

c. Rhetorical Function Labeling: We tease apart the abstract-level topic distribution $\theta_{a,t}$ assigned by the topic model (step a) along the segments found by the abstract parser (step b), and find the topic weights on each label $\theta_{l,t}(a)$ by calculating topic weights for segments that are assigned label l, i.e., $\{s_i \in S(a) : l(s_i) = l\}$. We calculate the topic weights for each segment by aggregating the topic weights on each word derived from $\phi_{t,w}$ inferred by the topic model:

$$\theta_{l,t}(a) \propto \sum_{w_i \in s_i : l(s_i) = l} \phi_{t,w} \quad (1)$$

We first describe the abstract parsing system (step b) we built for this purpose in Section 5, before discussing the execution details of each of the above three steps in Section 6.

5 Abstract Parsing

Datasets with manual annotations for discourse structure of abstracts (e.g., Guo et al., 2010; Gupta and Manning, 2011) are few, small, and limited to specific domains. It is not clear how accurate an abstract parser trained on these datasets will perform on other domains. Since we want to obtain the structure of abstracts in a broad range of domains over different time-periods, a parser trained

on small datasets in specific domains may not be adequate for our purposes. Hence, we exploit the large number of abstracts in the WOS corpus in order to gather a dataset of self-labeled abstracts from a wide range of domains, over a period of two decades. By self-labeled abstracts, we refer to the abstracts where the authors have identified the discourse segments using explicit section labels.

5.1 Extracting Self-labeled Abstracts

In the first step, we extract all the patterns from the WOS corpus that could potentially be a segment label. For this, we look for a pattern that is commonly used by authors to label abstract segments — a capitalized phrase of one or more words occurring at the beginning of the abstract or preceded by a period, and followed by a ":". We obtained 455,972 matches for the above pattern, corresponding to 2,074 unique labels, majority of which were valid discourse segment labels. These include variations of the same labels (e.g., "OBJECTIVE" and "AIM", "CONCLUSION" and "CONCLUSIONS" etc.) and typos (e.g., "RESLUTS"). There were also instances where two common labels were combined (e.g., "DATA AND METHODS"). The extracted matches also contained a long tail of false positives (e.g., "BMI").

One of the challenges in using the set of abstracts we obtained above is that they do not follow a common labeling scheme. Hence, we manually analyzed the top 100 unique labels (which corresponds to labels with more than $\sim$50 instances) and mapped them into a unified labeling scheme, grouping together labels with similar intentions. This resulted in a typology of seven labels:

- BACKGROUND: The scientific context
- OBJECTIVE: The specific goal(s)
- DATA: The empirical dimension used
- DESIGN: The experimental setup
- METHOD: Means used to achieve the goal
- RESULT: What was found
- CONCLUSION: What was inferred

We use this mapping to obtain abstracts that are self-labeled. We exclude the abstracts that had combined labels, since they may add noise to the training data. We also exclude abstracts that contained only false positive matches. This preprocessing resulted in a dataset of 83,559 abstracts. We refer to this dataset as SL, hereafter. We divide the SL dataset into Train/Dev/Test subsets for our experiments (Table 1).

	Train	Dev	Test
# of abstracts	58,600	12,331	12,628
# of labeled segments	243,217	51,111	52,403
# of sentences	681,730	143,792	147,321

Table 1: Statistics of self-labeled abstracts

5.2 Automatic tagging of abstracts

We use the SL dataset to build a supervised learning system that can predict the abstract structure in unseen documents. We perform the prediction at the sentence level. We used the CRF algorithm to train the model, as it has been proven successful in similar tasks in prior work (Hirohata et al., 2008; Merity et al., 2009). In our preliminary experiments, we also tried using SVM, but CRF was faster and outperformed SVM by 4-5% points consistently. We use the following features:

1. Location: location of the sentence from the beginning or end of the abstract
2. Word ngrams: unigrams and bigrams of word lemmas
3. Part-of-speech ngrams: unigrams and bigrams of part-of-speech tags
4. Verb lemmas and part-of-speeches: lemmas of verbs, and their part-of-speech tags in order to identify the tense of verb usage
5. Verb classes: we looked up each verb in the VerbNet (Kipper-Schuler, 2005) index and added the VerbNet class to the feature set if the verb maps to a unique verb class.
6. Concreteness rating: we used the max, min, and mean concreteness ratings of words based on (Brysbaert et al., 2014).

Most of these features are commonly used in similar tasks, while the concreteness features are new (and significantly improved performance). We do not use parse features, however, since our pipeline emphasizes computational efficiency, and parse features showed minimal utility relative to their computational cost in prior work (Guo et al., 2010).

We evaluate the performance of our learned model in terms of overall accuracy as well as per-class precision, recall and F-measure of predicting the segment labels at the sentence level. We performed experiments on the Dev set to choose the best feature configuration (e.g., tuning for word and part-of-speech ngram length). Each feature set described in the previous paragraph were con-

	Precision	Recall	F-measure
BACKGROUND	74.6	77.2	75.8
OBJECTIVE	85.2	81.8	83.5
DATA	82.6	76.8	79.6
DESIGN	68.0	64.8	66.3
METHOD	80.4	80.1	80.2
RESULT	90.8	93.3	92.0
CONCLUSION	93.8	92.0	92.9
Accuracy		86.6	

Table 2: Results of parsing abstract structure

tributing features to the best performance obtained on the Dev set. The concreteness features we introduced significantly improved the overall accuracy by around 2%.

Table 2 shows the results obtained on testing the final classifier system on the Test subset of SL.[1] We obtain an overall high accuracy of 86.6% at the sentence level. While RESULT and CONCLUSION obtained F-measures above 90%, OBJECTIVE and METHOD reported reasonable F-measures above 80%. DESIGN obtained the lowest precision, recall and F-measure. Overall, the performance we obtain is in the range of other reported results in similar tasks (Guo et al., 2013).

6 Analysis Setup

In the rest of this paper, we apply the rhetorical function labeling system described in Section 4 to analyze the growth and decline of scientific topics. We chose four diverse fields from the WoS corpus with large numbers of abstracts for our analysis, which are: *Biochemistry & Molecular Biology* (BIO): 850,394 abstracts, *Applied Physics* (PHY): 558,934 abstracts, *Physical Chemistry* (CHM): 533,871 abstracts, and *Neurosciences* (NEU): 477,197 abstracts. We apply the steps (a), (b), and (c) of rhetorical function labeling as described in Section 4 to these fields as follows:

Topic Modeling: We use the LightLDA implementation (Yuan et al., 2015) of LDA. It employs a highly efficient and parallelized Metropolis-Hastings sampler that allows us to scale our approach to massive datasets (e.g., millions of abstracts in our case). For all our experiments, we ran the algorithm for 1000 iterations, as this was sufficient for convergence. We use 500 topics for all four fields, but otherwise use the default hyperparameter settings from the LightLDA package.[2]

Abstract Parsing We applied the 7-label abstract parsing system described in Section 5 on all abstracts in each of the four disciplines.

Rhetorical Function Labeling Once the steps (a) and (b) are completed, we applied the rhetorical function labeling step (c) from Section 4 in order to obtain topic weights for each segment. In addition, we calculate a rhetorical function label distribution (referred to as *label distribution* hereafter) for each topic associated with an abstract.

7 Dissecting Topic Trajectories

In this section, we investigate whether the label distribution of the topics (i.e., across the rhetorical function labels) sheds light on the kind of trajectories they follow; in particular, whether it can predict the up-trend vs. down-trend of topics. We formalize the problem as follows: given two sets of topics clustered based on their historical trajectories, do the label-distribution based features of a topic have predictive power to classify the topic to be in either set?

7.1 Tracking topic popularities

For each field, we first calculated the number of articles in which each topic occurred in at least one of the rhetorical functions for each year in the 20-year period 1991-2010. Since the fields themselves grow over the years, we divide this number by the number of articles within that field to obtain the popularity of each topic in any year:

$$popularity(t, y) = DocsWithTopic(t)/|A_y|,$$

where A_y denotes the subset of articles in year y.

7.2 Detecting growing vs. declining topics

We are interested in the set of topics in each field that are either growing or declining. Figure 4 shows example popularity trends for two such topics in neuroscience: stem cell research, which sees

[1] We report only the results obtained in unseen abstracts in the Test set due to lack of space. Similar performance was obtained in the Dev set as well.

[2] In preliminary experiments, we used 100, 500 and 1000 topics. Upon manual inspection, we found that 500 topics resulted in a granularity that better captures the scientific intellectual movements within fields.

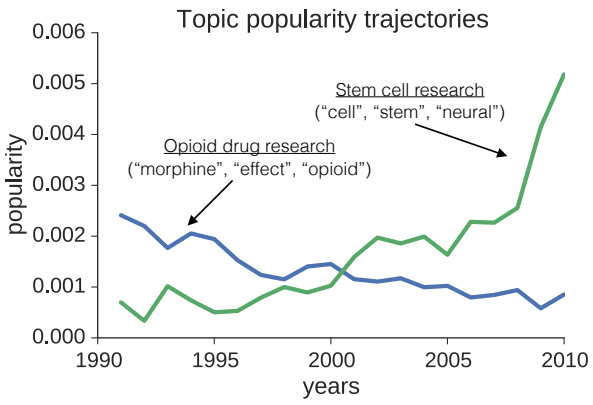


Figure 4: Example topic popularity curves from 1991 till 2010. Stem cell research (green) sees an increase in popularity over this time period, while Opioid research (blue) declines in popularity.

a dramatic increase in popularity from 1991-2010, and opioid[3] drug research, which declines in popularity during the same time-period.

Of course, topics do not always follow trajectories of pure growth or decline. A topic may have risen and subsequently fallen in popularity over the period of 20 years, or may stay more or less the same throughout (Griffiths and Steyvers, 2004). Hence, categorizing topics to be growing or declining solely based on the popularity at the beginning and end of the time period is problematic. We circumvent this issue and avoid manually-defined thresholds by discovering topical growth/decline curves in an unsupervised fashion. We use the K-Spectral Centroid (K-SC) algorithm (Yang and Leskovec, 2011), which groups different time-series into clusters, such that similar shapes get assigned to the same cluster, irrespective of scaling and translation (unlike other popular time series clustering algorithms such as Dynamic Time Warping). We run the clustering algorithm using $K = 3$, and choose the cluster with the upward trending centroid to be the set of growing topics and the cluster with the downward trending centroid to be the set of declining topics.[4] Figure 5 shows example centroids from Physical Chemistry, which clearly exhibit decreasing, increasing, and non-changing trends.

[3]Opioids are a popular class of analgesic (i.e., painkilling) drugs, including morphine.

[4]We also performed experiments using K = 5, 10 and 15 and grouped the clusters that are growing vs. declining. We obtained similar results in those experiments as well.

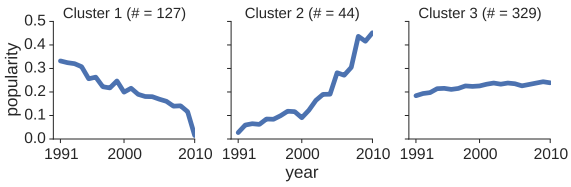


Figure 5: Cluster centroids in Physical Chemistry
Cluster 1: topics that declined in popularity;
Cluster 2: topics that grew in popularity;
Cluster 3: topics that stayed mostly the same.

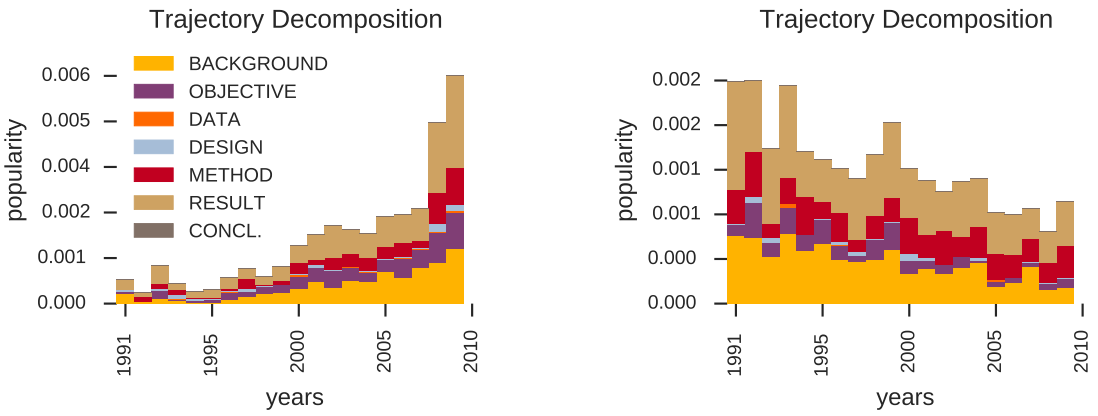
7.3 Characterizing topic growth vs. decline

We not only seek to detect growing vs. declining topics; our goal is to *characterize* these trajectories based upon the rhetorical functions that the topics are fulfilling during different points of their life-cycles. Figure 6 shows how the rhetorical function label distributions of the opioid and stem cell research topics shift from 1991 to 2010. The opioid research topic, which declines in popularity during this time-period, is frequently discussed in the RESULT and BACKGROUND roles during the early years. In contrast, the stem cell research topic, which is dramatically increasing in popularity, only begins to be discussed frequently in these roles towards the end of the time-period. Intuitively, these shifts make sense: topics become results-oriented and mentioned as background as they reach their peak; this peak is seen at the beginning of the time-period for opioid research, while stem cell research appears to be increasing towards a peak by the end of our data (i.e., 2010). These observations indicate that the rhetorical functions which a topic is fulfilling may be indicative of its current growth vs. decline.

7.4 Experiments

We perform two sets of experiments to quantify the qualitative insights noted above, i.e. that a topic's rhetorical function distribution is indicative of its eventual growth vs. decline. First, we show that a topic's rhetorical function distribution over the entire time-period can be used to *classify* the topic as either growing or declining. Next, we show that using only the first five years of data, we can *predict* whether a topic will eventually grow or decline. We use two sets of features:

- **label-distribution-percents (LD-%)**: seven features corresponding to the percentage of topics across the seven rhetorical function labels (e.g., % of time the topic is a *METHOD*)



(a) Stem cell topic.

(b) Opioid topic.

Figure 6: Examples of topics with changing rhetorical functions over time.

- **label-distribution-changes (LD-Δ)**: seven features that aggregate the mean change in this percentage over the years (e.g., is the % of being METHOD going up/down?).

These features are fed into a standard $L2$-regularized logistic regression classifier (with regularization parameter $C = 10$), which predicts whether a topic belongs to the growing or declining cluster.[5] We use random prediction and majority prediction as two uninformed baselines.

Classification task

System	ALL	BIO	PHY	CHM	NEU
Random	50.3	47.2	47.8	50.9	51.2
Majority	56.1	56.3	81.6	74.3	56.6
LR	74.2	81.0	83.3	81.9	74.8
LR - LD-R	71.3	77.7	81.6	73.1	70.5

Table 3: Results on classifying trajectories
LR: Logistic Regression using LD-%, LD-Δ, and LD-R.
LR - LD-R: Logistic Regression without using LD-R.

For the task of classifying a topic as either growing or declining, we compute the LD-% and LD-Δ features separately over the first and last five years of the data. This allows the model to analyze how the rhetorical functions are differentially expressed at the start vs. the end of the period under consideration. We also add a feature **label-distribution-ratio (LD-R)**, which is the ratio of the end to beginning LD-% values, so that the model has access to relative increases/decreases in the label distributions over the entire time-period.

[5]Similar performance was obtained with a linear SVM

Table 3 shows the performance of our model on this task. As expected a topic's label distribution over its entire life-time is very informative with respect to classifying the topic as growing or declining. We achieve a significant improvement over the baselines on the full dataset (32.3% relative improvement over majority prediction), and this trend holds across each field separately. The ratio feature proved to be extremely predictive in this task, i.e. relative increases/decreases in a topic being used in different functional roles are very predictive of the type of its trajectory.

Prediction task

We now turn to the more difficult task of *predicting* whether a topic will grow or decline, given access only to the first five years of its label distribution. The setup here is identical to the classification task, except that we now have access only to the LD-Δ and LD-% features aggregated over the first five years of data.

System	Accuracy on ALL
LD-% + LD-Δ	72.1
LD-% only	71.0
LD-Δ only	60.4

Table 4: Results on predicting trajectory

Table 4 shows the performance of our model on this task. (The baseline performances are the same as in the classification task). These results show that we can accurately predict whether a topic will grow or decline using only a small amount of data.

Label	BKGRND.	OBJ.	DATA	DESIGN	METHOD	RESULT	CONC.
LD-% Weight	-1.21	-0.10	6.38	1.21	3.82	-8.65	1.67
LD-Δ Weight	2.05	-0.01	2.20	-1.08	-1.63	-0.26	-1.27

Table 5: Logistic Regression feature weights for the prediction task on the full (ALL) dataset.

Moreover, we see that both percentage and delta features are necessary for this task.

7.5 Analysis

The feature weights of our learned linear model also provide insights into the dynamics of scientific trends. Table 5 contains the learned feature weights for the prediction task. Overall, these weights reinforce the conclusions drawn from the case study in Section 7.3. The strongest feature is the LD-% feature for the RESULT rhetorical function, which is highly negative. This indicates that topics that are currently being discussed as a result are likely at the peak of their popularity, and are thus likely to decrease in the future. Interestingly, the weights on the LD-% features for the methodological rhetorical functions (METHODS, DATA, and DESIGN) are all significantly positive. This suggests that topics occupying these functions may have reached a level of maturity where they are active areas of research and are being consumed by a large number of researchers, but that they have not yet peaked in their popularity. Finally, we see that the weights for the BACKGROUND and CONCLUSION roles have opposite trends: growing topics are more often mentioned as conclusions whereas dying topics, i.e. topics at the peak of their life-cycles, tend to be mentioned in background, or contextualizing, statements.

8 Conclusion

We introduce a novel framework for assigning rhetorical functions to associations between scientific topics and papers, and we show how these rhetorical functions are predictive of a topic's growth vs. decline.

Our analysis reveals important regularities with respect to how a topic's usage evolves over its life-cycle. We find that topics that are currently discussed as results tend to be in decline, whereas topics that are playing a methodological role tend to be in the early phases of growth. In some ways these results are counter-intuitive; for example, one might expect topics that are being discussed as results to be the focus of current cutting edge research, and that methodological topics might be more mundane and lacking in novelty. Instead our results suggest that results-oriented topics are at the peak of their life-cycle, while methodological topics still have room to grow. This result has important implications for research funding and public policy: the most promising topics—in terms of potential for future growth—are not those that are currently generating the most results, but are instead those that are active areas of methodological inquiry.

Our analysis does suffer from some limitations. Examining only 20 years of scientific progress prevents us from analyzing drastic scientific changes, e.g. paradigm shifts, that are only obvious over longer time-scales (Kuhn, 2012). Access to longer time-spans—along with varying data sources such as grants and patents—would also allow us to more completely model the trajectory of a topic as it moves from being active area of research to potentially impacting commercial industries and economic development. Nonetheless, we hope this work offers another step towards using computational tools to better understand the 'rhetorical structure of science' (Latour, 1987).

Acknowledgments

This work was supported by the NSF Award IIS-1159679, by the Stanford Data Science Initiative, and the Brown Institute for Media Innovation. W.H. was supported by the SAP Stanford Graduate Fellowship. We would also like to thank the ACL anonymous reviewers for their constructive feedback.

References

Ashton Anderson, Dan McFarland, and Dan Jurafsky. 2012. Towards a computational history of the ACL: 1980-2008. In *Proceedings of the ACL-2012 Special Workshop on Rediscovering 50 Years of Discoveries*, pages 13–21. Association for Computational Linguistics.

David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *The Journal of machine Learning research*, 3:993–1022.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Grace Y. Chung. 2009. Sentence retrieval for abstracts of randomized controlled trials. *BMC Medical Informatics and Decision Making*, 9(1):1–13.

Danish Contractor, Yufan Guo, and Anna Korhonen. 2012. Using argumentative zones for extractive summarization of scientific articles. In *COLING*, volume 12, pages 663–678. Citeseer.

Graham Crookes. 1986. Towards a validated analysis of scientific text structure. *Applied linguistics*, 7(1):57–70.

Anita de Waard and Henk Pander Maat. 2012. Verb form indicates discourse segment type in biological research papers: Experimental evidence. *Journal of English for Academic Purposes*, 11(4):357 – 366.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235.

Yufan Guo, Anna Korhonen, Maria Liakata, Ilona Silins Karolinska, Lin Sun, and Ulla Stenius. 2010. Identifying the information structure of scientific abstracts: an investigation of three different schemes. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, pages 99–107. Association for Computational Linguistics.

Hanning Guo, Scott Weingart, and Katy Börner. 2011a. Mixed-indicators model for identifying emerging research areas. *Scientometrics*, 89(1):421–435.

Yufan Guo, Anna Korhonen, Maria Liakata, Ilona Silins, Johan Hogberg, and Ulla Stenius. 2011b. A comparison and user-based evaluation of models of textual information structure in the context of cancer risk assessment. *BMC bioinformatics*, 12(1):1.

Yufan Guo, Anna Korhonen, and Thierry Poibeau. 2011c. A weakly-supervised approach to argumentative zoning of scientific documents. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 273–283, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Yufan Guo, Ilona Silins, Ulla Stenius, and Anna Korhonen. 2013. Active learning-based information structure analysis of full scientific articles and two applications for biomedical literature review. *Bioinformatics*, 29(11):1440–1447.

Sonal Gupta and Christopher Manning. 2011. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1–9, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

David Hall, Daniel Jurafsky, and Christopher D. Manning. 2008. Studying the history of ideas using topic models. In *Proceedings of the conference on empirical methods in natural language processing*, pages 363–371. Association for Computational Linguistics.

Kenji Hirohata, Naoaki Okazaki, Sophia Ananiadou, and Mitsuru Ishizuka. 2008. Identifying sections in scientific abstracts using conditional random fields. In *In Proc. of the IJCNLP 2008*.

Douwe Kiela, Yufan Guo, Ulla Stenius, and Anna Korhonen. 2015. Unsupervised discovery of information structure in biomedical documents. *Bioinformatics*, 31(7):1084–1092.

S Nam Kim, David Martinez, Lawrence Cavedon, and Lars Yencken. 2011. Automatic classification of sentences to support evidence based medicine. *BMC bioinformatics*, 12(2):1.

Karin Kipper-Schuler. 2005. *VerbNet: a broad-coverage, comprehensive verb lexicon*. Ph.D. thesis, Computer and Information Science Department, Universiy of Pennsylvania, Philadelphia, PA.

Jon Kleinberg. 2003. Bursty and hierarchical structure in streams. *Data Mining and Knowledge Discovery*, 7(4):373–397.

Thomas S Kuhn. 2012. *The structure of scientific revolutions*. University of Chicago press.

Bruno Latour. 1987. *Science in action: How to follow scientists and engineers through society*. Harvard university press.

Maria Liakata. 2010. Zones of conceptualisation in scientific papers: a window to negative and speculative statements. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 1–4, Uppsala, Sweden, July. University of Antwerp.

Ketan K Mane and Katy Börner. 2004. Mapping topics and topic bursts in pnas. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5287–5290.

Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008. Topic modeling with network regularization. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 101–110, New York, NY, USA. ACM.

Stephen Merity, Tara Murphy, and James R Curran. 2009. Accurate argumentative zoning with maximum entropy models. In *Proceedings of the 2009 Workshop on Text and Citation Analysis for Scholarly Digital Libraries*, pages 19–26. Association for Computational Linguistics.

Ramesh M. Nallapati, Amr Ahmed, Eric P. Xing, and William W. Cohen. 2008. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 542–550, New York, NY, USA. ACM.

Ramesh Maruthi Nallapati, Xiaolin Shi, Daniel McFarland, Jure Leskovec, and Daniel Jurafsky. 2011. Leadlag lda: Estimating topic specific leads and lags of information outlets. In *Fifth International AAAI Conference on Weblogs and Social Media*.

Naoki Shibata, Yuya Kajikawa, Yoshiyuki Takeda, and Katsumori Matsushima. 2008. Detecting emerging research fronts based on topological measures in citation networks of scientific publications. *Technovation*, 28(11):758–775.

Naoki Shibata, Yuya Kajikawa, Yoshiyuki Takeda, and Katsumori Matsushima. 2009. Comparative study on methods of detecting research fronts using different types of citation. *Journal of the American Society for Information Science and Technology*, 60(3):571–580.

Henry Small. 2006. Tracking and predicting growth areas in science. *Scientometrics*, 68(3):595–610.

Henry Small. 2011. Interpreting maps of science using citation context sentiments: A preliminary investigation. *Scientometrics*, 87(2):373–388, May.

Mark Steyvers, Padhraic Smyth, Michal Rosen-Zvi, and Thomas Griffiths. 2004. Probabilistic author-topic models for information discovery. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 306–315, New York, NY, USA. ACM.

Yuka Tateisi, Yo Shidahara, Yusuke Miyao, and Akiko Aizawa. 2014. Annotation of computer science papers for semantic relation extrac-tion. In *LREC*, pages 1423–1429.

Simone Teufel and Marc Moens. 2002. Summarizing scientific articles: experiments with relevance and rhetorical status. *Computational linguistics*, 28(4):409–445.

Simone Teufel. 2000. *Argumentative zoning: information extraction from scientific text*. Ph.D. thesis, University of Edinburgh.

Mathias Verbeke, Vincent Van Asch, Roser Morante, Paolo Frasconi, Walter Daelemans, and Luc De Raedt. 2012. A statistical relational learning approach to identifying evidence based medicine categories. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 579–589. Association for Computational Linguistics.

Chong Wang and David M. Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 448–456, New York, NY, USA. ACM.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 424–433. ACM.

Jaewon Yang and Jure Leskovec. 2011. Patterns of temporal variation in online media. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 177–186. ACM.

Dani Yogatama, Michael Heilman, Brendan O'Connor, Chris Dyer, Bryan R Routledge, and Noah A Smith. 2011. Predicting a scientific community's response to an article. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 594–604. Association for Computational Linguistics.

Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, and Wei-Ying Ma. 2015. LightLDA: Big Topic Models on Modest Computer Clusters. In *Proceedings of the 24th International Conference on World Wide Web*.

Compositional Sequence Labeling Models for Error Detection in Learner Writing

Marek Rei
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
marek.rei@cl.cam.ac.uk

Helen Yannakoudakis
The ALTA Institute
Computer Laboratory
University of Cambridge
United Kingdom
helen.yannakoudakis@cl.cam.ac.uk

Abstract

In this paper, we present the first experiments using neural network models for the task of error detection in learner writing. We perform a systematic comparison of alternative compositional architectures and propose a framework for error detection based on bidirectional LSTMs. Experiments on the CoNLL-14 shared task dataset show the model is able to outperform other participants on detecting errors in learner writing. Finally, the model is integrated with a publicly deployed self-assessment system, leading to performance comparable to human annotators.

1 Introduction

Automated systems for detecting errors in learner writing are valuable tools for second language learning and assessment. Most work in recent years has focussed on error correction, with error detection performance measured as a byproduct of the correction output (Ng et al., 2013; Ng et al., 2014). However, this assumes that systems are able to propose a correction for every detected error, and accurate systems for correction might not be optimal for detection. While closed-class errors such as incorrect prepositions and determiners can be modeled with a supervised classification approach, content-content word errors are the 3rd most frequent error type and pose a serious challenge to error correction frameworks (Leacock et al., 2014; Kochmar and Briscoe, 2014). Evaluation of error correction is also highly subjective and human annotators have rather low agreement on gold-standard corrections (Bryant and Ng, 2015). Therefore, we treat error detection in learner writing as an independent task and propose a system for labeling each token as being correct or incorrect in context.

Common approaches to similar sequence labeling tasks involve learning weights or probabilities for context n-grams of varying sizes, or relying on previously extracted high-confidence context patterns. Both of these methods can suffer from data sparsity, as they treat words as independent units and miss out on potentially related patterns. In addition, they need to specify a fixed context size and are therefore often limited to using a small window near the target.

Neural network models aim to address these weaknesses and have achieved success in various NLP tasks such as language modeling (Bengio et al., 2003) and speech recognition (Dahl et al., 2012). Recent developments in machine translation have also shown that text of varying length can be represented as a fixed-size vector using convolutional networks (Kalchbrenner and Blunsom, 2013; Cho et al., 2014a) or recurrent neural networks (Cho et al., 2014b; Bahdanau et al., 2015).

In this paper, we present the first experiments using neural network models for the task of error detection in learner writing. We perform a systematic comparison of alternative compositional structures for constructing informative context representations. Based on the findings, we propose a novel framework for performing error detection in learner writing, which achieves state-of-the-art results on two datasets of error-annotated learner essays. The sequence labeling model creates a single variable-size network over the whole sentence, conditions each label on all the words, and predicts all labels together. The effects of different datasets on the overall performance are investigated by incrementally providing additional training data to the model. Finally, we integrate the error detection framework with a publicly deployed self-assessment system, leading

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1181–1191,
Berlin, Germany, August 7-12, 2016.

to performance comparable to human annotators.

2 Background and Related Work

The field of automatically detecting errors in learner text has a long and rich history. Most work has focussed on tackling specific types of errors, such as usage of incorrect prepositions (Tetreault and Chodorow, 2008; Chodorow et al., 2007), articles (Han et al., 2004; Han et al., 2006), verb forms (Lee and Seneff, 2008), and adjective-noun pairs (Kochmar and Briscoe, 2014).

However, there has been limited work on more general error detection systems that could handle all types of errors in learner text. Chodorow and Leacock (2000) proposed a method based on mutual information and the chi-square statistic to detect sequences of part-of-speech tags and function words that are likely to be ungrammatical in English. Gamon (2011) used Maximum Entropy Markov Models with a range of features, such as POS tags, string features, and outputs from a constituency parser. The pilot Helping Our Own shared task (Dale and Kilgarriff, 2011) also evaluated grammatical error detection of a number of different error types, though most systems were error-type specific and the best approach was heavily skewed towards article and preposition errors (Rozovskaya et al., 2011). We extend this line of research, working towards general error detection systems, and investigate the use of neural compositional models on this task.

The related area of grammatical error *correction* has also gained considerable momentum in the past years, with four recent shared tasks highlighting several emerging directions (Dale and Kilgarriff, 2011; Dale et al., 2012; Ng et al., 2013; Ng et al., 2014). The current state-of-the-art approaches can broadly be separated into two categories:

1. Phrase-based statistical machine translation techniques, essentially translating the incorrect source text into the corrected version (Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2014)

2. Averaged Perceptrons and Naive Bayes classifiers making use of native-language error correction priors (Rozovskaya et al., 2014; Rozovskaya et al., 2013).

Error correction systems require very specialised models, as they need to generate an improved version of the input text, whereas a wider range of tagging and classification models can be deployed on error detection. In addition, automated writing feedback systems that indicate the presence and location of errors may be better from a pedagogic point of view, rather than providing a panacea and correcting all errors in learner text. In Section 7 we evaluate a neural sequence tagging model on the latest shared task test data, and compare it to the top participating systems on the task of error detection.

3 Sequence Labeling Architectures

We construct a neural network sequence labeling framework for the task of error detection in learner writing. The model receives only a series of tokens as input, and outputs the probability of each token in the sentence being correct or incorrect in a given context. The architectures start with the vector representations of individual words, $[x_1, ..., x_T]$, where T is the length of the sentence. Different composition functions are then used to calculate a hidden vector representation of each token in context, $[h_1, ..., h_T]$. These representations are passed through a softmax layer, producing a probability distribution over the possible labels for every token in context:

$$p_t = softmax(W_o h_t) \quad (1)$$

where W_o is the weight matrix between the hidden vector h_t and the output layer.

We investigate six alternative neural network architectures for the task of error detection: convolutional, bidirectional recurrent, bidirectional LSTM, and multi-layer variants of each of them. In the **convolutional** neural network (CNN, Figure 1a) for token labeling, the hidden vector h_t is calculated based on a fixed-size context window. The convolution acts as a feedforward network, using surrounding context words as input, and therefore it will learn to detect the presence of different types of n-grams. The assumption behind the convolutional architecture is that memorising erroneous token sequences from the training data is sufficient for performing error detection.

The convolution uses d_w tokens on either side of the target token, and the vectors for these tokens are concatenated, preserving the ordering:

$$c_t = x_{t-d_w} : ... : x_{t+d_w} \quad (2)$$

where $x_1 : x_2$ is used as notation for vector concatenation of x_1 and x_2. The combined vector is

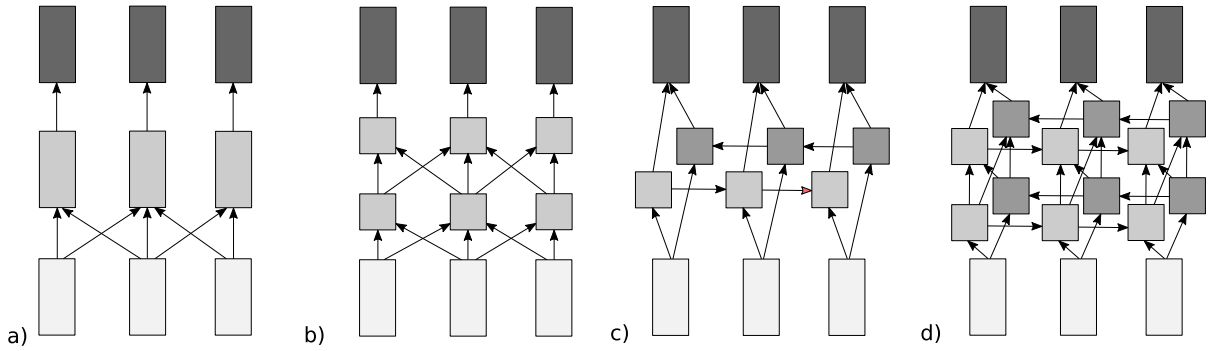


Figure 1: Alternative neural composition architectures for error detection. a) Convolutional network b) Deep convolutional network c) Recurrent bidirectional network d) Deep recurrent bidirectional network. The bottom layers are embeddings for individual tokens. The middle layers are context-dependent representations, built using different composition functions. The top layers are softmax output layers, predicting a label distribution for every input token.

then passed through a non-linear layer to produce the hidden representation:

$$h_t = tanh(W_c c_t) \tag{3}$$

The **deep convolutional** network (Figure 1b) adds an extra convolutional layer to the architecture, using the first layer as input. It creates convolutions of convolutions, thereby capturing more complex higher-order features from the dataset.

In a **recurrent** neural network (RNN), each hidden representation is calculated based on the current token embedding and the hidden vector at the previous time step:

$$h_t = f(Wx_t + Vh_{t-1}) \tag{4}$$

where $f(z)$ is a nonlinear function, such as the sigmoid function. Instead of a fixed context window, information is passed through the sentence using a recursive function and the network is able to learn which patterns to disregard or pass forward. This recurrent network structure is referred to as an Elman-type network, after Elman (1990).

The **bidirectional RNN** (Figure 1c) consists of two recurrent components, moving in opposite directions through the sentence. While the unidirectional version takes into account only context on the left of the target token, the bidirectional version recursively builds separate context representations from either side of the target token. The left and right context are then concatenated and used as the hidden representation:

$$h_t^{\rightarrow} = f(W_r x_t + V_r h_{t-1}^{\rightarrow}) \tag{5}$$

$$h_t^{\leftarrow} = f(W_l x_t + V_l h_{t+1}^{\leftarrow}) \tag{6}$$

$$h_t = h_t^{\rightarrow} : h_t^{\leftarrow} \tag{7}$$

Recurrent networks have been shown to perform well on the task of language modeling (Mikolov et al., 2011; Chelba et al., 2013), where they learn an incremental composition function for predicting the next token in the sequence. However, while language models can estimate the probability of each token, they are unable to differentiate between infrequent and incorrect token sequences. For error detection, the composition function needs to learn to identify semantic anomalies or ungrammatical combinations, independent of their frequency. The bidirectional model provides extra information, as it allows the network to use context on both sides of the target token.

Irsoy and Cardie (2014) created an extension of this architecture by connecting together multiple layers of bidirectional Elman-type recurrent network modules. This **deep bidirectional RNN** (Figure 1d) calculates a context-dependent representation for each token using a bidirectional RNN, and then uses this as input to another bidirectional RNN. The multi-layer structure allows the model to learn more complex higher-level features and effectively perform multiple recurrent passes through the sentence.

The long-short term memory (**LSTM**) (Hochreiter and Schmidhuber, 1997) is an advanced alternative to the Elman-type networks that has recently become increasingly popular. It uses

two separate hidden vectors to pass information between different time steps, and includes gating mechanisms for modulating its own output. LSTMs have been successfully applied to various tasks, such as speech recognition (Graves et al., 2013), machine translation (Luong et al., 2015), and natural language generation (Wen et al., 2015).

Two sets of gating values (referred to as the *input* and *forget* gates) are first calculated based on the previous states of the network:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + V_f c_{t-1} + b_i) \quad (8)$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1} + b_f) \quad (9)$$

where x_t is the current input, h_{t-1} is the previous hidden state, b_i and b_f are biases, c_{t-1} is the previous internal state (referred to as the *cell*), and σ is the logistic function. The new internal state is calculated based on the current input and the previous hidden state, and then interpolated with the previous internal state using f_t and i_t as weights:

$$\widetilde{c_t} = tanh(W_c x_t + U_c h_{t-1} + b_c) \quad (10)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \widetilde{c_t} \quad (11)$$

where $\odot$ is element-wise multiplication. Finally, the hidden state is calculated by passing the internal state through a $tanh$ nonlinearity, and weighting it with o_t. The values of o_t are conditioned on the new internal state (c_t), as opposed to the previous one (c_{t-1}):

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t + b_o) \quad (12)$$

$$h_t = o_t \odot tanh(c_t) \quad (13)$$

Because of the linear combination in equation (11), the LSTM is less susceptible to vanishing gradients over time, thereby being able to make use of longer context when making predictions. In addition, the network learns to modulate itself, effectively using the gates to predict which operation is required at each time step, thereby incorporating higher-level features.

In order to use this architecture for error detection, we create a **bidirectional LSTM**, making use of the advanced features of LSTM and incorporating context on both sides of the target token. In addition, we experiment with a **deep bidirectional LSTM**, which includes two consecutive layers of bidirectional LSTMs, modeling even more complex features and performing multiple passes through the sentence.

For comparison with non-neural models, we also report results using **CRFs** (Lafferty et al., 2001), which are a popular choice for sequence labeling tasks. We trained the CRF++ [1] implementation on the same dataset, using as features unigrams, bigrams and trigrams in a 7-word window surrouding the target word (3 words before and after). The predicted label is also conditioned on the previous label in the sequence.

4 Experiments

We evaluate the alternative network structures on the publicly released First Certificate in English dataset (FCE-public, Yannakoudakis et al. (2011)). The dataset contains short texts, written by learners of English as an additional language in response to exam prompts eliciting free-text answers and assessing mastery of the upper-intermediate proficiency level. The texts have been manually error-annotated using a taxonomy of 77 error types. We use the released test set for evaluation, containing 2,720 sentences, leaving 30,953 sentences for training. We further separate 2,222 sentences from the training set for development and hyper-parameter tuning.

The dataset contains manually annotated error spans of various types of errors, together with their suggested corrections. We convert this to a token-level error detection task by labeling each token inside the error span as being incorrect. In order to capture errors involving missing words, the error label is assigned to the token immediately after the incorrect gap – this is motivated by the intuition that while this token is correct when considered in isolation, it is incorrect in the current context, as another token should have preceeded it.

As the main evaluation measure for error detection we use $F_{0.5}$, which was also the measure adopted in the CoNLL-14 shared task on error correction (Ng et al., 2014). It combines both precision and recall, while assigning twice as much weight to precision, since accurate feedback is often more important than coverage in error detection applications (Nagata and Nakatani, 2010). Following Chodorow et al. (2012), we also report raw counts for predicted and correct tokens. Related evaluation measures, such as the M^2-scorer (Ng et al., 2014) and the I-measure (Felice and

[1] https://taku910.github.io/crfpp/

	Development			Test				
	P	R	$F_{0.5}$	predicted	correct	P	R	$F_{0.5}$
CRF	62.2	13.6	36.3	914	516	56.5	8.2	25.9
CNN	52.4	24.9	42.9	3518	1620	46.0	25.7	39.8
Deep CNN	48.4	26.2	41.4	3992	1651	41.4	26.2	37.1
Bi-RNN	**63.9**	18.0	42.3	2333	1196	**51.3**	19.0	38.2
Deep Bi-RNN	60.3	17.6	40.6	2543	1255	49.4	19.9	38.1
Bi-LSTM	54.5	**28.2**	**46.0**	3898	1798	46.1	**28.5**	**41.1**
Deep Bi-LSTM	56.7	21.3	42.5	2822	1359	48.2	21.6	38.6

Table 1: Performance of the CRF and alternative neural network structures on the public FCE dataset for token-level error detection in learner writing.

Briscoe, 2015), require the system to propose a correction and are therefore not directly applicable on the task of error detection.

During the experiments, the input text was lowercased and all tokens that occurred less than twice in the training data were represented as a single *unk* token. Word embeddings were set to size 300 and initialised using the publicly released pretrained Word2Vec vectors (Mikolov et al., 2013). The convolutional networks use window size 3 on either side of the target token and produce a 300-dimensional context-dependent vector. The recurrent networks use hidden layers of size 200 in either direction. We also added an extra hidden layer of size 50 between each of the composition functions and the output layer – this allows the network to learn a separate non-linear transformation and reduces the dimensionality of the compositional vectors. The parameters were optimised using gradient descent with initial learning rate 0.001, the ADAM algorithm (Kingma and Ba, 2015) for dynamically adapting the learning rate, and batch size of 64 sentences. $F_{0.5}$ on the development set was evaluated at each epoch, and the best model was used for final evaluations.

5 Results

Table 1 contains results for experiments comparing different composition architectures on the task of error detection. The CRF has the lowest $F_{0.5}$ score compared to any of the neural models. It memorises frequent error sequences with high precision, but does not generalise sufficiently, resulting in low recall. The ability to condition on the previous label also does not provide much help on this task – there are only two possible labels and the errors are relatively sparse.

The architecture using convolutional networks performs well and achieves the second-highest result on the test set. It is designed to detect error patterns from a fixed window of 7 words, which is large enough to not require the use of more advanced composition functions. In contrast, the performance of the bidirectional recurrent network (Bi-RNN) is somewhat lower, especially on the test set. In Elman-type recurrent networks, the context signal from distant words decreases fairly rapidly due to the sigmoid activation function and diminishing gradients. This is likely why the Bi-RNN achieves the highest precision of all systems – the predicted label is mostly influenced by the target token and its immediate neighbours, allowing the network to only detect short high-confidence error patterns. The convolutional network, which uses 7 context words with equal attention, is able to outperform the Bi-RNN despite the fixed-size context window.

The best overall result and highest $F_{0.5}$ is achieved by the bidirectional LSTM composition model (Bi-LSTM). This architecture makes use of the full sentence for building context vectors on both sides of the target token, but improves on Bi-RNN by utilising a more advanced composition function. Through the application of a linear update for the internal cell representation, the LSTM is able to capture dependencies over longer distances. In addition, the gating functions allow it to adaptively decide which information to include in the hidden representations or output for error detection.

We found that using multiple layers of compositional functions in a deeper network gave comparable or slightly lower results for all the composition architectures. This is in contrast to Ir-

Training data	Dev $F_{0.5}$	Test $F_{0.5}$
FCE-public	46.0	41.1
+NUCLE	39.0	41.0
+IELTS	45.6	50.7
+FCE	57.2	61.1
+CPE	59.0	62.1
+CAE	**60.7**	**64.3**

Table 2: Results on the public FCE test set when incrementally providing more training data to the error detection model.

soy and Cardie (2014), who experimented with Elman-type networks and found some improvements using multiple layers of Bi-RNNs. The differences can be explained by their task benefiting from alternative features: the evaluation was performed on opinion mining where most target sequences are longer phrases that need to be identified based on their semantics, whereas many errors in learner writing are short and can only be identified by a contextual mismatch. In addition, our networks contain an extra hidden layer before the output, which allows the models to learn higher-level representations without adding complexity through an extra compositional layer.

6 Additional Training Data

There are essentially infinitely many ways of committing errors in text and introducing additional training data should alleviate some of the problems with data sparsity. We experimented with incrementally adding different error-tagged corpora into the training set and measured the resulting performance. This allows us to provide some context to the results obtained by using each of the datasets, and gives us an estimate of how much annotated data is required for optimal performance on error detection. The datasets we consider are as follows:

- FCE-public – the publicly released subset of FCE (Yannakoudakis et al., 2011), as described in Section 4.
- NUCLE – the NUS Corpus of Learner English (Dahlmeier et al., 2013), used as the main training set for CoNLL shared tasks on error correction.
- IELTS – a subset of the IELTS examination dataset extracted from the Cambridge Learner Corpus (CLC, Nicholls (2003)), containing 68,505 sentences from all proficiency levels, also used by Felice et al. (2014).
- FCE – a larger selection of FCE texts from the CLC, containing 323,192 sentences.
- CPE – essays from the proficient examination level in the CLC, containing 210,678 sentences.
- CAE – essays from the advanced examination level in the CLC, containing 219,953 sentences.

Table 2 contains results obtained by incrementally adding training data to the Bi-LSTM model. We found that incorporating the NUCLE dataset does not improve performance over using only the FCE-public dataset, which is likely due to the two corpora containing texts with different domains and writing styles. The texts in FCE are written by young intermediate students, in response to prompts eliciting letters, emails and reviews, whereas NUCLE contains mostly argumentative essays written by advanced adult learners. The differences in the datasets offset the benefits from additional training data, and the performance remains roughly the same.

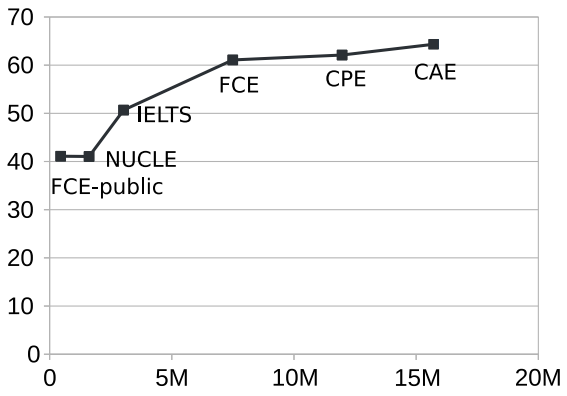


Figure 2: $F_{0.5}$ measure on the public FCE test set, as a function of the total number of tokens in the training set.

In contrast, substantial improvements are obtained when introducing the IELTS and FCE datasets, with each of them increasing the $F_{0.5}$ score by roughly 10%. The IELTS dataset contains essays from all proficiency levels, and FCE from mid-level English learners, which provides the model with a distribution of 'average' errors to learn from. Adding even more training data from

		Annotation 1				Annotation 2			
	predicted	correct	P	R	$F_{0.5}$	correct	P	R	$F_{0.5}$
Annotator 1	2992	-	-	-	-	1800	60.2	42.9	55.7
Annotator 2	4199	1800	42.9	60.2	45.5	-	-	-	-
CAMB	2170	731	33.7	**24.4**	31.3	1052	48.5	**25.1**	40.8
CUUI	1582	550	34.8	18.4	29.5	755	47.7	18.0	35.9
AMU	1260	479	38.0	16.0	29.8	643	51.0	15.3	34.8
P1+P2+S1+S2	887	388	**43.7**	13.0	29.7	535	**60.3**	12.7	34.5
Bi-LSTM (FCE-public)	4449	683	15.4	22.8	16.4	1052	23.6	25.1	23.9
Bi-LSTM (full)	1540	627	40.7	21.0	**34.3**	911	59.2	21.7	**44.0**

Table 3: Error detection results on the two official annotations for the CoNLL-14 shared task test dataset.

high-proficiency essays in CPE and CAE only provides minor further improvements.

Figure 2 also shows $F_{0.5}$ on the FCE-public test set as a function of the total number of tokens in the training data. The optimal trade-off between performance and data size is obtained at around 8 million tokens, after introducing the FCE dataset.

7 CoNLL-14 Shared Task

The CoNLL-14 shared task (Ng et al., 2014) focussed on automatically correcting errors in learner writing. The NUCLE dataset was provided as the main training dataset, but participants were allowed to include other annotated corpora and external resources. For evaluation, 25 students were recruited to each write two new essays, which were then annotated by two experts.

We used the same methods from Section 4 for converting the shared task annotation to a token-level labeling task in order to evaluate the models on error detection. In addition, the correction outputs of all the participating systems were made available online, therefore we are able to report their performance on this task. In order to convert their output to error detection labels, the corrected sentences were aligned with the original input using Levenshtein distance, and any changes proposed by the system resulted in the corresponding source words being labeled as errors.

The results on the two annotations of the shared task test data can be seen in Table 3. We first evaluated each of the human annotators with respect to the other, in order to estimate the upper bound on this task. The average $F_{0.5}$ of roughly 50% shows that the task is difficult and even human experts have a rather low agreement. It has been shown before that correcting grammatical errors is highly subjective (Bryant and Ng, 2015), but these results indicate that trained annotators can disagree even on the number and location of errors.

In the same table, we provide error detection results for the top 3 participants in the shared task: CAMB (Felice et al., 2014), CUUI (Rozovskaya et al., 2014), and AMU (Junczys-Dowmunt and Grundkiewicz, 2014). They each preserve their relative ranking also in the error detection evaluation. The CAMB system has a lower precision but the highest recall, also resulting in the highest $F_{0.5}$. CUUI and AMU are close in performance, with AMU having slightly higher precision.

After the official shared task, Susanto et al. (2014) published a system which combines several alternative models and outperforms the shared task participants when evaluated on error correction. However, on error detection it receives lower results, ranking 3rd and 4th when evaluated on $F_{0.5}$ (P1+P2+S1+S2 in Table 3). The system has detected a small number of errors with high precision, and does not reach the highest $F_{0.5}$.

Finally, we present results for the Bi-LSTM sequence labeling system for error detection. Using only FCE-public for training, the overall performance is rather low as the training set is very small and contains texts from a different domain. However, these results show that the model behaves as expected – since it has not seen similar language during training, it labels a very large portion of tokens as errors. This indicates that the network is trying to learn correct language constructions from the limited data and classifies unseen structures as errors, as opposed to simply memorising error sequences from the training data.

When trained on all the datasets from Section 6, the model achieves the highest $F_{0.5}$ of all systems on both of the CoNLL-14 shared task test annotations, with an absolute improvement of 3% over the previous best result. It is worth noting that the full Bi-LSTM has been trained on more data than the other CoNLL contestants. However, as the shared task systems were not restricted to the NUCLE training set, all the submissions also used differing amounts of training data from various sources. In addition, the CoNLL systems are mostly combinations of many alternative models: the CAMB system is a hybrid of machine translation, a rule-based system, and a language model re-ranker; CUUI consists of different classifiers for each individual error type; and P1+P2+S1+S2 is a combination of four different error correction systems. In contrast, the Bi-LSTM is a single model for detecting all error types, and therefore represents a more scalable data-driven approach.

8 Essay Scoring

In this section, we perform an extrinsic evaluation of the efficacy of the error detection system and examine the extent to which it generalises at higher levels of granularity on the task of automated essay scoring. More specifically, we replicate experiments using the *text-level* model described by Andersen et al. (2013), which is currently deployed in a self-assessment and tutoring system (SAT), an online automated writing feedback tool actively used by language learners.[2]

The SAT system predicts an overall score for a given text, which provides a holistic assessment of linguistic competence and language proficiency. The authors trained a supervised ranking perceptron model on the FCE-public dataset, using features such as error-rate estimates from a language model and various lexical and grammatical properties of text (e.g., word n-grams, part-of-speech n-grams and phrase-structure rules). We replicate this experiment and add the average probability of each token in the essay being correct, according to the error detection model, as an additional feature for the scoring framework. The system was then retrained on FCE-public and evaluated on correctly predicting the assigned essay score. Table 4 presents the experimental results.

The human performance on the test set is calculated as the average inter-annotator correlation on the same data, and the existing SAT system has demonstrated levels of performance that are very close to that of human assessors. Nevertheless, the Bi-LSTM model trained only on FCE-public complements the existing features, and the combined model achieves an absolute improvement of around 1% percent, corresponding to 20-31% relative error reduction with respect to the human performance. Even though the Bi-LSTM is trained on the same dataset and the SAT system already includes various linguistic features for capturing errors, our error detection model manages to further improve its performance.

	r	ρ
Human annotators	79.6	79.2
SAT	75.1	76.0
SAT + Bi-LSTM (FCE-public)	76.0	77.0
SAT + Bi-LSTM (full)	**78.0**	**79.9**

Table 4: Pearson's correlation r and Spearman's correlation ρ on the public FCE test set on the task of automated essay scoring.

When the Bi-LSTM is trained on all the available data from Section 6, the combination achieves further substantial improvements. The relative error reduction on Pearson's correlation is 64%, and the system actually outperforms human annotators on Spearman's correlation.

9 Conclusions

In this paper, we presented the first experiments using neural network models for the task of error detection in learner writing. Six alternative compositional network architectures for modeling context were evaluated. Based on the findings, we propose a novel error detection framework using token-level embeddings, bidirectional LSTMs for context representation, and a multi-layer architecture for learning more complex features. This structure allows the model to classify each token as being correct or incorrect, using the full sentence as context. The self-modulation architecture of LSTMs was also shown to be beneficial, as it allows the network to learn more advanced composition rules and remember dependencies over longer distances.

Substantial performance improvements were achieved by training the best model on additional

[2] http://www.cambridgeenglish.org/learning-english/free-resources/write-and-improve/

datasets. We found that the largest benefit was obtained from training on 8 million tokens of text from learners with varying levels of language proficiency. In contrast, including even more data from higher-proficiency learners gave marginal further improvements. As part of future work, it would be beneficial to investigate the effect of automatically generated training data for error detection (e.g., Rozovskaya and Roth (2010)).

We evaluated the performance of existing error correction systems from CoNLL-14 on the task of error detection. The experiments showed that success on error correction does not necessarily mean success on error detection, as the current best correction system (P1+P2+S1+S2) is not the same as the best shared task detection system (CAMB). In addition, the neural sequence tagging model, specialised for error detection, was able to outperform all other participating systems.

Finally, we performed an extrinsic evaluation by incorporating probabilities from the error detection system as features in an essay scoring model. Even without any additional data, the combination further improved performance which is already close to the results from human annotators. In addition, when the error detection model was trained on a larger training set, the essay scorer was able to exceed human-level performance.

Acknowledgments

We would like to thank Prof Ted Briscoe and the reviewers for providing useful feedback.

References

Øistein E. Andersen, Helen Yannakoudakis, Fiona Barker, and Tim Parish. 2013. Developing and testing a self-assessment and tutoring system. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *International Conference on Learning Representations*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A Neural Probabilistic Language Model Yoshua. *Journal of Machine Learning Research*, 3.

Christopher Bryant and Hwee Tou Ng. 2015. How Far are We from Fully Automatic High Quality Grammatical Error Correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*.

Ciprian Chelba, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. In *arXiv preprint*.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*.

Martin Chodorow and Claudia Leacock. 2000. An unsupervised method for detecting grammatical errors. In *Proceedings of the first conference on North American chapter of the Association for Computational Linguistics*.

Martin Chodorow, Joel R. Tetreault, and Na-Rae Han. 2007. Detection of grammatical errors involving prepositions. In *Proceedings of the 4th ACL-SIGSEM Workshop on Prepositions*.

Martin Chodorow, Markus Dickinson, Ross Israel, and Joel Tetreault. 2012. Problems in evaluating grammatical error detection systems. In *COLING 2012*.

George. E. Dahl, Dong Yu, Li Deng, and Alex Acero. 2012. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*.

Robert Dale and Adam Kilgarriff. 2011. Helping Our Own: The HOO 2011 Pilot Shared Task. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

Robert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the Preposition and Determiner Error Correction Shared Task. In *The Seventh Workshop on Building Educational Applications Using NLP*.

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

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *The 2015 Annual Conference of the North American Chapter of the ACL*.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. *Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014)*.

Michael Gamon. 2011. High-Order Sequence Modeling for Language Learner Error Detection. *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*.

Alex Graves, Navdeep Jaitly, and Abdel Rahman Mohamed. 2013. Hybrid speech recognition with Deep Bidirectional LSTM. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU 2013)*.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2004. Detecting Errors in English Article Usage with a Maximum Entropy Classifier Trained on a Large, Diverse Corpus. *Proceedings of the 4th International Conference on Language Resources and Evaluation*.

Na-Rae Han, Martin Chodorow, and Claudia Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Natural Language Engineering*, 12.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural Computation*, 9.

Ozan Irsoy and Claire Cardie. 2014. Opinion Mining with Deep Recurrent Neural Networks. In *EMNLP-2014*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2014. The AMU System in the CoNLL-2014 Shared Task: Grammatical Error Correction by Data-Intensive and Feature-Rich Statistical Machine Translation. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014)*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*.

Diederik P. Kingma and Jimmy Lei Ba. 2015. Adam: a Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Ekaterina Kochmar and Ted Briscoe. 2014. Detecting Learner Errors in the Choice of Content Words Using Compositional Distributional Semantics. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*. Citeseer.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel R. Tetreault. 2014. *Automated Grammatical Error Detection for Language Learners: Second Edition*.

John Lee and Stephanie Seneff. 2008. Correcting misuse of verb forms. In *Proceedings of the 46th Annual Meeting of the ACL*.

Mnih-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Tomáš Mikolov, Stefan Kombrink, Anoop Deoras, Lukáš Burget, and Jan Černocký. 2011. RNNLM-Recurrent neural network language modeling toolkit. In *ASRU 2011 Demo Session*.

Tomáš Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)*.

Ryo Nagata and Kazuhide Nakatani. 2010. Evaluating performance of grammatical error detection to maximize learning effect. *Coling 2010: Poster Volume*.

Hwee Tou Ng, Yuanbin Wu, and Christian Hadiwinoto. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. *Computational Natural Language Learning (CoNLL), Shared Task*.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

Diane Nicholls. 2003. The Cambridge Learner Corpus - error coding and analysis for lexicography and ELT. *Proceedings of the Corpus Linguistics 2003 Conference*.

Alla Rozovskaya and Dan Roth. 2010. Training Paradigms for Correcting Errors in Grammar and Usage. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*.

Alla Rozovskaya, Mark Sammons, Joshua Gioja, and Dan Roth. 2011. University of Illinois System in HOO Text Correction Shared Task. *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG)*.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. 2013. The University of Illinois System in the CoNLL-2013 Shared Task. *In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, Dan Roth, and Nizar Habash. 2014. The Illinois-Columbia System in the CoNLL-2014 Shared Task. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task (CoNLL-2014)*.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System Combination for Grammatical Error Correction. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014)*.

Joel R. Tetreault and Martin Chodorow. 2008. The Ups and Downs of Preposition Error Detection in ESL Writing. *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. *EMNLP 2015*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

Neural Semantic Role Labeling with Dependency Path Embeddings

Michael Roth and **Mirella Lapata**
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
{mroth,mlap}@inf.ed.ac.uk

Abstract

This paper introduces a novel model for semantic role labeling that makes use of neural sequence modeling techniques. Our approach is motivated by the observation that complex syntactic structures and related phenomena, such as nested subordinations and nominal predicates, are not handled well by existing models. Our model treats such instances as subsequences of lexicalized dependency paths and learns suitable embedding representations. We experimentally demonstrate that such embeddings can improve results over previous state-of-the-art semantic role labelers, and showcase qualitative improvements obtained by our method.

1 Introduction

The goal of *semantic role labeling* (SRL) is to identify and label the arguments of semantic predicates in a sentence according to a set of predefined relations (e.g., "who" did "what" to "whom"). Semantic roles provide a layer of abstraction beyond syntactic dependency relations, such as *subject* and *object*, in that the provided labels are insensitive to syntactic alternations and can also be applied to nominal predicates. Previous work has shown that semantic roles are useful for a wide range of natural language processing tasks, with recent applications including statistical machine translation (Aziz et al., 2011; Xiong et al., 2012), plagiarism detection (Osman et al., 2012; Paul and Jamal, 2015), and multi-document abstractive summarization (Khan et al., 2015).

The task of semantic role labeling (SRL) was pioneered by Gildea and Jurafsky (2002). In their work, features based on syntactic constituent trees were identified as most valuable for labeling predicate-argument relationships. Later work confirmed the importance of syntactic parse features (Pradhan et al., 2005; Punyakanok et al., 2008) and found that dependency parse trees provide a better form of representation to assign role labels to arguments (Johansson and Nugues, 2008).

System	Analysis
mate-tools	*He had [trouble$_{A0}$] **raising** [funds$_{A1}$].
mateplus	*He had [trouble$_{A0}$] **raising** [funds$_{A1}$].
TensorSRL	*He had trouble **raising** [funds$_{A1}$].
easySRL	*He had trouble **raising** [funds$_{A1}$].
This work	[He$_{A0}$] had trouble **raising** [funds$_{A1}$].

Table 1: Outputs of SRL systems for the sentence *He had trouble raising funds.* Arguments of **raise** are shown with predicted roles as defined in PropBank (A0: getter of money; A1: money). Asterisks mark flawed analyses that miss the argument *He*.

Most semantic role labeling approaches to date rely heavily on lexical and syntactic indicator features. Through the availability of large annotated resources, such as PropBank (Palmer et al., 2005), statistical models based on such features achieve high accuracy. However, results often fall short when the input to be labeled involves instances of linguistic phenomena that are relevant for the labeling decision but appear infrequently at training time. Examples include control and raising verbs, nested conjunctions or other recursive structures, as well as rare nominal predicates. The difficulty lies in that simple lexical and syntactic indicator features are not able to model interactions triggered by such phenomena. For instance, con-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1192–1202,
Berlin, Germany, August 7-12, 2016.

sider the sentence *He had trouble raising funds* and the analyses provided by four publicly available tools in Table 1 (mate-tools, Björkelund et al. (2010); mateplus, Roth and Woodsend (2014); TensorSRL, Lei et al. (2015); and easySRL, Lewis et al. (2015)). Despite all systems claiming state-of-the-art or competitive performance, none of them is able to correctly identify *He* as the agent argument of the predicate *raise*. Given the complex dependency path relation between the predicate and its argument, none of the systems actually identifies *He* as an argument at all.

In this paper, we develop a new neural network model that can be applied to the task of semantic role labeling. The goal of this model is to better handle control predicates and other phenomena that can be observed from the dependency structure of a sentence. In particular, we aim to model the semantic relationships between a predicate and its arguments by analyzing the dependency path between the predicate word and each argument head word. We consider lexicalized paths, which we decompose into sequences of individual items, namely the words and dependency relations on a path. We then apply long-short term memory networks (Hochreiter and Schmidhuber, 1997) to find a recurrent composition function that can reconstruct an appropriate representation of the full path from its individual parts (Section 2). To ensure that representations are indicative of semantic relationships, we use semantic roles as target labels in a supervised setting (Section 3).

By modeling dependency paths as sequences of words and dependencies, we implicitly address the data sparsity problem. This is the case because we use single words and individual dependency relations as the basic units of our model. In contrast, previous SRL work only considered full syntactic paths. Experiments on the CoNLL-2009 benchmark dataset show that our model is able to outperform the state-of-the-art in English (Section 4), and that it improves SRL performance in other languages, including Chinese, German and Spanish (Section 5).

2 Dependency Path Embeddings

In the context of neural networks, the term *embedding* refers to the output of a function f within the network, which transforms an arbitrary input into a real-valued vector output. Word embeddings, for instance, are typically computed by forwarding a one-hot word vector representation from the input layer of a neural network to its first hidden layer, usually by means of matrix multiplication and an optional non-linear function whose parameters are learned during neural network training.

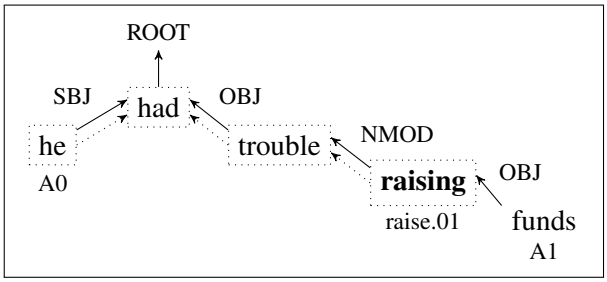


Figure 1: Dependency path (dotted) between the predicate *raising* and the argument *he*.

Here, we seek to compute real-valued vector representations for *dependency paths* between a pair of words $\langle w_i, w_j \rangle$. We define a dependency path to be the *sequence* of nodes (representing words) and edges (representing relations between words) to be traversed on a dependency parse tree to get from node w_i to node w_j. In the example in Figure 1, the dependency path from *raising* to *he* is $raising \xrightarrow{\text{NMOD}} trouble \xrightarrow{\text{OBJ}} had \xleftarrow{\text{SBJ}} he$.

Analogously to how word embeddings are computed, the simplest way to embed paths would be to represent each sequence as a one-hot vector. However, this is suboptimal for two reasons: Firstly, we expect only a subset of dependency paths to be attested frequently in our data and therefore many paths will be too sparse to learn reliable embeddings for them. Secondly, we hypothesize that dependency paths which share the same words, word categories or dependency relations should impact SRL decisions in similar ways. Thus, the words and relations on the path should drive representation learning, rather than the full path on its own. The following sections describe how we address representation learning by means of modeling dependency paths as sequences of items in a recurrent neural network.

2.1 Recurrent Neural Networks

The recurrent model we use in this work is a variant of the long-short term memory (LSTM) network. It takes a sequence of items $X = x_1, ..., x_n$ as input, recurrently processes each item $x_t \in X$ at a time, and finally returns one embedding state $\mathbf{e}_n$ for the complete input sequence. For each time

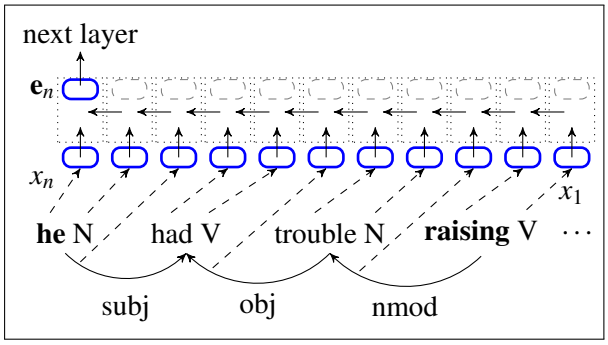


Figure 2: Example input and embedding computation for the path from *raising* to *he*, given the sentence *he had trouble raising funds*. LSTM time steps are displayed from right to left.

step t, the LSTM model updates an internal memory state $\mathbf{m}_t$ that depends on the current input as well as the previous memory state $\mathbf{m}_{t-1}$. In order to capture long-term dependencies, a so-called gating mechanism controls the extent to which each component of a memory cell state will be modified. In this work, we employ input gates $\mathbf{i}$, output gates $\mathbf{o}$ and (optional) forget gates $\mathbf{f}$. We formalize the state of the network at each time step t as follows:

$$\mathbf{i}_t = \sigma([\mathbf{W}^{\mathrm{mi}}\mathbf{m}_{t-1}] + \mathbf{W}^{\mathrm{xi}}x_t + \mathbf{b}^{\mathrm{i}}) \quad (1)$$

$$\mathbf{f}_t = \sigma([\mathbf{W}^{\mathrm{mf}}\mathbf{m}_{t-1}] + \mathbf{W}^{\mathrm{xf}}x_t + \mathbf{b}^{\mathrm{f}}) \quad (2)$$

$$\mathbf{m}_t = \mathbf{i}_t \odot (\mathbf{W}^{\mathrm{xm}}x_t) + \mathbf{f}_t \odot \mathbf{m}_{t-1} + \mathbf{b}^{\mathrm{m}} \quad (3)$$

$$\mathbf{o}_t = \sigma([\mathbf{W}^{\mathrm{mo}}\mathbf{m}_t] + \mathbf{W}^{\mathrm{xo}}x_t + \mathbf{b}^{\mathrm{o}}) \quad (4)$$

$$\mathbf{e}_t = \mathbf{o}_t \odot \sigma(\mathbf{m}_t) \quad (5)$$

In each equation, $\mathbf{W}$ describes a matrix of weights to project information between two layers, $\mathbf{b}$ is a layer-specific vector of bias terms, and σ is the logistic function. Superscripts indicate the corresponding layers or gates. Some models described in Section 3 do not make use of forget gates or memory-to-gate connections. In case no forget gate is used, we set $\mathbf{f}_t = \mathbf{1}$. If no memory-to-gate connections are used, the terms in square brackets in (1), (2), and (4) are replaced by zeros.

2.2 Embedding Dependency Paths

We define the *embedding of a dependency path* to be the final memory output state of a recurrent LSTM layer that takes a path as input, with each input step representing a binary indicator for a part-of-speech tag, a word form, or a dependency relation. In the context of semantic role labeling, we define each path as a sequence from a predicate to its potential argument.[1] Specifically, we define the first item x_1 to correspond to the part-of-speech tag of the predicate word w_i, followed by its actual word form, and the relation to the next word w_{i+1}. The embedding of a dependency path corresponds to the state $\mathbf{e}_n$ returned by the LSTM layer after the input of the last item, x_n, which corresponds to the word form of the argument head word w_j. An example is shown in Figure 2.

The main idea of this model and representation is that word forms, word categories and dependency relations can all influence role labeling decisions. The word category and word form of the predicate first determine which roles are plausible and what kinds of path configurations are to be expected. The relations and words seen on the path can then manipulate these expectations. In Figure 2, for instance, the verb *raising* complements the phrase *had trouble*, which makes it likely that the subject *he* is also the logical subject of *raising*.

By using word forms, categories and dependency relations as input items, we ensure that specific words (e.g., those which are part of complex predicates) as well as various relation types (e.g., subject and object) can appropriately influence the representation of a path. While learning corresponding interactions, the network is also able to determine which phrases and dependency relations might not influence a role assignment decision (e.g., coordinations).

2.3 Joint Embedding and Feature Learning

Our SRL model consists of four components depicted in Figure 3: (1) an LSTM component takes lexicalized dependency paths as input, (2) an additional input layer takes binary features as input, (3) a hidden layer combines dependency path embeddings and binary features using rectified linear units, and (4) a softmax classification layer produces output based on the hidden layer state as input. We therefore learn path embeddings jointly with feature detectors based on traditional, binary indicator features.

Given a dependency path X, with steps $x_k \in \{x_1, \ldots, x_n\}$, and a set of binary features B as input, we use the LSTM formalization from equations (1–5) to compute the embedding $\mathbf{e}_n$ at time

[1] We experimented with different sequential orders and found this to lead to the best validation set results.

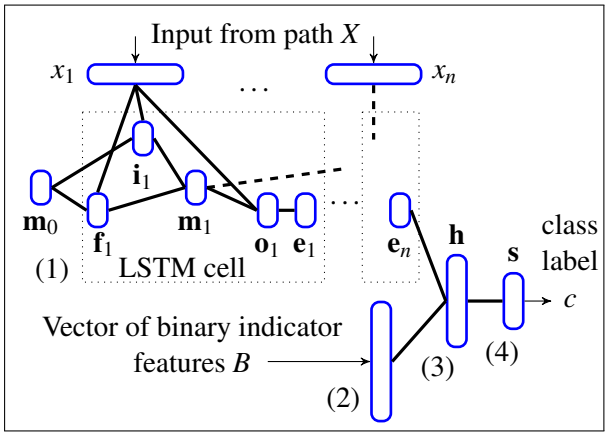


Figure 3: Neural model for joint learning of path embeddings and higher-order features: The path sequence $x_1 \ldots x_n$ is fed into a LSTM layer, a hidden layer $\mathbf{h}$ combines the final embedding $\mathbf{e}_n$ and binary input features B, and an output layer $\mathbf{s}$ assigns the highest probable class label c.

step n and formalize the state of the hidden layer $\mathbf{h}$ and softmax output $\mathbf{s}_c$ for each class category c as follows:

$$\mathbf{h} = \max(0, \mathbf{W}^{\text{Bh}} B + \mathbf{W}^{\text{eh}} \mathbf{e}_n + \mathbf{b}^{\text{h}}) \quad (6)$$

$$\mathbf{s}_c = \frac{\mathbf{W}_c^{\text{es}} \mathbf{e}_n + \mathbf{W}_c^{\text{hs}} \mathbf{h} + \mathbf{b}_c^{\text{s}}}{\Sigma_i (\mathbf{W}_i^{\text{es}} \mathbf{e}_n + \mathbf{W}_i^{\text{hs}} \mathbf{h} + \mathbf{b}_i^{\text{s}})} \quad (7)$$

3 System Architecture

The overall architecture of our SRL system closely follows that of previous work (Toutanova et al., 2008; Björkelund et al., 2009) and is depicted in Figure 4. We use a pipeline that consists of the following steps: predicate identification and disambiguation, argument identification, argument classification, and re-ranking. The neural-network components introduced in Section 2 are used in the last three steps. The following sub-sections describe all components in more detail.

3.1 Predicate Identification and Disambiguation

Given a syntactically analyzed sentence, the first two steps in an end-to-end SRL system are to identify and disambiguate the semantic predicates in the sentence. Here, we focus on verbal and nominal predicates but note that other syntactic categories have also been construed as predicates in the NLP literature (e.g., prepositions; Srikumar and Roth (2013)). For both identification and disambiguation steps, we apply the same logistic re-

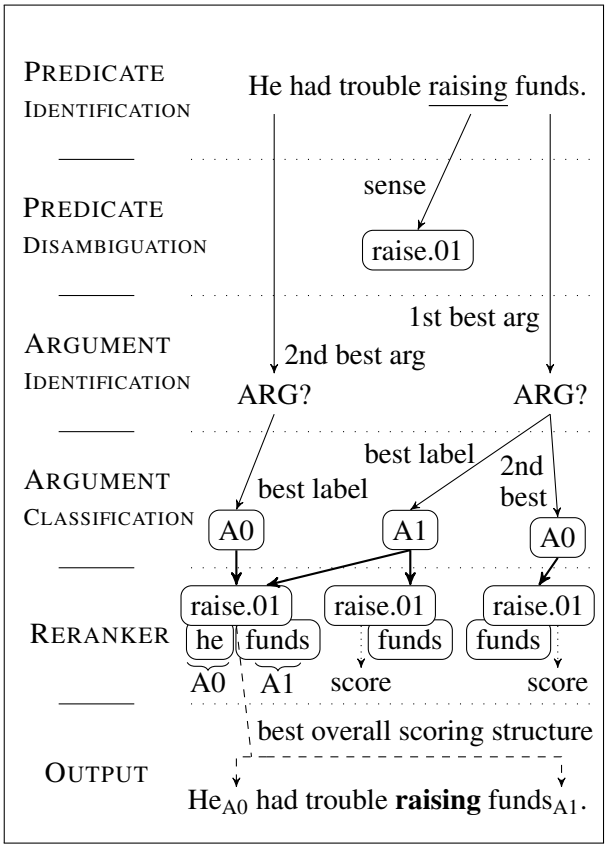


Figure 4: Pipeline architecture of our SRL system.

gression classifiers used in the SRL components of mate-tools (Björkelund et al., 2010). The classifiers for both tasks make use of a range of lexico-syntactic indicator features, including predicate word form, its predicted part-of-speech tag as well as dependency relations to all syntactic children.

3.2 Argument Identification and Classification

Given a sentence and a set of sense-disambiguated predicates in it, the next two steps of our SRL system are to identify all arguments of each predicate and to assign suitable role labels to them. For both steps, we train several LSTM-based neural network models as described in Section 2. In particular, we train separate networks for nominal and verbal predicates and for identification and classification. Following the findings of earlier work (Xue and Palmer, 2004), we assume that different feature sets are relevant for the respective tasks and hence different embedding representations should be learned. As binary input features, we use the following sets from the SRL literature (Björkelund et al., 2010).

Argument labeling step	forget gate	memory→gates	$\|e\|$	$\|h\|$	alpha	dropout rate
Identification (verb)	−	+	25	90	0.0006	0.42
Identification (noun)	−	+	16	125	0.0009	0.25
Classification (verb)	+	−	5	300	0.0155	0.50
Classification (noun)	−	−	88	500	0.0055	0.46

Table 2: Hyperparameters selected for best models and training procedures

Lexico-syntactic features Word form and word category of the predicate and candidate argument; dependency relations from predicate and argument to their respective syntactic heads; full dependency path sequence from predicate to argument.

Local context features Word forms and word categories of the candidate argument's and predicate's syntactic siblings and children words.

Other features Relative position of the candidate argument with respect to the predicate (left, self, right); sequence of part-of-speech tags of all words between the predicate and the argument.

3.3 Reranker

As all argument identification (and classification) decisions are independent of one another, we apply as the last step of our pipeline a global reranker. Given a predicate p, the reranker takes as input the n best sets of identified arguments as well as their n best label assignments and predicts the best overall argument structure. We implement the reranker as a logistic regression classifier, with hidden and embedding layer states of identified arguments as features, offset by the argument label, and a binary label as output (1: best predicted structure, 0: any other structure). At test time, we select the structure with the highest overall score, which we compute as the geometric mean of the global regression and all argument-specific scores.

4 Experiments

In this section, we demonstrate the usefulness of dependency path embeddings for semantic role labeling. Our hypotheses are that (1) modeling dependency paths as sequences will lead to better representations for the SRL task, thus increasing labeling precision overall, and that (2) embeddings will address the problem of data sparsity, leading to higher recall. To test both hypotheses, we experiment on the in-domain and out-of-domain test sets provided in the CoNLL-2009 shared task (Hajič et al., 2009) and compare results of our system, henceforth PathLSTM, with systems that do not involve path embeddings. We compute precision, recall and F_1-score using the official CoNLL-2009 scorer.[2] The code is available at `https://github.com/microth/PathLSTM`.

Model selection We train argument identification and classification models using the XLBP toolkit for neural networks (Monner and Reggia, 2012). The hyperparameters for each step were selected based on the CoNLL 2009 development set. For direct comparison with previous work, we use the same preprocessing models and predicate-specific SRL components as provided with mate-tools (Bohnet, 2010; Björkelund et al., 2010). The types and ranges of hyperparameters considered are as follows: learning rate $\alpha \in [0.00006, 0.3]$, dropout rate $d \in [0.0, 0.5]$, and hidden layer sizes $|e| \in [0, 100]$, $|h| \in [0, 500]$. In addition, we experimented with different gating mechanisms (with/without forget gate) and memory access settings (with/without connections between all gates and the memory layer, cf. Section 2). The best parameters were chosen using the Spearmint hyperparameter optimization toolkit (Snoek et al., 2012), applied for approx. 200 iterations, and are summarized in Table 2.

Results The results of our in- and out-of-domain experiments are summarized in Tables 3 and 5, respectively. We present results for different system configurations: 'local' systems make classification decisions independently, whereas 'global' systems include a reranker or other global inference mechanisms; 'single' refers to one model and 'ensemble' refers to combinations of multiple models.

In the in-domain setting, our PathLSTM model achieves 87.7% (single) and 87.9% (ensemble) F_1-score, outperforming previously published best re-

[2] Some recently proposed SRL models are only evaluated on the CoNLL 2005 and 2012 data sets, which lack nominal predicates or dependency annotations. We do not list any results from those models here.

System (local, single)	P	R	F_1
Björkelund et al. (2010)	87.1	84.5	85.8
Lei et al. (2015)	–	–	86.6
FitzGerald et al. (2015)	–	–	**86.7**
PathLSTM w/o reranker	**88.1**	**85.3**	**86.7**
System (global, single)	P	R	F_1
Björkelund et al. (2010)	88.6	85.2	86.9
Roth and Woodsend (2014)[3]	–	–	86.3
FitzGerald et al. (2015)	–	–	87.3
PathLSTM	**90.0**	**85.5**	**87.7**
System (global, ensemble)	P	R	F_1
FitzGerald et al. 10 models	–	–	87.7
PathLSTM 3 models	**90.3**	**85.7**	**87.9**

Table 3: Results on the CoNLL-2009 in-domain test set. All numbers are in percent.

PathLSTM	P (%)	R (%)	F_1 (%)
w/o path embeddings	65.7	87.3	75.0
w/o binary features	73.2	33.3	45.8

Table 4: Ablation tests in the in-domain setting.

sults by 0.4 and 0.2 percentage points, respectively. At a F_1-score of 86.7%, our local model (using no reranker) reaches the same performance as state-of-the-art local models. Note that differences in results between systems might originate from the application of different preprocessing techniques as each system comes with its own syntactic components. For direct comparison, we evaluate against mate-tools, which use the same preprocessing techniques as PathLSTM. In comparison, we see improvements of +0.8–1.0 percentage points absolute in F_1-score.

In the out-of-domain setting, our system achieves new state-of-the-art results of 76.1% (single) and 76.5% (ensemble) F_1-score, outperforming the previous best system by Roth and Woodsend (2014) by 0.2 and 0.6 absolute points, respectively. In comparison to mate-tools, we observe absolute improvements in F_1-score of +0.4–0.8%.

Discussion To determine the sources of individual improvements, we test PathLSTM models without specific feature types and directly compare PathLSTM and mate-tools, both of which use

[3]Results are taken from Lei et al. (2015).

System (local, single)	P	R	F_1
Björkelund et al. (2010)	75.7	72.2	73.9
Lei et al. (2015)	–	–	**75.6**
FitzGerald et al. (2015)	–	–	75.2
PathLSTM w/o reranker	**76.9**	**73.8**	75.3
System (global, single)	P	R	F_1
Björkelund et al. (2010)	77.9	73.6	75.7
Roth and Woodsend (2014)[3]	–	–	75.9
FitzGerald et al. (2015)	–	–	75.2
PathLSTM	**78.6**	**73.8**	**76.1**
System (global, ensemble)	P	R	F_1
FitzGerald et al. 10 models	–	–	75.5
PathLSTM 3 models	79.7	73.6	**76.5**

Table 5: Results on the CoNLL-2009 out-of-domain test set. All numbers are in percent.

the same preprocessing methods. Table 4 presents in-domain test results for our system when specific feature types are omitted. The overall low results indicate that a combination of dependency path embeddings and binary features is required to identify and label arguments with high precision.

Figure 5 shows the effect of dependency path embeddings at mitigating sparsity: if the path between a predicate and its argument has not been observed at training time or only infrequently, conventional methods will often fail to assign a role. This is represented by the recall curve of mate-tools, which converges to zero for arguments with unseen paths. The higher recall curve for PathLSTM demonstrates that path embeddings can alleviate this problem to some extent. For unseen paths, we observe that PathLSTM improves over mate-tools by an order of magnitude, from 0.9% to 9.6%. The highest absolute gain, from 12.8% to 24.2% recall, can be observed for dependency paths that occurred between 1 and 10 times during training.

Figure 7 plots role labeling performance for sentences with varying number of words. There are two categories of sentences in which the improvements of PathLSTM are most noticeable: Firstly, it better handles short sentences that contain expletives and/or nominal predicates (+0.8% absolute in F_1-score). This is probably due to the fact that our learned dependency path representations are lexicalized, making it possible to model

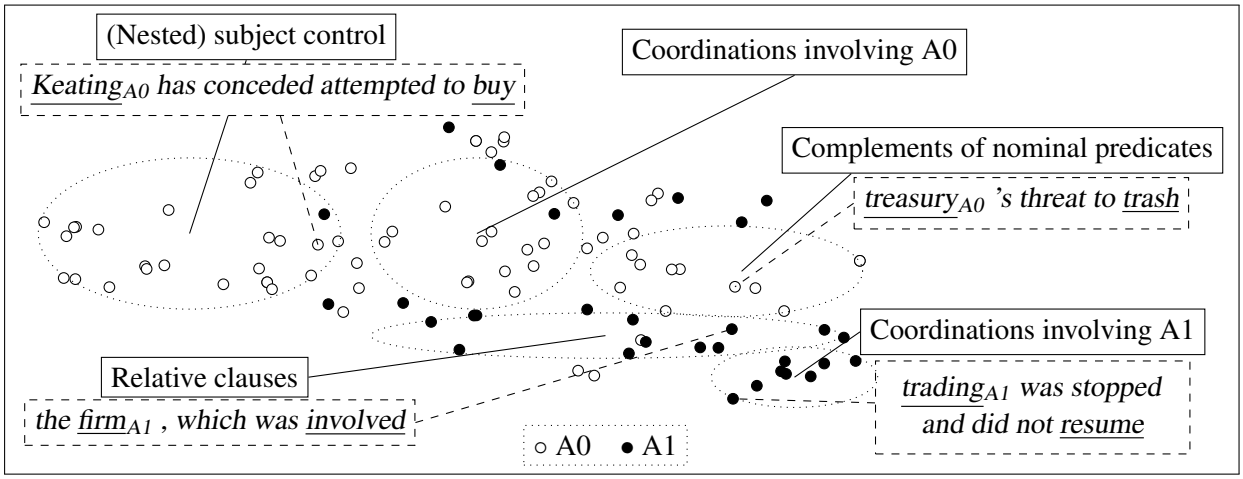


Figure 6: Dots correspond to the path representation of a predicate-argument instance in 2D space. White/black color indicates A0/A1 gold argument labels. Dotted ellipses denote instances exhibiting related syntactic phenomena (see rectangles for a description and dotted rectangles for linguistic examples). Example phrases show actual output produced by PathLSTM (underlined).

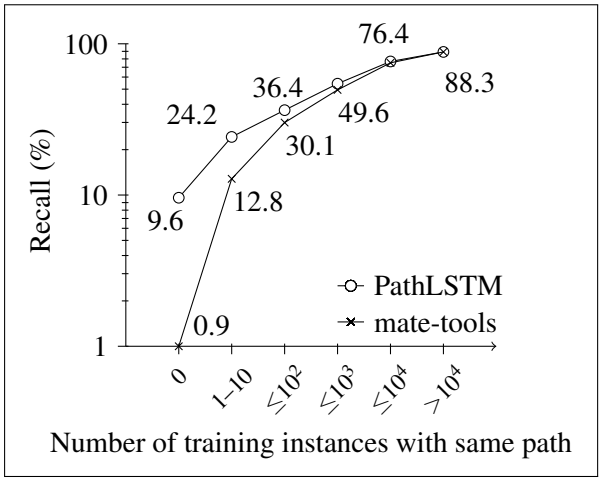


Figure 5: Results on in-domain test instances, grouped by the number of training instances that have an identical (unlexicalized) dependency path.

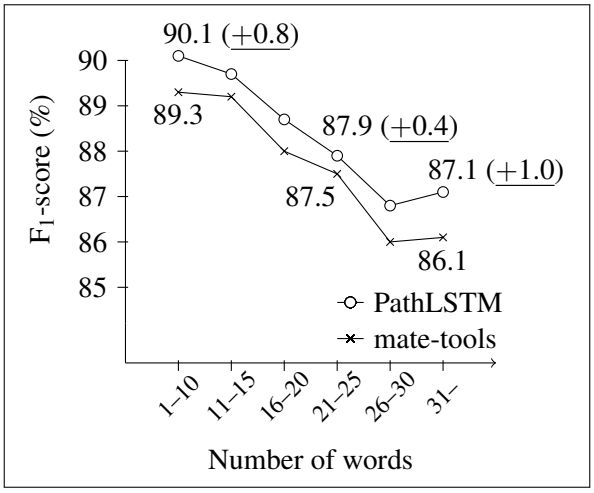


Figure 7: Results by sentence length. Improvements over mate-tools shown in parentheses.

argument structures of different nominals and distinguishing between expletive occurrences of 'it' and other subjects. Secondly, it improves performance on longer sentences (up to +1.0% absolute in F_1-score). This is mainly due to the handling of dependency paths that involve complex structures, such as coordinations, control verbs and nominal predicates.

We collect instances of different syntactic phenomena from the development set and plot the learned dependency path representations in the embedding space (see Figure 6). We obtain a projection onto two dimensions using t-SNE (Van der Maaten and Hinton, 2008). Interestingly, we can see that different syntactic configurations are clustered together in different parts of the space and that most instances of the PropBank roles A0 and A1 are separated. Example phrases in the figure highlight predicate-argument pairs that are correctly labeled by PathLSTM but not by mate-tools. Path embeddings are essential for handling these cases as indicator features do not generalize well enough.

Finally, Table 6 shows results for nominal and verbal predicates as well as for different (gold) role labels. In comparison to mate-tools, we can see that PathLSTM improves precision for all argument types of nominal predicates. For verbal predicates, improvements can be observed in

Predicate POS & Role Label	PathLSTM		Improvement over mate-tools	
	P (%)	R (%)	P (%)	R (%)
verb / A0	90.8	**89.2**	−0.4	+1.8
verb / A1	91.0	**91.9**	+0.0	+1.1
verb / A2	**84.3**	76.9	+1.5	+0.0
verb / AM	**82.2**	72.4	+2.9	−2.0
noun / A0	**86.9**	**78.2**	+0.8	+3.3
noun / A1	**87.5**	**84.4**	+2.6	+2.2
noun / A2	**82.4**	**76.8**	+1.0	+2.1
noun / AM	**79.5**	69.2	+0.9	−2.8

Table 6: Results by word category and role label.

terms of recall of proto-agent (A0) and proto-patient (A1) roles, with slight gains in precision for the A2 role. Overall, PathLSTM does slightly worse with respect to modifier roles, which it labels with higher precision but at the cost of recall.

5 Path Embeddings in other Languages

In this section, we report results from additional experiments on Chinese, German and Spanish data. The underlying question is to which extent the improvements of our SRL system for English also generalize to other languages. To answer this question, we train and test separate SRL models for each language, using the system architecture and hyperparameters discussed in Sections 3 and 4, respectively.

We train our models on data from the CoNLL-2009 shared task, relying on the same features as one of the participating systems (Björkelund et al., 2009), and evaluate with the official scorer. For direct comparison, we rely on the (automatic) syntactic preprocessing information provided with the CoNLL test data and compare our results with the best two systems for each language that make use of the same preprocessing information.

The results, summarized in Table 7, indicate that PathLSTM performs better than the system by Björkelund et al. (2009) in all cases. For German and Chinese, PathLSTM achieves the best overall F_1-scores of 80.1% and 79.4%, respectively.

Chinese	P	R	F_1
PathLSTM	**83.2**	**75.9**	**79.4**
Björkelund et al. (2009)	82.4	75.1	78.6
Zhao et al. (2009)	80.4	75.2	77.7
German	P	R	F_1
PathLSTM	81.8	**78.5**	**80.1**
Björkelund et al. (2009)	81.2	78.3	79.7
Che et al. (2009)	**82.1**	75.4	78.6
Spanish	P	R	F_1
Zhao et al. (2009)	83.1	**78.0**	**80.5**
PathLSTM	**83.2**	77.4	80.2
Björkelund et al. (2009)	78.9	74.3	76.5

Table 7: Results (in percentage) on the CoNLL-2009 test sets for Chinese, German and Spanish.

6 Related Work

Neural Networks for SRL Collobert et al. (2011) pioneered neural networks for the task of semantic role labeling. They developed a feed-forward network that uses a convolution function over windows of words to assign SRL labels. Apart from constituency boundaries, their system does not make use of any syntactic information. Foland and Martin (2015) extended their model and showcased significant improvements when including binary indicator features for dependency paths. Similar features were used by FitzGerald et al. (2015), who include role labeling predictions by neural networks as factors in a global model.

These approaches all make use of binary features derived from syntactic parses either to indicate constituency boundaries or to represent full dependency paths. An extreme alternative has been recently proposed in Zhou and Xu (2015), who model SRL decisions with a multi-layered LSTM network that takes word sequences as input but no syntactic parse information at all.

Our approach falls in between the two extremes: we rely on syntactic parse information but rather than solely making using of sparse binary features, we explicitly model dependency paths in a neural network architecture.

Other SRL approaches Within the SRL literature, recent alternatives to neural network architectures include sigmoid belief networks (Henderson et al., 2013) as well as low-rank tensor models (Lei et al., 2015). Whereas Lei et al. only make use of dependency paths as binary indicator features, Henderson et al. propose a joint model for syntactic and semantic parsing that learns and ap-

plies incremental dependency path representations to perform SRL decisions. The latter form of representation is closest to ours, however, we do not build syntactic parses incrementally. Instead, we take syntactically preprocessed text as input and focus on the SRL task only.

Apart from more powerful models, most recent progress in SRL can be attributed to novel features. For instance, Deschacht and Moens (2009) and Huang and Yates (2010) use latent variables, learned with a hidden markov model, as features for representing words and word sequences. Zapirain et al. (2013) propose different selection preference models in order to deal with the sparseness of lexical features. Roth and Woodsend (2014) address the same problem with word embeddings and compositions thereof. Roth and Lapata (2015) recently introduced features that model the influence of discourse on role labeling decisions.

Rather than coming up with completely new features, in this work we proposed to revisit some well-known features and represent them in a novel way that generalizes better. Our proposed model is inspired both by the necessity to overcome the problems of sparse lexico-syntactic features and by the recent success of SRL models based on neural networks.

Dependency-based embeddings The idea of embedding dependency structures has previously been applied to tasks such as relation classification and sentiment analysis. Xu et al. (2015) and Liu et al. (2015) use neural networks to embed dependency paths between entity pairs. To identify the relation that holds between two entities, their approaches make use of pooling layers that detect parts of a path that indicate a specific relation. In contrast, our work aims at modeling an individual path as a complete sequence, in which every item is of relevance. Tai et al. (2015) and Ma et al. (2015) learn embeddings of dependency structures representing full sentences, in a sentiment classification task. In our model, embeddings are learned jointly with other features, and as a result problems that may result from erroneous parse trees are mitigated.

7 Conclusions

We introduced a neural network architecture for semantic role labeling that jointly learns embeddings for dependency paths and feature combinations. Our experimental results indicate that our model substantially increases classification performance, leading to new state-of-the-art results. In a qualitive analysis, we found that our model is able to cover instances of various linguistic phenomena that are missed by other methods.

Beyond SRL, we expect dependency path embeddings to be useful in related tasks and downstream applications. For instance, our representations may be of direct benefit for semantic and discourse parsing tasks. The jointly learned feature space also makes our model a good starting point for cross-lingual transfer methods that rely on feature representation projection to induce new models (Kozhevnikov and Titov, 2014).

Acknowledgements We thank the three anonymous ACL referees whose feedback helped to substantially improve the present paper. The support of the Deutsche Forschungsgemeinschaft (Research Fellowship RO 4848/1-1; Roth) and the European Research Council (award number 681760; Lapata) is gratefully acknowledged.

References

Wilker Aziz, Miguel Rios, and Lucia Specia. 2011. Shallow semantic trees for smt. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 316–322, Edinburgh, Scotland.

Anders Björkelund, Love Hafdell, and Pierre Nugues. 2009. Multilingual semantic role labeling. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 43–48, Boulder, Colorado.

Anders Björkelund, Bernd Bohnet, Love Hafdell, and Pierre Nugues. 2010. A high-performance syntactic and semantic dependency parser. In *Coling 2010: Demonstration Volume*, pages 33–36, Beijing, China.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97, Beijing, China.

Wanxiang Che, Zhenghua Li, Yongqiang Li, Yuhang Guo, Bing Qin, and Ting Liu. 2009. Multilingual dependency-based syntactic and semantic parsing. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 49–54, Boulder, Colorado.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Koen Deschacht and Marie-Francine Moens. 2009. Semi-supervised semantic role labeling using the Latent Words Language Model. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 21–29, Singapore.

Nicholas FitzGerald, Oscar Täckström, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 960–970, Lisbon, Portugal.

William Foland and James Martin. 2015. Dependency-based semantic role labeling using convolutional neural networks. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 279–288, Denver, Colorado.

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

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18, Boulder, Colorado.

James Henderson, Paola Merlo, Ivan Titov, and Gabriele Musillo. 2013. Multilingual joint parsing of syntactic and semantic dependencies with a latent variable model. *Computational Linguistics*, 39(4):949–998.

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

Fei Huang and Alexander Yates. 2010. Open-domain semantic role labeling by modeling word spans. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 968–978, Uppsala, Sweden.

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 393–400, Manchester, United Kingdom.

Atif Khan, Naomie Salim, and Yogan Jaya Kumar. 2015. A framework for multi-document abstractive summarization based on semantic role labelling. *Applied Soft Computing*, 30:737–747.

Mikhail Kozhevnikov and Ivan Titov. 2014. Cross-lingual model transfer using feature representation projection. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 579–585, Baltimore, Maryland.

Tao Lei, Yuan Zhang, Lluís Màrquez, Alessandro Moschitti, and Regina Barzilay. 2015. High-order low-rank tensors for semantic role labeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1150–1160, Denver, Colorado.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint A* CCG parsing and semantic role labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1444–1454, Lisbon, Portugal.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 285–290, Beijing, China.

Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. 2015. Dependency-based convolutional neural networks for sentence embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 174–179, Beijing, China.

Derek Monner and James A Reggia. 2012. A generalized LSTM-like training algorithm for second-order recurrent neural networks. *Neural Networks*, 25:70–83.

Ahmed Hamza Osman, Naomie Salim, Mohammed Salem Binwahlan, Rihab Alteeb, and Albaraa Abuobieda. 2012. An improved plagiarism detection scheme based on semantic role labeling. *Applied Soft Computing*, 12(5):1493–1502.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Merin Paul and Sangeetha Jamal. 2015. An improved SRL based plagiarism detection technique using sentence ranking. *Procedia Computer Science*, 46:223–230.

Sameer Pradhan, Kadri Hacioglu, Wayne Ward, James H. Martin, and Daniel Jurafsky. 2005. Semantic role chunking combining complementary syntactic views. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 217–220, Ann Arbor, Michigan.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2):257–287.

Michael Roth and Mirella Lapata. 2015. Context-aware frame-semantic role labeling. *Transactions of the Association for Computational Linguistics*, 3:449–460.

Michael Roth and Kristian Woodsend. 2014. Composition of word representations improves semantic role labelling. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 407–413, Doha, Qatar.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959, Lake Tahoe, Nevada.

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

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

Kristina Toutanova, Aria Haghighi, and Christopher Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

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

Deyi Xiong, Min Zhang, and Haizhou Li. 2012. Modeling the translation of predicate-argument structure for smt. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 902–911, Jeju Island, Korea.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1785–1794, Lisbon, Portugal.

Nianwen Xue and Martha Palmer. 2004. Calibrating features for semantic role labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 88–94, Barcelona, Spain.

Beñat Zapirain, Eneko Agirre, Lluís Màrquez, and Mihai Surdeanu. 2013. Selectional preferences for semantic role classification. *Computational Linguistics*, 39(3):631–663.

Hai Zhao, Wenliang Chen, Jun'ichi Kazama, Kiyotaka Uchimoto, and Kentaro Torisawa. 2009. Multilingual dependency learning: Exploiting rich features for tagging syntactic and semantic dependencies. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009): Shared Task*, pages 61–66, Boulder, Colorado.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1127–1137, Beijing, China.

Prediction of Prospective User Engagement with Intelligent Assistants

Shumpei Sano, Nobuhiro Kaji, and Manabu Sassano
Yahoo Japan Corporation
9-7-1 Akasaka, Minato-ku, Tokyo 107-6211, Japan
{shsano, nkaji, msassano}@yahoo-corp.jp

Abstract

Intelligent assistants on mobile devices, such as Siri, have recently gained considerable attention as novel applications of dialogue technologies. A tremendous amount of real users of intelligent assistants provide us with an opportunity to explore a novel task of predicting whether users will continually use their intelligent assistants in the future. We developed prediction models of prospective user engagement by using large-scale user logs obtained from a commercial intelligent assistant. Experiments demonstrated that our models can predict prospective user engagement reasonably well, and outperforms a strong baseline that makes prediction based past utterance frequency.

1 Introduction

Intelligent assistants on mobile devices, such as Siri,[1] have recently gained considerable attention as novel applications of dialogue technologies (Jiang et al., 2015). They receive instructions from users via voice control to execute a wide range of tasks (*e.g.*, searching the Web, setting alarms, making phone calls, and so on). Some are able to even chat or play games with users (Kobayashi et al., 2015).

Intelligent assistants possess a unique characteristic as an object of dialogue study. Popular intelligent assistants have thousands or even millions of real users, thanks to the prevalence of mobile devices. Some of those users continually use intelligent assistants for a long period of time, while others stop using them after a few trials. Such user behaviors are rarely observed in conventional experimental environments, where dialogue systems have only a small number of experimental participants who almost always continue to use the systems for the whole duration of the experiment.

This paper explores a novel task of predicting whether a user will continue to use intelligent assistants in the future (This task is referred to as *prospective user engagement prediction* and its definition is given in Section 3). We attempt to develop such a prediction model, which would contribute to enhancing intelligent assistants in many ways. For example, if users who are likely to stop using systems can be identified, intelligent assistants can take actions to gain or maintain their interest (*e.g.*, by sending push notifications).

This task is related to, but is significantly different from, user engagement detection, which has been extensively explored in prior dialogue studies (Wang and Hirschberg, 2011; Forbes-Riley et al., 2012; Forbes-Riley and Litman, 2013; Oertel and Salvi, 2013). The prior studies attempt to predict how strongly users are currently engaged in dialogues with systems. On the other hand, the goal of this study is to predict how strongly users will be engaged with intelligent assistants in the future. The largest difference lies in whether the prediction target is user engagement at present or in the future. Also, our definition of engagement is slightly different from the prior ones. In this study, engagement is considered as a sentiment as to whether users like intelligent assistants and feel like they want to use them continually.

To develop and evaluate models of prospective user engagement prediction, we exploit large-scale user logs obtained from a commercial intelligent assistant. Since monitoring users' long-term behaviors is considered crucial for precise prediction of their prospective engagement, we tailor various features by extracting usage patterns from a long history of user dialogues. The resulting features are contrastive to those previously used for user

[1] http://www.apple.com/ios/siri

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1203–1212,
Berlin, Germany, August 7-12, 2016.

engagement detection, in which features are basically extracted from a single user utterance.

Experimental results demonstrated that our models are able to predict prospective user engagement reasonably well and are overwhelmingly better than a strong baseline that makes predictions based on past utterance frequency. We also discuss the trade-off between prediction accuracy and instancy. Specifically, we investigate how the prediction performance improves as we wait for more user dialogues to be collected.

2 Yahoo! Voice Assist

This section summarizes *Yahoo! Voice Assist*[2] (hereafter Voice Assist), a commercial intelligent assistant that is investigated in this study. Although our investigation exclusively focused on this system, we will discuss how our findings can be generalized to other intelligent assistants in Section 5.5

Table 1 illustrates example dialogues of Voice Assist users.[3] As illustrated, Voice Assist offers a variety of functions to mobile users. They are largely categorized into two types:

device operation Voice Assist allows users to operate mobile devices through dialogue. This includes setting alarms, making phone calls, searching the Web, launching an app, and so on (*e.g.*, **V1**, **V3**, **V4**, and **V5**).

chat Voice Assist can give greetings to, have conversations with, and play games with users (*e.g.*, **V2** and **V6**). In contrast to device operations for accomplishing certain tasks, these functions are offered for fun or for facilitating smooth communication.

Voice Assist currently supports 66 functions (including setting alarms, the word chain game, etc.) and they can further be classified into fine-grained types, although a detailed description of them is beyond the scope of this paper.

Voice Assist users can register personal profile such as their home address and birthday, with which the system makes personalized responses. For example, the home address are used to estimate users' location when launching weather forecast apps (*i.e.*, response **V3**), while knowing birthdays allows Voice Assist to send greeting messages to users on their birthdays.

U1	Wake me up at 8 o'clock tomorrow.
V1	OK. Set the alarm to 8 am. (show the timer configuration)
U2	Good morning.
V2	Good morning. Thank you for talking to me.
U3	Check today's weather.
V3	Tokyo's weather will be fine today. (launch weather forecast app)
U4	From Tokyo to Kyoto.
V4	A rapid express is available at 9:30. (launch transit app to show timetable)
U5	What time will it arrive?
V5	It will arrive at Kyoto at 11:50. (show the timetable again)
U6	Let's play the word chain game.
V6	OK. Apple...

Table 1: Example dialogues of Voice Assist users. **U** and **V** indicate the user and Voice Assist, respectively. The notes in parentheses represent actions that Voice Assist takes after the responses.

3 Prospective User Engagement Prediction

This section specifies the task of prospective user engagement prediction. We first explain the user log data used in our experiments. We then describe two kinds of task settings.

3.1 User log data

We conducted an empirical study in which we examined Voice Assist user logs. We randomly sampled 348,295 users who used the system at least once from March 2015 to June 2015 (16 weeks) and extracted all their dialogue histories during that period. The log data included 7,472,915 utterances in total.

Table 2 illustrates examples of user logs. We used the following seven attributes: user ID, nickname, birthday, time stamp, user utterance, system response, and response type. Because it is not mandatory to register the personal profiles (including nicknames, birthdays, etc.), they are sometimes missing, as indicated by N/A in the table. The response type represents the 66 functions supported by Voice Assist. The time stamps were used to segment utterances into sessions, as rep-

[2] `http://v-assist.yahoo.co.jp` (in Japanese)

[3] Because Voice Assist supports only Japanese, all utterances are made in Japanese. In this paper, we present English translations rather than the original Japanese to facilitate non-Japanese readers' understanding.

ID	Nickname	Birthday	Time Stamp	Use Utterance	System Response	Type
A	John	2000-1-1	2015-3-1 23:50	Wake me up at 8 am tomorrow.	OK. Set the alarm to 8 am.	ALARM
			2015-3-2 08:10	Good morning.	Good morning.	CHAT
			2015-3-2 08:13	Check today's weather.	Tokyo's weather will be fine today.	WEATHER
B	N/A	2002-1-1	2015-3-1 08:00	From Tokyo to Kyoto.	A rapid express is available at 9:30.	TRANSIT
			2015-3-1 08:01	What time will it arrive?	It will arrive at Kyoto at 11:50.	TRANSIT
			2015-3-5 19:10	Let's play the word chain game.	OK. Apple...	WORD CHAIN

Table 2: User log examples. The dashed line represents the session boundary.

resented by dashed lines in the table. We follow (Jiang et al., 2015) to define sessions as utterance sequences in which the interval of two adjacent utterances does not exceed 30 minutes.

3.2 Task definition

We propose two types of prospective user engagement prediction tasks. In both tasks, we collect user dialogues from the first eight weeks of the user logs (referred to as *observation period*. We will discuss on length of observation period in Section 5.4), and then use those past dialogues to predict whether users are engaged with the intelligent assistant in the last eight weeks of the log data (referred to as *prediction period*).[4] We specifically explored two prediction tasks as follows.

Dropout prediction The first task is to predict whether a given user will not at all use the system in the prediction period. This task is referred to as *dropout prediction* and is formulated as a binary classification problem. The model of dropout prediction would allow intelligent assistants to take proactive actions against users who are likely to stop using the system. There are 71,330 dropout users, who does not at all use the system in the prediction period, among 275,630 in our data set.

Engagement level prediction The second task aims at predicting how frequently the system will be used in the prediction period by a given user. Because there are outliers, or heavy users, who use the system extremely frequently (one user used the system as many as 1,099 times in the eight weeks), we do not attempt to directly predict the number of utterances or sessions. Instead, we define engagement levels as detailed below, and aim at predicting those values.

The engagement levels are defined as follows. First, users are sorted in the ascending order of the number of sessions they made in the prediction period. We then split users into four equally-sized groups. The engagement levels of users in the four groups are defined as 1, 2, 3, and 4, respectively (Table 3). Note that a larger value of the engagement level means that the users are more engaged with the intelligent assistants. This task is referred to as *engagement level prediction* and is formulated as a regression problem.

Level	# of sessions	# of users
1	0	71,330
2	1–3	66,626
3	4–13	69,551
4	14–	68,123

Table 3: User distribution over the four engagement levels. The second column represents intervals of the number of sessions corresponding to the four levels.

The engagement level prediction has different applications from the dropout prediction. For example, it would allow us to detect in advance that a user's engagement level will change from four to three in the near future. It is beyond the scope of dropout prediction task to foresee such a change.

4 Features

The dropout prediction is performed using linear support vector machine (SVM) (Fan et al., 2008), while the engagement level prediction is performed using support vector regression (SVR) (Smola and Schölkopf, 2004) on the same feature set. Here, we divide the features into four categories by their function: utterance frequency features, response frequency features, time interval features, and user profile features. Table 4 lists these features.

4.1 Utterance frequency features

Here, we describe the features related to utterance frequency. These features attempt to capture our

[4] We removed users from the log data if the number of sessions was only once in the observation period, because such data lack a sufficient amount of dialogue histories for making a reliable prediction.

#Features	Name	Definition
1	**Utterance**	The number of utterances
7	**Utterance**w**Weeks**	The number of utterances in recent w weeks
1	**LongUtterance**	The number of lengthy utterances
1	**UrgedUtterance**	The number of utterances made in response to push notifications
1	**Restatement**	The number of restatement utterances
100	**UtteranceTopic**i	The number of utterances including words in the i-th cluster
1	**Session**	The number of sessions
7	**Session**w**Weeks**	The number of sessions in recent w weeks
7	**SessionByDay**	The number of sessions during each day of the week
66	**Response**(t)	The number of responses with response type t
66	**FirstResponse**(t)	**Response**(t) computed by using only the first responses in sessions
1	**LongResponse**	The number of lengthy responses
1	**ErrorMessage**	The number of error messages
1	**MaxInterval**	Max days between adjacent utterances
1	**MinInterval**	Min days between adjacent utterances
1	**AvgInterval**	Average days between adjacent utterances
1	**InactivePeriod**	Days from the last utterance date
66	**InactivePeriod**(t)	**InactivePeriod** computed for each type of the last response
1	**Nickname**	Whether or not a user has provided nickname information
1	**Birthday**	Whether or not a user has provided birthday information
6	**Age**	User's age category

Table 4: List of features. The utterance frequency features, response frequency features, and time interval features are all scaled.

intuition that users who frequently use intelligent assistants are likely to be engaged with them.

Utterance The number of utterances in the observation period. For scaling purposes, the value of this feature is set to $\log_{10}(x+1)$, where x is the number of utterances. The same scaling is performed on all features but user profile features.

Utterance$w$**Weeks** The number of utterances in the last w $(1 \leq w < 8)$ weeks of the observation period.

LongUtterance The number of lengthy utterances (more than 20 characters long). Jiang et al. (2015) pointed out that long utterances are prone to cause ASR errors. Since ASR errors are a factor that decreases user engagement, users who are prone to make long utterances are likely to be disengaged.

UrgedUtterance The number of utterances made in response to push notifications sent from the system. We expect that engaged users tend to react to push notifications.

Restatement The number of restatements made by users. Jiang et al. (2015) found that users tend to repeat previous utterances in case of ASR errors. An utterance is regarded as a restatement of the previous one if their normalized edit distance (Li and Liu, 2007) is below 0.5.

UtteranceTopic$i$ The number of utterances including a keyword belonging to i-th word cluster. To induce the word clusters, 100-dimensional word embeddings are first learned from the log data using WORD2VEC (Mikolov et al., 2013)[5], and then K-means clustering (K=100) is performed (MacQueen, 1967). All options of WORD2VEC are set to the default values. These features aim at capturing topics on utterances or speech acts. Table 5 illustrates example words in the clusters. For example, utterances including words in the cluster ID 36 and 63 are considered to be greeting acts and sports-related conversations, respectively.

[5] https://code.google.com/archive/p/word2vec

Cluster ID	Example words
14 (Weather)	pollen, typhoon, temperature
23 (Curse)	die, stupid, shit, shurrup, dorf
36 (Greeting)	thanks, good morning, hello
48 (Sentiment)	funny, cute, good, awesome
63 (Sports)	World cup, Nishikori, Yankees

Table 5: Example words in the clusters. Cluster names (presented in parentheses) are manually provided by the authors to help readers understand the word clusters.

Session The number of sessions in the observation period.

Session$w$Weeks The number of sessions in the last w $(1 \leq w < 8)$ weeks of the observation period.

SessionByDay The number of sessions in each day of week. There are seven different features of this type.

4.2 Response frequency features

Here, we describe the features of the response frequency.

Response(t) The number of system responses with response type t.

FirstResponse(t) **Response(t)** features that are computed by using only the first responses in sessions. Our hypothesis is that first responses in sessions crucially affect user engagement.

LongResponse The number of lengthy responses (more than 50 characters long). Because longer responses require a longer reading time, they are prone to irritate users and consequently decrease user engagement.

ErrorMessage The number of error messages. Voice Assist returns error messages (*Sorry, I don't know.*) when it fails to find appropriate responses to the user's utterances. We consider that these error messages decrease user engagement.

4.3 Time interval features

Here, we describe the features related to the session interval times.

MaxInterval The maximum interval (in days) between adjacent sessions in the observation period.

MinInterval The minimum interval (in days) between adjacent sessions in the observation period.

AvgInterval The average interval (in days) between adjacent sessions in the observation period.

InactivePeriod The time span (in days) from the last utterance to the end of the observation period.

InactivePeriod(t) **InactivePeriod** computed separately for each type t of the last response.

4.4 User profile features

Here, we describe the features of the user's profile information. Since it is not mandotory for users to register their profiles, we expect that those who have provided profile information are likely to be engaged with the system.

Nickname A binary feature representing whether or not the user has provided their nickname.

Birthday A binary feature representing whether or not the user has provided their birthday.

Age Six binary features representing the user's age. They respectively indicate whether the user is less than twenty years, in their 20's, 30's, 40's, or 50's, or is more than 60 years old. Note that these features are available only if the user has provided their birthday.

5 Experiments

In this section, we describe our experimental results and discuss them.

5.1 Experimental settings

We randomly divided the log data into training, development, and test sets with the ratio of 8:1:1. Note that we confirmed that the users in different data sets do not overlap with each other. We trained the model with the training set and optimized hyperparameters with the development set. The test set was used for a final blind test to evaluate the learnt model.

We used the LIBLINEAR tool (Fan et al., 2008) to train the SVM for the dropout prediction and

	Accuracy	F–measure
Baseline	56.8	0.482
Proposed	**77.6**	**0.623**
Utterance frequency	70.2	0.578
Response frequency	54.8	0.489
Time interval	74.6	0.617
User profile	39.9	0.406

Table 6: Classification accuracies and F–measures in the dropout prediction task.

	Precision	Recall
Baseline	0.350	0.774
Proposed	**0.553**	0.714
Utterance frequency	0.458	0.785
Response frequency	0.346	**0.831**
Time interval	0.507	0.789
User profile	0.273	0.793

Table 7: Precisions and Recalls in the dropout prediction task.

the SVR for the engagement level prediction task. We optimized the C parameter on the development set. In the dropout prediction task, we used the `-w` option to weigh the C parameter of each class with the inverse ratio of the number of users in that class. We also used the `-B` option to introduce the bias term.

Next, we describe the evaluation metrics. We used accuracy and F_1–measure in the dropout prediction task. Mean squared error (MSE) and Spearman rank correlation coefficient were used in the engagement level prediction task. These evaluation metrics are commonly used in classification and regression tasks.

We compare the proposed models with baseline method. Because we have no previous work on both tasks, we defined baseline method of our own. The baseline method was trained in the same framework as the proposed methods except that they used only **Session** feature. We chose **Session** for baseline because frequency of use features such as **Session** were shown predictive to similar tasks (Kloft et al., 2014; Sadeque et al., 2015) to prospective user engagement.

5.2 Results

Table 6 illustrates the result of dropout prediction task. The first row compares the proposed method with the baseline. We can see that the proposed

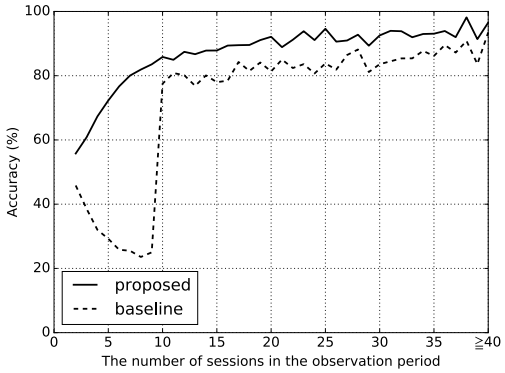


Figure 1: Accuracies per the number of sessions in the observation period of the proposed method and the baseline. The rightmost points represent the accuracy of the users whose number of sessions in the observation period are equal to or more than 40.

	MSE	Spearman
Baseline	0.784	0.595
Proposed	**0.578**	**0.727**
Utterance frequency	0.632	0.693
Response frequency	0.798	0.584
Time interval	0.645	0.692
User profile	1.231	0.146

Table 8: MSE and Spearman's ρ in the engagement level prediction task.

model outperforms the baseline. This indicates the effectiveness of our feature set. The second row illustrates the performances of the proposed method when only one feature type is used. This result suggests that the utterance frequency and time interval features are especially useful, while the combination of all types of features performs the best. We conducted McNemar test (McNemar, 1947) to investigate the significance of these improvements, and confirmed that all improvements are statistically significant ($p < 0.01$).

Table 7 shows the precisions and the recalls of dropout prediction task. As shown in Table 7, the precision of the proposed method performs the best while the recall is worst. We consider that the performance of the precision is more important for our model because taking proactive actions against users who are likely to stop using the system is one of the assumed applications. Taking proactive actions (*e.g.*, push notifications) against users continually using the system might irritate them and de-

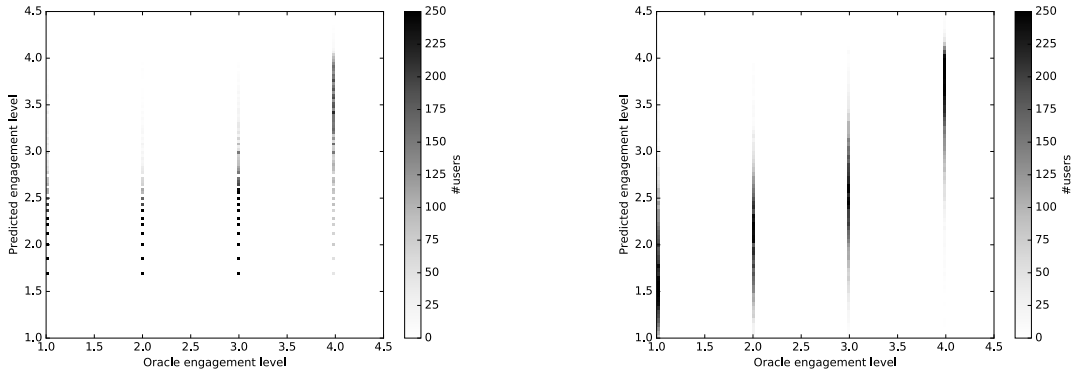


Figure 2: Correlation between the oracle engagement levels and the ones predicted by the baseline method (left) and by the proposed method (right).

crease their user engagement. Therefore, the rate of the users who actually intend to stop using the system in the users predicted as dropout affects the effectiveness of these proactive actions. The result that the precision of the proposed method is 0.553 and that of the baseline is 0.350 is, in other words, using the proposed model improves the effectiveness by 20% absolute in taking these actions.

Figure 1 shows the accuracies per the number of sessions in the observation period of the proposed method and the baseline. The proposed method consistently outperforms the baseline throughout the number of sessions in the observation period. In particular, the proposed method predicts well the dropout of users whose number of sessions is around five compared to the baseline. These results again indicate the effectiveness of the combination of our feature set.

Table 8 shows the result of engagement level prediction task. We again observe similar trends to the dropout prediction task. The proposed method outperforms the baseline. The utterance frequency and time interval features are the most effective, while the combination of all four feature types achieves the best performance in both evaluation metrics.

Figure 2 visualizes the correlation between the oracle engagement levels and the ones predicted by the baseline (left) and by the proposed method (right). We can intuitively reconfirm that the proposed method is able to predict the engagement levels reasonably well.

5.3 Investigation of feature weights

We investigate weights of the features learned by the SVR for figuring out what features contribute to the precise prediction of prospective user engagement.

Table 9 exemplifies features that received large weights for the four feature types. We observe that most features with large positive or negative weights are from the utterance frequency and time interval features. Those include **Session**, **Utterance**, and **InactivePeriod**. It is interesting to see that **UrgedUtterance**, which is based on an utterance type specific to mobile users, also receives a large positive weight.

Further detailed analysis revealed that the proposed model captures some linguistic properties that correlate with the prospective user engagement. For example, **UtteranceTopic36** and **UtteranceTopic23** recieve positive and negative weights, respectively. This follows our intuition since those clusters correspond to greeting and curse words (*c.f.* Table 5). We also observe **Response(WORD CHAIN)**, **Response(QUIZ)** (word association quiz), and **Response(TRIVIA)** (showing some trivia) receive positive weights. This means that playing games or showing some trivia attract users. It is interesting to see that this result is consistent with findings in (Kobayashi et al., 2015). It also follows our intuition that the weight of **ErrorMessage** feature is negative.

5.4 Discussion on length of observation period

Next, we investigate how the length of the observation period affects the prediction performance. We varied the length of the observation periods from one to eight weeks, and evaluated the results (Figure 3).

Figure 3 demonstrates that the model perfor-

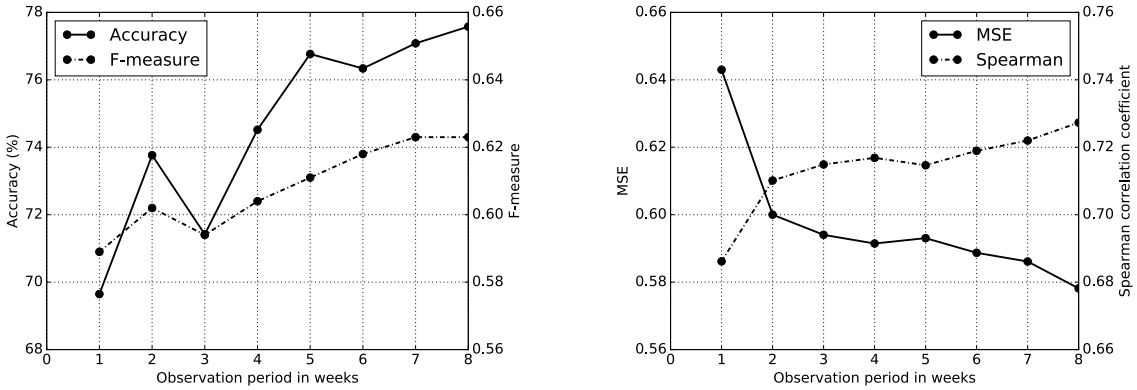


Figure 3: Results of dropout prediction (left) and engagement level prediction (right) across different observation periods (in weeks).

Weight	Feature
0.67	**Session**
0.59	**Utterance**
0.28	**Session7Weeks**
0.26	**UrgedUtterance**
0.02	**UtteranceTopic36**
-0.05	**UtteranceTopic23**
0.08	**Response(WORD CHAIN)**
0.08	**Response(QUIZ)**
0.04	**Response(TRIVIA)**
-0.03	**ErrorMessage**
-0.23	**InactivePeriod(ALARM)**
-0.46	**InactivePeriod**
0.05	**Birthday**
0.04	**Age60s**

Table 9: Feature weights learned by the SVR.

mance generally improves as the observation period becomes longer in both tasks. When we increase the length of the observation period from one week to eight weeks, the accuracy increases by 7.9% in the dropout prediction and Spearman's ρ increases by 4.1 point in the engagement level prediction. The most significant improvements are achieved when we increase the length from one week to two weeks in the three metrics except the F–measure. This suggests that it is generally effective to collect user dialogues of two weeks long, rather than as long as eight weeks or more. This approach would allow to make predictions promptly without waiting for user dialogues to be collected for a long time, while harming accuracy (or other evaluation metrics) as little as possible.

5.5 Application to other intelligent assistants

Here, we discuss how well our approach applies to intelligent assistants other than Voice Assist. The results of this study are considered to apply to other intelligent assistants so long as user logs like the ones in Table 2 are available. The concern is that some attributes in Table 2 may not be available in other systems. In the following, we investigate two attributes, response types and profiles, that are specific to Voice Assist.

We consider that response types like ours are available in user logs of many other intelligent assistants as well. Because our response types mostly correspond to commands issued when operating mobile devices, response types analogous to ours can be obtained by simply logging the commands. Alternatively, it would be possible to employ taggers like (Jiang et al., 2015) to automatically type system responses.

As for profiles, it is likely that similar information is also available in many other intelligent assistants because profile registration is a common function in many IT services including intelligent assistants. For example, Cortana offers greetings and other activities on special days registered by users.[6] Even if user profiles were not at all available, we consider that it would not seriously spoil the significance of this study, because our experiments revealed that user profiles are among the least predictive features.

[6] `http://m.windowscentral.com/articles` (an article posted on Dec. 5, 2015)

6 Related Work

Many dialogue studies have explored the issue of detecting user engagement as well as related affects such as interest and uncertainty (Wang and Hirschberg, 2011; Forbes-Riley et al., 2012; Forbes-Riley and Litman, 2013; Oertel and Salvi, 2013). As discussed in Section 1, these studies typically use a single user utterance to predict whether the user is currently engaged in dialogues with systems. We introduced a new perspective on this line of research by exploring models of predicting prospective user engagement in a large-scale empirical study.

Kobayashi et al. (2015) investigated how games played with intelligent assistants affect prospective user engagement. Although their research interest was prospective user engagement like ours, they exclusively studied the effect of playing game, and left other factors unexplored. In addition, they did not develop any prediction models.

Recently, user satisfaction for intelligent assistants gain attention(Jiang et al., 2015; Kiseleva et al., 2016a; Kiseleva et al., 2016b). Jiang et al. (2015) proposed an automatic method of assessing user satisfaction with intelligent assistants. Kiseleva et al. extended the study of Jiang et al. for prediction (2016a) and detailed understanding (2016b) of user satisfaction with intelligent assistants. Although both satisfaction and engagement are affective states worth considering by intelligent assistants, their research goals were quite different from ours. In their studies, user satisfaction was measured as to whether intelligent assistants can accomplish predefined tasks (*e.g.*, checking the exchange rate between US dollars and Australian dollars). This virtually assesses task-level response accuracy, which is a different notion from user engagement.

Nevertheless, we consider that their studies are closely related to ours and indeed helpful for improving the proposed model. Since user satisfaction is considered to greatly affect prospective user engagement, it might be a good idea to use automatically evaluated satisfaction levels as additional features. The proposed model currently uses **ErrorMessage** feature as an alternative that can be implemented with ease.

Several studies have investigated the chances of predicting continuous participation in SNSs such as MOOC and health care forum (Rosé and Siemens, 2014; Kloft et al., 2014; Ramesh et al., 2014; Sadeque et al., 2015). Unlike those studies, this study exclusively investigates a specific type of dialogue system, namely intelligent assistants, and aims at uncovering usage and/or response patterns that strongly affect prospective user engagement. Consequently, many of the proposed features are specially designed to analyze intelligent assistant users rather than SNS participants.

Our work also relates to the evaluation of dialogue systems. Walker et al. (1997) presented the offline evaluation framework for spoken dialog system (PARADISE). They integrate various evaluation metrics such as dialogue success and dialogue costs into one performance measure function. Although our goal is to predict prospective user engagement and different from theirs, some measures (*e.g.*, the number of utterances) are useful to predict prospective user engagement with intelligent assistants.

7 Conclusion

This paper explored two tasks of predicting prospective user engagement with intelligent assistants: dropout prediction and engagement level prediction. The experiments successfully demonstrated that reasonable performance can be archived in both tasks. Also, we examined how the length of the observation period affects prediction performance, and investigated the trade-off between prediction accuracy and instancy. The future work includes using those prediction models in a real service to take targeted actions to users who are likely to stop using intelligent assistants.

References

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Kate Forbes-Riley and Diane Litman. 2013. When does disengagement correlate with performance in spoken dialog computer tutoring? *International Journal of Artificial Intelligence in Education*, 22(1-2):39–58.

Kate Forbes-Riley, Diane Litman, Heather Friedberg, and Joanna Drummond. 2012. Intrinsic and extrinsic evaluation of an automatic user disengagement detector for an uncertainty-adaptive spoken dialogue system. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 91–102. Association for Computational Linguistics.

Jiepu Jiang, Ahmed Hassan Awadallah, Rosie Jones, Umut Ozertem, Imed Zitouni, Ranjitha Gurunath Kulkarni, and Omar Zia Khan. 2015. Automatic online evaluation of intelligent assistants. In *Proceedings of the 24th International Conference on World Wide Web*, pages 506–516. International World Wide Web Conferences Steering Committee.

Julia Kiseleva, Kyle Williams, Jiepu Jiang, Ahmed Hassan Awadallah, Imed Zitouni, Aidan C Crook, and Tasos Anastasakos. 2016a. Predicting user satisfaction with intelligent assistants. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 495–505. ACM.

Julia Kiseleva, Kyle Williams, Jiepu Jiang, Ahmed Hassan Awadallah, Aidan C Crook, Imed Zitouni, and Tasos Anastasakos. 2016b. Understanding user satisfaction with intelligent assistants. In *Proceedings of the 2016 ACM SIGIR Conference on Human Information Interaction and Retrieval*, pages 121–130. ACM.

Marius Kloft, Felix Stiehler, Zhilin Zheng, and Niels Pinkwart. 2014. Predicting MOOC dropout over weeks using machine learning methods. In *Proceedings of the 2014 Empirical Methods in Natural Language Processing*, pages 60–65. Association for Computational Linguistics.

Hayato Kobayashi, Kaori Tanio, and Manabu Sassano. 2015. Effects of game on user engagement with spoken dialogue system. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 422–426. Association for Computational Linguistics.

Yujian Li and Bo Liu. 2007. A normalized Levenshtein distance metric. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1091–1095.

James MacQueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematics, Statistics and Probability, Vol. 1*, pages 281–297.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Catharine Oertel and Giampiero Salvi. 2013. A gaze-based method for relating group involvement to individual engagement in multimodal multiparty dialogue. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*, pages 99–106. ACM.

Arti Ramesh, Dan Goldwasser, Bert Huang, Hal Daumé III, and Lise Getoor. 2014. Learning latent engagement patterns of students in online courses. In *Proceedings of the Twenty-Eighth AAAI Conference Artificial Intelligence*, pages 1272–1278. Association for the Advancement of Artificial Intelligence.

Carolyn Rosé and George Siemens. 2014. Shared task on prediction of dropout over time in massively open online courses. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing Workshop on Modeling Large Scale Social Interaction in Massively Open Online Courses*, pages 39–41. Association for Computational Linguistics.

Farig Sadeque, Thamar Solorio, Ted Pedersen, Prasha Shrestha, and Steven Bethard. 2015. Predicting continued participation in online health forums. In *Proceedings of the Sixth International Workshop on Health Text Mining and Information Analysis*, pages 12–20. Association for Computational Linguistics.

Alex J. Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

Marilyn A Walker, Diane J Litman, Candace A Kamm, and Alicia Abella. 1997. Paradise: A framework for evaluating spoken dialogue agents. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 271–280. Association for Computational Linguistics.

William Yang Wang and Julia Hirschberg. 2011. Detecting levels of interest from spoken dialog with multistream prediction feedback and similarity based hierarchical fusion learning. In *Proceedings of the 12th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 152–161. Association for Computational Linguistics.

Resolving References to Objects in Photographs using the Words-As-Classifiers Model

David Schlangen Sina Zarrieß Casey Kennington
Dialogue Systems Group // CITEC // Faculty of Linguistics and Literary Studies
Bielefeld University, Germany
first.last@uni-bielefeld.de

Abstract

A common use of language is to refer to visually present objects. Modelling it in computers requires modelling the link between language and perception. The "words as classifiers" model of grounded semantics views words as classifiers of perceptual contexts, and composes the meaning of a phrase through composition of the denotations of its component words. It was recently shown to perform well in a game-playing scenario with a small number of object types. We apply it to two large sets of real-world photographs that contain a much larger variety of object types and for which referring expressions are available. Using a pre-trained convolutional neural network to extract image region features, and augmenting these with positional information, we show that the model achieves performance competitive with the state of the art in a reference resolution task (*given expression, find bounding box of its referent*), while, as we argue, being conceptually simpler and more flexible.

1 Introduction

A common use of language is to refer to objects in the shared environment of speaker and addressee. Being able to simulate this is of particular importance for verbal human/robot interfaces (HRI), and the task has consequently received some attention in this field (Matuszek et al., 2012; Tellex et al., 2011; Krishnamurthy and Kollar, 2013).

Here, we study a somewhat simpler precursor task, namely that of resolution of reference to objects in static images (photographs), but use a larger set of object types than is usually done in HRI work ($>$ 300, see below). More formally, the task is to retrieve, given a referring expression e and an image I, the region bb^* of the image that is most likely to contain the referent of the expression. As candidate regions, we use both manually annotated regions as well as automatically computed ones.

As our starting point, we use the "words-as-classifiers" model recently proposed by Kennington and Schlangen (2015). It has before only been tested in a small domain and with specially designed features; here, we apply it to real-world photographs and use learned representations from a convolutional neural network (Szegedy et al., 2015). We learn models for between 400 and 1,200 words, depending on the training data set. As we show, the model performs competitive with the state of the art (Hu et al., 2016; Mao et al., 2016) on the same data sets.

Our background interest in situated interaction makes it important for us that the approach we use is 'dialogue ready'; and it is, in the sense that it supports incremental processing (giving results while the incoming utterance is going on) and incremental learning (being able to improve performance from interactive feedback). However, in this paper we focus purely on 'batch', non-interactive performance.[1]

2 Related Work

The idea of connecting words to what they denote in the real world via perceptual features goes back at least to Harnad (1990), who coined "The Symbol Grounding Problem": "*[H]ow can the semantic interpretation of a formal symbol system be made* intrinsic *to the system, rather than just parasitic on the meanings in our heads?*" The pro-

[1]The code for reproducing the results reported in this paper can be found at `https://github.com/dsg-bielefeld/image_wac`.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1213–1223,
Berlin, Germany, August 7-12, 2016.

posed solution was to link 'categorial representations' with "learned and innate feature detectors that pick out the invariant features of object and event categories from their sensory projections".

This suggestion has variously been taken up in computational work. An early example is Deb Roy's work from the early 2000s (Roy et al., 2002; Roy, 2002; Roy, 2005). In (Roy et al., 2002), computer vision techniques are used to detect object boundaries in a video feed, and to compute colour features (mean colour pixel value), positional features, and features encoding the relative spatial configuration of objects. These features are then associated in a learning process with certain words, resulting in an association of colour features with colour words, spatial features with prepositions, etc., and based on this, these words can be interpreted with reference to the scene currently presented to the video feed.

Of more recent work, that of Matuszek et al. (2012) is closely related to the approach we take. The task in this work is to compute (sets of) referents, given a (depth) image of a scene containing simple geometric shapes and a natural language expression. In keeping with the formal semantics tradition, a layer of logical form representation is assumed; it is not constructed via syntactic parsing rules, however, but by a learned mapping (*semantic parsing*). The non-logical constants of this representation then are interpreted by linking them to classifiers that work on perceptual features (representing shape and colour of objects). Interestingly, both mapping processes are trained jointly, and hence the links between classifiers and non-logical constants on the one hand, and non-logical constants and lexemes on the other are induced from data. In the work presented here, we take a simpler approach that forgoes the level of semantic representation and directly links lexemes and perceptions, but does not yet learn the composition.

Most closely related on the formal side is recent work by Larsson (2015), which offers a very direct implementation of the 'words as classifiers' idea (couched in terms of type theory with records (TTR; (Cooper and Ginzburg, 2015)) and not model-theoretic semantics). In this approach, some lexical entries are enriched with classifiers that can judge, given a representation of an object, how applicable the term is to it. The paper also describes how these classifiers could be trained (or adapted) in interaction. The model is only specified theoretically, however, with hand-crafted classifiers for a small set of words, and not tested with real data.

The second area to mention here is the recently very active one of image-to-text generation, which has been spurred on by the availability of large datasets and competitions structured around them. The task here typically is to generate a description (a caption) for a given image. A frequently taken approach is to use a convolutional neural network (CNN) to map the image to a dense vector (which we do as well, as we will describe below), and then condition a neural language model (typically, an LSTM) on this to produce an output string (Vinyals et al., 2015; Devlin et al., 2015). Fang et al. (2015) modify this approach somewhat, by using what they call "word detectors" first to specifically propose words for image regions, out of which the caption is then generated. This has some similarity to our word models as described below, but again is tailored more towards generation.

Socher et al. (2014) present a more compositional variant of this type of approach where sentence representations are composed along the dependency parse of the sentence. The representation of the root node is then mapped into a multimodal space in which distance between sentence and image representation can be used to guide image retrieval, which is the task in that paper. Our approach, in contrast, composes on the level of denotations and not that of representation.

Two very recent papers carry this type of approach over to the problem of resolving references to objects in images. Both (Hu et al., 2015) and (Mao et al., 2015) use CNNs to encode image information (and interestingly, both combine, in different ways, information from the candidate region with more global information about the image as a whole), on which they condition an LSTM to get a prediction score for fit of candidate region and referring expression. As we will discuss below, our approach has some similarities, but can be seen as being more compositional, as the expression score is more clearly composed out of individual word scores (with rule-driven composition, however). We will directly compare our results to those reported in these papers, as we were able to use the same datasets.

3 The "Words-As-Classifiers" Model

We now briefly review (and slightly reformulate) the model introduced by Kennington and Schlangen (2015). It has several components:

A Model of Word Meanings Let w be a word whose meaning is to be modelled, and let $\mathbf{x}$ be a representation of an object in terms of its visual features. The core ingredient then is a classifier then takes this representation and returns a score $f_w(\mathbf{x})$, indicating the "appropriateness" of the word for denoting the object.

Noting a (loose) correspondence to Montague's (1974) intensional semantics, where the intension of a word is a function from possible worlds to extensions (Gamut, 1991), the *intensional* meaning of w is then defined as the classifier itself, a function from a representation of an object to an "appropriateness score":[2]

$$[\![w]\!]_{obj} = \lambda \mathbf{x}.f_w(\mathbf{x}) \quad (1)$$

(Where $[\![.]\!]$ is a function returning the meaning of its argument, and $\mathbf{x}$ is a feature vecture as given by f_{obj}, the function that computes the representation for a given object.)

The *extension* of a word in a given (here, visual) discourse universe W can then be modelled as a probability distribution ranging over all candidate objects in the given domain, resulting from the application of the word intension to each object ($\mathbf{x}_i$ is the feature vector for object i, $normalize()$ vectorized normalisation, and I a random variable ranging over the k candidates):

$$\begin{aligned}&[\![w]\!]^W_{obj} = \\ &\quad normalize(([\![w]\!]_{obj}(\mathbf{x}_1), \ldots, [\![w]\!]_{obj}(\mathbf{x}_k))) = \\ &\quad normalize((f_w(\mathbf{x}_1), \ldots, f_w(\mathbf{x}_k))) = P(I|w) \end{aligned} \quad (2)$$

Composition Composition of word meanings into phrase meanings in this approach is governed by rules that are tied to syntactic constructions. In the following, we only use simple multiplicative composition for nominal constructions:

$$[\![[_{nom} w_1, \ldots, w_k]]\!]^W = [\![\text{NOM}]\!]^W [\![w_1, \ldots, w_k]\!]^W = \circ_{/N} ([\![w_1]\!]^W, \ldots, [\![w_k]\!]^W) \quad (3)$$

where $\circ_{/N}$ is defined as

$$\begin{aligned}&\circ_{/N} ([\![w_1]\!]^W, \ldots, [\![w_k]\!]^W) = P_\circ(I|w_1, \ldots, w_k) \\ &\quad \text{with } P_\circ(I = i|w_1, \ldots, w_k) = \\ &\quad \frac{1}{Z}(P(I = i|w_1) * \cdots * P(I = i|w_k)) \text{ for } i \in I \end{aligned} \quad (4)$$

(Z takes care that the result is normalized over all candidate objects.)

[2](Larsson, 2015) develops this intension/extension distinction in more detail for his formalisation.

Selection To arrive at the desired extension of a full referring expression—an individual object, in our case—, one additional element is needed, and this is contributed by the determiner. For uniquely referring expressions ("the red cross"), what is required is to pick the most likely candidate from the distribution:

$$[\![the]\!] = \lambda x. \arg\max_{Dom(x)} x \quad (5)$$

$$[\![[the]\ [_{nom} w_1, \ldots, w_k]]\!]^W = \arg\max_{i \in W} [\ [\![[_{nom} w_1, \ldots, w_k]]\!]^W\ ] \quad (6)$$

In other words, the prediction of an expression such as "the brown shirt guy on right" is computed by first getting the responses of the classifiers corresponding to the words, individually for each object. I.e., the classifier for "brown" is applied to objects $o_1, \ldots, o_n$. This yields a vector of responses (of dimensionality n, the number of candidate objects); similarly for all other words. These vectors are then multiplied, and the predicted object is the maximal component of the resulting vector. Figure 1 gives a schematic overview of the model as implemented here, including the feature extraction process.

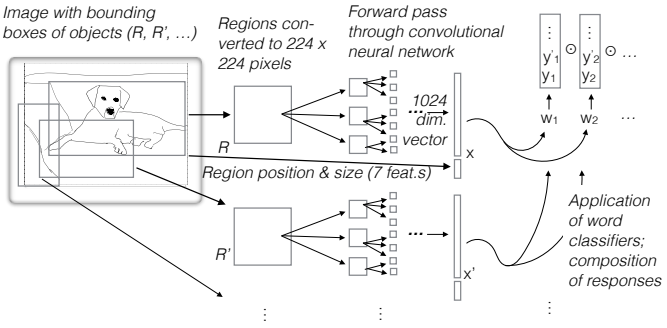


Figure 1: Overview of the model

4 Data: Images & Referring Expressions

SAIAPR TC-12 / ReferItGame The basis of this data set is the IAPR TC-12 image retrieval benchmark collection of "20,000 still natural images taken from locations around the world and comprising an assorted cross-section of still natural images" (Grubinger et al., 2006). A typical example of an image from the collection is shown in Figure 2 on the left.

This dataset was later augmented by Escalante et al. (2010) with segmentation masks identifying objects in the images (an average of 5 objects per image). Figure 2 (middle) gives an example of such a segmentation. These segmentations were

Figure 2: Image 27437 from IAPR TC-12 (left), with region masks from SAIAPR TC-12 (middle); "brown shirt guy on right" is a referring expression in REFERITGAME for the region singled out on the right

done manually and provide close maskings of the objects. This extended dataset is also known as "SAIAPR TC-12" (for "segmented and annotated IAPR TC-12").

The third component is provided by Kazemzadeh et al. (2014), who collected a large number of expressions referring to (pre-segmented) objects from these images, using a crowd-sourcing approach where two players were paired and a director needed to refer to a predetermined object to a matcher, who then selected it. (An example is given in Figure 2 (right).) This corpus contains 120k referring expressions, covering nearly all of the 99.5k regions from SAIAPR TC-12.[3] The average length of a referring expression from this corpus is 3.4 tokens. The 500k token realise 10,340 types, with 5785 hapax legomena. The most frequent tokens (other than articles and prepositions) are "left" and "right", with 22k occurrences. (In the following, we will refer to this corpus as REFERIT.)

This combination of segmented images and referring expressions has recently been used by Hu et al. (2015) for learning to resolve references, as we do here. The authors also tested their method on region proposals computed using the EdgeBox algorithm (Zitnick and Dollár, 2014). They kindly provided us with this region proposal data (100 best proposals per image), and we compare our results on these region proposals with theirs below. The authors split the dataset evenly into 10k images (and their corresponding referring expressions) for training and 10k for testing. As we needed more training data, we made a 90/10 split, ensuring that all our test images are from their test split.

MSCOCO / GoogleRefExp / ReferItGame

The second dataset is based on the "Microsoft Common Objects in Context" collection (Lin et al., 2014), which contains over 300k images with object segmentations (of objects from 80 pre-specified categories), object labels, and image captions. Figure 3 shows some examples of images containing objects of type "person".

This dataset was augmented by Mao et al. (2015) with what they call 'unambiguous object descriptions', using a subset of 27k images that contained between 2 and 4 instances of the same object type within the same image. The authors collected and validated 100k descriptions in a crowd-sourced approach as well, but unlike in the ReferItGame setup, describers and validators were not connected live in a game setting.[4] The average length of the descriptions is 8.3 token. The 790k token in the corpus realise 14k types, with 6950 hapax legomena. The most frequent tokens other than articles and prepositions are "man" (15k occurrences) and "white" (12k). (In the following, we will refer to this corpus as GREXP.)

The authors also computed automatic region proposals for these images, using the multibox method of Erhan et al. (2014) and classifying those using a model trained on MSCOCO categories, retaining on average only 8 per image. These region proposals are on average of a much higher quality than those we have available for the other dataset.

As mentioned in (Mao et al., 2015), Tamara Berg and colleagues have at the same time used their ReferItGame paradigm to collect referring expressions for MSCOCO images as well. Upon request, Berg and colleagues also kindly provided us with this data—140k referring expressions, for 20k images, average length 3.5 token, 500k token altogether, 10.3k types, 5785 hapax legomena; most frequent also "left" (33k occurrences)

[3]The IAPR TC-12 and SAIAPR TC-12 data is available from `http://imageclef.org`; REFERITGAME from `http://tamaraberg.com/referitgame`.

[4]The data is available from `https://github.com/mjhucla/Google_Refexp_toolbox`.

REFCOCO:
green woman
GREXP:
a woman wearing blue jeans

REFCOCO:
person left
GREXP:
a boy is ready to play who is wearing green color pant

Figure 3: Examples from MSCOCO

and "right" (32k). (In the following, we will call this corpus REFCOCO.) In our experiments, we use the training/validation/test splits on the images suggested by Berg et al., as the splits provided by Mao et al. (2015) are on the level of objects and have some overlap in images.

It is interesting to note the differences in the expressions from REFCOCO and GREXP, the latter on average being almost 5 token longer. Figure 3 gives representative examples. We can speculate that the different task descriptions ("refer to this object" vs. "produce an unambiguous description") and the different settings (live to a partner vs. offline, only validated later) may have caused this. As we will see below, the GREXP descriptions did indeed cause more problems to our approach, which is meant for reference in interaction.

5 Training the Word/Object Classifiers

The basis of the approach we use are the classifiers that link words and images. These need to be trained from data; more specifically, from pairings of image regions and referring expressions, as provided by the corpora described in the previous section.

Representing Image Regions The first step is to represent the information from the image regions. We use a deep convolutional neural network, "GoogLeNet" (Szegedy et al., 2015), that was trained on data from the Large Scale Visual Recognition Challenge 2014 (ILSVRC2014) from the ImageNet corpus (Deng et al., 2009) to extract features.[5] It was optimised to recognise categories from that challenge, which are different from those occurring in either SAIAPR or COCO, but in any case we only use the final fully-connected layer before the classification layer, to give us a 1024 dimensional representation of the region. We augment this with 7 features that encode information about the region relative to the image: the (relative) coordinates of two corners, its (relative) area, distance to the center, and orientation of the image. The full representation hence is a vector of 1031 features. (See also Figure 1 above.)

Selecting candidate words How do we select the words for which we train perceptual classifiers? There is a technical consideration to be made here and a semantic one. The technical consideration is that we need sufficient training data for the classifiers, and so can only practically train classifiers for words that occur often enough in the training corpus. We set a threshold here of a minimum of 40 occurences in the training corpus, determined empirically on the validation set to provide a good tradeoff between vocabulary coverage and number of training instances.

The semantic consideration is that intuitively, the approach does not seem appropriate for all types of words; where it might make sense for attributes and category names to be modelled as image classifiers, it does less so for prepositions and other function words. Nevertheless, for now, we make the assumption that all words in a referring expression contribute information to the *visual* identification of its referent. We discuss the consequences of this decision below.

This assumption is violated in a different way in phrases that refer via a landmark, such as in "the thing next to the woman with the blue shirt". Here we cannot assume for example that the referent region provides a good instance of "blue" (since it is not the target object in the region that is described as blue), and so we exclude such phrases from the training set (by looking for a small set of expressions such as "left of", "behind", etc.; see appendix for a full list). This reduces the train-

[5] http://www.image-net.org/challenges/LSVRC/2014/.
We use the sklearn-theano (http://sklearn-theano.github.io/feature_extraction/index.html#feature-extraction) port of the Caffe replication and re-training (https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet) of this network structure.

ing portions of REFERIT, REFCOCO and GREXP to 86%, 95%, and 82% of their original size, respectively (counting referring expressions, not tokens).

Now that we have decided on the set of words for which to train classifiers, how do we assemble the training data?

Positive Instances Getting positive instances from the corpus is straightforward: We pair each word in a referring expression with the representation of the region it refers to. That is, if the word "left" occurs 20,000 times in expressions in the training corpus, we have 20,000 positive instances for training its classifier.

Negative Instances Acquiring negative instances is less straightforward. The corpus does not record inappropriate uses of a word, or 'negative referring expressions' (as in "this is not a red chair"). To create negative instances, we make a second assumption which again is not generally correct, namely that when a word was *never* in the corpus used to refer to an object, this object can serve as a negative example for that word/object classifier. In the experiments reported below, we randomly selected 5 image regions from the training corpus whose referring expressions (if there were any) did not contain the word in question.[6]

The classifiers Following this regime, we train binary logistic regression classifiers (with $\ell 1$ regularisation) on the visual object features representations, for all words that occurred at least 40 times in the respective training corpus.[7]

To summarise, we train separate binary classifiers for each word (not making any a-priori distinction between function words and others, or attribute labels and category labels), giving them the task to predict how likely it would be that the word they represent would be used to refer to the image region they are given. All classifiers are presented during training with data sets with the same balance of positive and negative examples (here, a fixed ratio of 1 positive to 5 negative). Hence, the classifiers themselves do not reflect any word frequency effects; our claim (to be validated in future work) is that any potential effects of this type are better modelled separately.

	%tst	acc	mrr	arc	>0	acc
REFERIT	1.00	0.65	0.79	0.89	0.97	0.67
REFERIT; NR	0.86	0.68	0.82	0.91	0.97	**0.71**
(Hu et al., 2015)	–	0.73	–	–	–	–
REFCOCO	1.00	0.61	0.77	0.91	0.98	0.62
REFCOCO; NR	0.94	0.63	0.78	0.92	0.98	**0.64**
(Mao et al., 2015)	–	0.70	–	–	–	–
GREXP	1.00	0.43	0.65	0.86	1.00	0.43
GREXP; NR	0.82	0.45	0.67	0.88	1.00	**0.45**
(Mao et al., 2015)	–	0.61	–	–	–	–

Table 1: Results; separately by corpus. See text for description of columns and rows.

6 Experiments

The task in our experiments is the following: Given an image I together with bounding boxes of regions $(bb_1, \ldots, bb_n)$ within it, and a referring expression e, predict which of these regions contains the referent of the expression.

By Corpus We start with training and testing models for all three corpora (REFERIT, REFCOCO, GREXP) separately. But first, we establish some baselines. The first is just randomly picking one of the candidate regions. The second is a 1-rule classifier that picks the largest region. The respective accuracies on the corpora are as follows: REFERIT 0.20/0.19; REFCOCO 0.16/0.23; GREXP 0.19/0.20.

Training on the training sets of REFERIT, REFCOCO and GREX with the regime described above (min. 40 occurrences) gives us classifiers for 429, 503, and 682 words, respectively. Table 1 shows the evaluation on the respective test parts: accuracy (*acc*), mean reciprocal rank (*mrr*) and for how much of the expression, on average, a word classifier is present (*arc*). '>0' shows how much of the testcorpus is left if expressions are filtered out for which not even a single word is the model (which we evaluate by default as false), and accuracy for that reduced set. The 'NR' rows give the same numbers for reduced test sets in which all relational expressions have been removed; '%tst' shows how much of a reduction that is relative to the full testset. The rows with the citations give the best reported results from the literature.[8]

As this shows, in most cases we come close, but do not quite reach these results. The distance is the biggest for GREXP with its much longer expressions. As discussed above, not only are the descriptions longer on average in this corpus, the

[6] This approach is inspired by the negative sampling technique of Mikolov et al. (2013) for training textual word embeddings.

[7] We used the implementation in the `scikit learn` package (Pedregosa et al., 2011).

[8] Using a different split than (Mao et al., 2015), as their train/test set overlaps on the level of images.

vocabulary size is also much higher. Many of the descriptions contain action descriptions ("the man smiling at the woman"), which do not seem to be as helpful to our model. Overall, the expressions in this corpus do appear to be more like 'mini-captions' describing the region rather than referring expressions that efficiently single it out among the set of distractors; our model tries to capture the latter.

Combining Corpora A nice effect of our setup is that we can freely mix the corpora for training, as image regions are represented in the same way regardless of source corpus, and we can combine occurrences of a word across corpora. We tested combining the testsets of REFERIT and REFCOCO (RI+RC in the Table below), REFCOCO and GREXP (RC+GR), and all three (REFERIT, REFCOCO, and GREXP; RI+RC+GR), yielding models for 793, 933, 1215 words, respectively (with the same "min. 40 occurrences" criterion). For all testsets, the results were at least stable compared to Table 1, for some they improved. For reasons of space, we only show the improvements here.

	%tst	acc	mrr	arc	>0	acc
RI+RC/RC	1.00	**0.63**	0.78	0.92	0.98	**0.64**
RI+RC/RC; NR	0.94	**0.65**	0.79	0.93	0.98	**0.66**
RI+RC+GR/RC	1.00	**0.63**	0.78	0.94	0.99	**0.64**
RI+RC+GR/RC; NR	0.94	**0.65**	0.79	0.95	0.99	**0.66**
RI+RC+GR/GR	1.00	**0.47**	0.68	0.90	1.00	**0.47**
RI+RC+GR/GR; NR	0.82	**0.49**	0.70	0.91	1.00	**0.49**

Table 2: Results, combined corpora

Computed Region Proposals Here, we cannot expect the system to retrieve exactly the ground truth bounding box, since we cannot expect the set of automatically computed regions to contain it. We follow Mao et al. (2015) in using *intersection over union* (IoU) as metric (the size of the intersective area between candidate and ground truth bounding box normalised by the size of the union) and taking an IoU ≥ 0.5 of the top candidate as a threshold for success (P@1). As a more relaxed metric, we also count for the SAIAPR proposals (of which there are 100 per image) as success when at least one among the top 10 candidates exceeds this IoU threshold (R@10). (For MSCOCO, there are only slightly above 5 proposals per image on average, so computing this more relaxed measure does not make sense.) The random baseline (RND) is computed by applying the P@1 criterion to a randomly picked region proposal. (That it is higher than 1/#regions for SAIAPR shows that the regions cluster around objects.)

	RP@1	RP@10	rnd
REFERIT	0.09	0.24	0.03
REFERIT; NR	**0.10**	0.26	0.03
(Hu et al., 2015)	0.18	0.45	
REFCOCO	0.52	–	0.17
REFCOCO; NR	**0.54**	–	0.17
(Mao et al., 2015)	0.52		
GREXP	0.36	–	0.16
GREXP; NR	**0.37**	–	0.17
(Mao et al., 2015)	0.45		

Table 3: Results on region proposals

With the higher quality proposals provided for the MSCOCO data, and the shorter, more prototypical referring expressions from REFCOCO, we narrowly beat the reported results. (Again, note that we use a different split that ensures separation on the level of images between training and test.) (Hu et al., 2015) performs relatively better on the region proposals (the gap is wider), on GREXP, we come relatively closer using these proposals. We can speculate that using automatically computed boxes of a lower selectivity (REFERIT) shifts the balance between needing to actually recognise the image and getting information from the shape and position of the box (our *positional* features; see Section 5).

Ablation Experiments To get an idea about what the classifiers actually pick up on, we trained variants given only the positional features (POS columns below in Table 4) and only object features (NOPOS columns). We also applied a variant of the model with only the top 20 classifiers (in terms of number of positive training examples; TOP20). We only show accuracy here, and repeat the relevant numbers from Table 1 for comparison (FULL).

	nopos	pos	full	top20
RI	0.53	0.60	0.65	0.46
RI; NR	0.56	0.62	0.68	0.48
RC	0.44	0.55	0.61	0.52
RC; NR	0.45	0.57	0.63	0.53

Table 4: Results with reduced models

This table shows an interesting pattern. To a large extent, the object image features and the positional features seem to carry redundant information, with the latter on their own performing better than the former on their own. The full model, however, still gains something from the combination

of the feature types. The top-20 classifiers (and consequently, top 20 most frequent words) alone reach decent performance (the numbers are shown for the full test set here; if reduced to only utterances where at least one word is known, the numbers rise, but the reduction of the testset is much more severe than for the full models with much larger vocabulary).

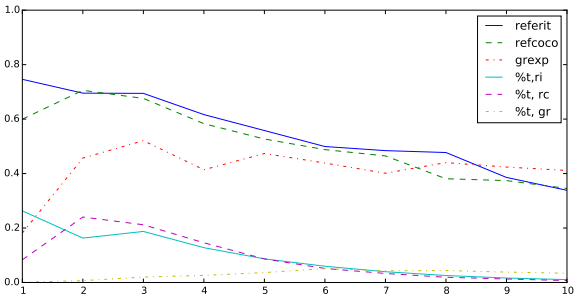


Figure 4: Accuracy by expression length (top 3 lines); percentage of expressions with this length (lower 3 lines).

Error Analysis Figure 4 shows the accuracy of the model split by length of the referring expression (top lines; lower lines show the proportion of expression of this length in the whole corpus). The pattern is similar for all corpora (but less pronounced for GREXP): shorter utterances fare better.

Manual inspection of the errors made by the system further corroborates the suspicion that composition as done here neglects too much of the internal structure of the expression. An example from REFERIT where we get a wrong prediction is "second person from left". The model clearly does not have a notion of counting, and here it wrongly selects the leftmost person. In a similar vein, we gave results above for a testset where spatial relations where removed, but other forms of relation (e.g., "child sitting on womans lap") that weren't modelled still remain in the corpus.

We see as an advantage of the model that we can inspect words individually. Given the performance of short utterances, we can conclude that the word/object classifiers themselves perform reasonably well. This seems to be somewhat independent of the number of training examples they received. Figure 5 shows, for REFERIT, # training instances (x-axis) vs. average accuracy on the validation set, for the whole vocabulary. As this shows, the classifiers tend to get better with more training instances, but there are good ones even with very little training material.

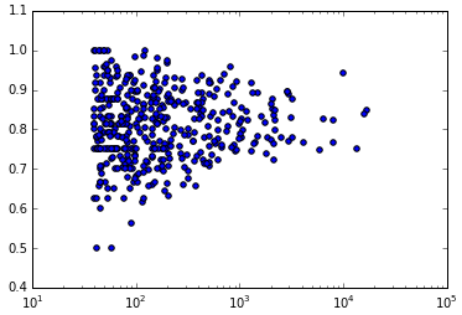


Figure 5: Average accuracy vs. # train instanc.

Mean average precision (i.e., area under the precision / recall curve) over all classifiers (exemplarily computed for the RI+RC set, 793 words) is 0.73 (std 0.15). Interestingly, the 155 classifiers in the top range (average precision over 0.85) are almost all for concrete nouns; the 128 worst performing ones (below 0.60) are mostly other parts of speech. (See appendix.) This is, to a degree, as expected: our assumption behind training classifiers for *all* ocurring words and not pre-filtering based on their part-of-speech or prior hypotheses about visual relevance was that words that can occur in all kinds of visual contexts will lead to classifiers whose contributions cancel out across all candidate objects in a scene.

However, the mean average precision of the classifiers for colour words is also relatively low at 0.6 (std 0.08), for positional words ("left", "right", "center", etc.) it is 0.54 (std 0.1). This might suggest that the features we take from the CNN might indeed be more appropriate for tasks close to what they were originally trained on, namely category and not attribute prediction. We will explore this in future work.

7 Conclusions

We have shown that the "words-as-classifiers" model scales up to a larger set of object types with a much larger variety in appearance (SAIAPR and MSCOCO); to a larger vocabulary and much less restricted expressions (REFERIT, REFCOCO, GREXP); and to use of automatically learned feature types (from a CNN). It achieves results that are comparable to those of more complex models.

We see as advantage that the model we use is "transparent" and modular. Its basis, the word/object classifiers, ties in more directly with

more standard approaches to semantic analysis and composition. Here, we have disregarded much of the internal structure of the expressions. But there is a clear path for bringing it back in, by defining other composition types for other construction types and different word models for other word types. Kennington and Schlangen (2015) do this for spatial relations in their simpler domain; for our domain, new and more richly annotated data such as VISUALgenome looks promising for learning a wide variety of relations.[9] The use of denotations / extensions might make possible transfer of methods from extensional semantics, e.g. for the addition of operators such as negation or generalised quantifiers. The design of the model, as mentioned in the introduction, makes it amenable for use in interactive systems that learn; we are currently exploring this avenue. Lastly, the word/object classifiers also show promise in the reverse task, generation of referring expressions (Zarrieß and Schlangen, 2016).

All this is future work. In its current state—besides, we believe, strongly motivating this future work—, we hope that the model can also serve as a strong baseline to other future approaches to reference resolution, as it is conceptually simple and easy to implement.

Acknowledgments

We thank Hu et al. (2016), Mao et al. (2016) and Tamara Berg for giving us access to their data. Thanks are also due to the anonymous reviewers for their very insightful comments. We acknowledge support by the Cluster of Excellence "Cognitive Interaction Technology" (CITEC; EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG), and by the DUEL project, also funded by DFG (grant SCHL 845/5-1).

References

Robin Cooper and Jonathan Ginzburg. 2015. Type theory with records for natural language semantics. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantic Theory 2nd edition*. Wiley-Blackwell.

Jia Deng, W. Dong, Richard Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 100–105, Beijing, China, July. Association for Computational Linguistics.

Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. 2014. Scalable Object Detection Using Deep Neural Networks. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2155–2162.

Hugo Jair Escalante, Carlos a. Hernández, Jesus a. Gonzalez, a. López-López, Manuel Montes, Eduardo F. Morales, L. Enrique Sucar, Luis Villaseñor, and Michael Grubinger. 2010. The segmented and annotated IAPR TC-12 benchmark. *Computer Vision and Image Understanding*, 114(4):419–428.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John Platt, Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *Proceedings of CVPR*, Boston, MA, USA, June. IEEE.

L. T. F. Gamut. 1991. *Logic, Language and Meaning: Intensional Logic and Logical Grammar*, volume 2. Chicago University Press, Chicago.

Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The IAPR TC-12 benchmark: a new evaluation resource for visual information systems. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2006)*, pages 13–23, Genoa, Italy.

Stevan Harnad. 1990. The symbol grounding problem. *Physica D*, 42:335–346.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2015. Natural language object retrieval. *ArXiv / CoRR*, abs/1511.04164.

Ronghang Hu, Huazhe Xu, Marcus Rohrbach, Jiashi Feng, Kate Saenko, and Trevor Darrell. 2016. Natural language object retrieval. In *Proceedings of CVPR 2016*, Las Vegas, USA, June.

Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara L Berg. 2014. ReferItGame: Referring to Objects in Photographs of Natural Scenes. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 787–798, Doha, Qatar.

Casey Kennington and David Schlangen. 2015. Simple learning and compositional application of perceptually grounded word meanings for incremental

[9] http://visualgenome.org/

reference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 292–301, Beijing, China, July. Association for Computational Linguistics.

Jayant Krishnamurthy and Thomas Kollar. 2013. Jointly learning to parse and perceive: Connecting natural language to the physical world. *Transactions of the Association for Computational Linguistics*, 1:193–206.

Staffan Larsson. 2015. Formal semantics for perceptual classification. *Journal of logic and computation*, 25(2):335–369.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C.Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision ECCV 2014*, volume 8693, pages 740–755. Springer International Publishing.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2015. Generation and comprehension of unambiguous object descriptions. *ArXiv / CoRR*, abs/1511.02283.

Junhua Mao, Jonathan Huang, Alexander Toshev, Oana Camburu, Alan L. Yuille, and Kevin Murphy. 2016. Generation and comprehension of unambiguous object descriptions. In *Proceedings of CVPR 2016*, Las Vegas, USA, June.

Cynthia Matuszek, Nicholas Fitzgerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A Joint Model of Language and Perception for Grounded Attribute Learning. In *Proceedings of the International Conference on Machine Learning (ICML 2012)*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013 (NIPS 2013)*, pages 3111–3119, Lake Tahoe, Nevada, USA.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Deb Roy, Peter Gorniak, Niloy Mukherjee, and Josh Juster. 2002. A trainable spoken language understanding system for visual object selection. In *Proceedings of the International Conference on Speech and Language Processing 2002 (ICSLP 2002)*, Colorado, USA.

Deb K. Roy. 2002. Learning visually-grounded words and syntax for a scene description task. *Computer Speech and Language*, 16(3).

Deb Roy. 2005. Grounding words in perception and action: Computational insights. *Trends in Cognitive Sciene*, 9(8):389–396.

Richard Socher, Andrej Karpathy, Quoc V Le, Christopher D Manning, and Andrew Y Ng. 2014. Grounded Compositional Semantics for Finding and Describing Images with Sentences. *Transactions of the ACL (TACL)*.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *CVPR 2015*, Boston, MA, USA, June.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. 2011. Understanding Natural Language Commands for Robotic Navigation and Mobile Manipulation. In *AAAI Conference on Artificial Intelligence*, pages 1507–1514.

Richmond H. Thomason, editor. 1974. *Formal Philosophy: Selected Papers of Richard Montague*. Yale University Press, New Haven and London.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*.

Sina Zarrieß and David Schlangen. 2016. Easy things first: Installments improve referring expression generation for objects in photographs. In *Proceedings of ACL 2016*, Berlin, Germany, August.

C Lawrence Zitnick and Piotr Dollár. 2014. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer.

A Supplemental Material

Filtering relational expressions As described above, we filter out all referring expressions during training that contain either of the following tokens:

```
RELWORDS = ['below',
            'above',
            'between',
            'not',
            'behind',
            'under',
            'underneath',
            'front of',
            'right of',
            'left of',
            'ontop of',
            'next to',
            'middle of']
```

Average Precision See Section 6.
Classifiers with average precision over 0.85:

```
['giraffe', 'coffee', 'court', 'riding', 'penguin', 'balloon', 'ball',
'mug', 'turtle', 'tennis', 'beer', 'seal', 'cow', 'bird', 'horse',
'drink', 'koala', 'sheep', 'ceiling', 'parrot', 'bike', 'cactus',
'sun', 'smoke', 'llama', 'fruit', 'ruins', 'waterfall', 'nightstand',
'books', 'night', 'coke', 'skirt', 'leaf', 'wheel', 'label', 'pot',
'animals', 'cup', 'tablecloth', 'pillar', 'flag', 'field', 'monkey',
'bowl', 'curtain', 'plate', 'van', 'surfboard', 'bottle', 'fish',
'umbrella', 'bus', 'shirtless', 'train', 'bed', 'painting', 'lamp',
'metal', 'paper', 'sky', 'luggage', 'player', 'face', 'going', 'desk',
'ship', 'raft', 'lying', 'vehicle', 'trunk', 'couch', 'palm', 'dress',
'doors', 'fountain', 'column', 'cars', 'flowers', 'tire', 'plane',
'against', 'bunch', 'car', 'shelf', 'bunk', 'boat', 'dog', 'vase',
'animal', 'pack', 'anyone', 'clock', 'glass', 'tile', 'window',
'chair', 'phone', 'across', 'cake', 'branches', 'bicycle', 'snow',
'windows', 'book', 'curtains', 'bear', 'guitar', 'dish', 'both',
'tower', 'truck', 'bridge', 'creepy', 'cloud', 'suit', 'stool', 'tv',
'flower', 'seat', 'buildings', 'shoes', 'bread', 'hut', 'donkey',
'had', 'were', 'fire', 'food', 'turned', 'mountains', 'city', 'range',
'inside', 'carpet', 'beach', 'walls', 'ice', 'crowd', 'mirror',
'brush', 'road', 'anything', 'blanket', 'clouds', 'island',
'building', 'door', '4th', 'stripes', 'bottles', 'cross', 'gold',
'smiling', 'pillow']
```

Classifiers with average precision below 0.6:

```
['shadow', "woman's", 'was', 'bright', 'lol', 'blue', 'her', 'yes',
'blk', 'this', 'from', 'almost', 'colored', 'looking', 'lighter',
'far', 'foreground', 'yellow', 'looks', 'very', 'second', 'its',
'dat', 'stack', 'dudes', 'men', 'him', 'arm', 'smaller', 'half',
'piece', 'out', 'item', 'line', 'stuff', 'he', 'spot', 'green',
'head', 'see', 'be', 'black', 'think', 'leg', 'way', 'women',
'furthest', 'rt', 'most', 'big', 'grey', 'only', 'like', 'corner',
'picture', 'shoulder', 'no', 'spiders', 'n', 'has', 'his', 'we',
'bit', 'spider', 'guys', '2', 'portion', 'are', 'section', 'us',
'towards', 'sorry', 'where', 'small', 'gray', 'image', 'but',
'something', 'center', 'i', 'closest', 'first', 'middle', 'those',
'edge', 'there', 'or', 'white', '-', 'little', 'them', 'barely',
'brown', 'all', 'mid', 'is', 'thing', 'dark', 'by', 'back', 'with',
'other', 'near', 'two', 'screen', 'so', 'front', 'you', 'photo', 'up',
'one', 'it', 'space', 'okay', 'side', 'click', 'part', 'pic', 'at',
'that', 'area', 'directly', 'in', 'on', 'and', 'to', 'just', 'of']
```

Code The code required for reproducing the results reported here can be found at `https://github.com/dsg-bielefeld/image_wac`.

RBPB: Regularization-Based Pattern Balancing Method for Event Extraction

Lei Sha[1,2], **Jing Liu**[2], **Chin-Yew Lin**[2], **Sujian Li**[1], **Baobao Chang**[1], **Zhifang Sui**[1]
[1] Key Laboratory of Computational Linguistics, Ministry of Education
School of Electronics Engineering and Computer Science, Peking University
[2]Knowledge Computing Group, Microsoft Research
[1]{shalei, lisujian, chbb, szf}@pku.edu.cn
[2]{liudani,cyl}@microsoft.com

Abstract

Event extraction is a particularly challenging information extraction task, which intends to identify and classify event triggers and arguments from raw text. In recent works, when determining event types (trigger classification), most of the works are either pattern-only or feature-only. However, although patterns cannot cover all representations of an event, it is still a very important feature. In addition, when identifying and classifying arguments, previous works consider each candidate argument separately while ignoring the relationship between arguments. This paper proposes a **R**egularization-**B**ased **P**attern **B**alancing Method (RBPB). Inspired by the progress in representation learning, we use trigger embedding, sentence-level embedding and pattern features together as our features for trigger classification so that the effect of patterns and other useful features can be balanced. In addition, RBPB uses a regularization method to take advantage of the relationship between arguments. Experiments show that we achieve results better than current state-of-art equivalents.

1 Introduction

Event extraction has become a popular research topic in the area of information extraction. ACE 2005 defines event extraction task[1] as three sub-tasks: identifying the trigger of an event, identifying the arguments of the event, and distinguishing their corresponding roles. As an example in Figure 1, there is an "Attack" event triggered by "tear through" with three arguments. Each argument has one role.

In the trigger classification stage, some previous approaches (Grishman et al., 2005; Ji and Grishman, 2008; Liao and Grishman, 2010; Huang and Riloff, 2012) use patterns to decide the types of event triggers. However, pattern-based approaches suffer from low recall since real world events usually have a large variety of representations. Some other approaches (Hong et al., 2011; Li et al., 2013; Lu and Roth, 2012) identify and classify event triggers using a large set of features without using patterns. Although these features can be very helpful, patterns are still indispensable in many cases because they can identify a trigger with the correct event type with more than 96% accuracy according to our data analysis on ACE 2005 data sets.

In argument identification and classification, most approaches identify each candidate argument separately without considering the relation between arguments. We define two kinds of argument relations here: (1) Positive correlation: if one candidate argument belongs to one event, then the other is more likely to belong to the same event. For example, in Figure 1, the entity "a waiting shed" shares a common dependency head "tore" with "a powerful bomb", so when the latter entity is identified as an argument, the former is more likely to be identified. (2) Negative correlation: if one candidate argument belongs to one event, then the other is less likely to belong to the same event. For example, in Figure 1, "bus" is irrelevant to other arguments, so if other entities are identified as arguments "bus" is less likely to be identified. Note that although all the above relation examples have something to do with dependency analysis, the positive/negative relationship depends not only on dependency parsing, but many other aspects as well.

[1]http://www.itl.nist.gov/iad/mig/tests/ace/2005/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1224–1234,
Berlin, Germany, August 7-12, 2016.

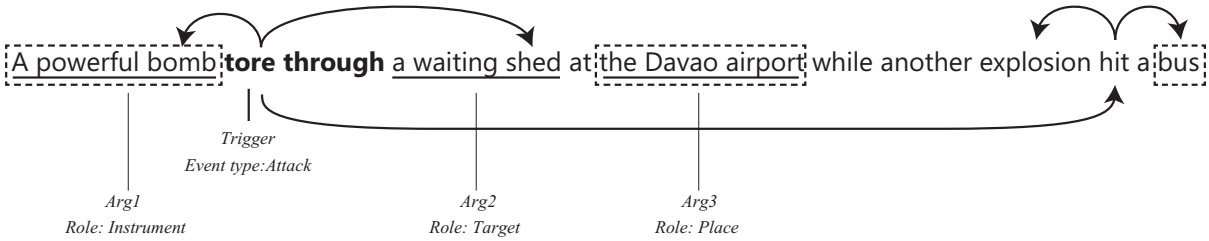


Figure 1: Event example: This is an event trigger by "tear through" with three arguments

In this paper, we propose using both patterns and elaborately designed features simultaneously to identify and classify event triggers. In addition, we propose using a regularization method to model the relationship between candidate arguments to improve the performance of argument identification. Our method is called Regularization-Based Pattern Balancing Method method.

The contributions of this paper are as follows:

- Inspired by the progress of representation learning, we use trigger embedding, sentence-level embedding, and pattern features together as the our features for balancing.
- We proposed a regularization-based method in order to make use of the relationship between candidate arguments. Our experiments on the ACE 2005 data set show that the regularization method does improve the performance of argument identification.

2 Related Work

There is a large body of previous work devoted to event extraction. Many traditional works focus on using pattern based methods for identifying event type (Kim and Moldovan, 1993; Riloff and others, 1993; Soderland et al., 1995; Huffman, 1996; Freitag, 1998b; Ciravegna and others, 2001; Califf and Mooney, 2003; Riloff, 1996; Riloff et al., 1999; Yangarber et al., 2000; Sudo et al., 2003; Stevenson and Greenwood, 2005; Grishman et al., 2005; Ji and Grishman, 2008; Liao and Grishman, 2010; Huang and Riloff, 2012). (Shinyama and Sekine, 2006; Sekine, 2006) are unsupervised methods of extracting patterns from open domain texts. Pattern is not always enough, although some methods (Huang and Riloff, 2012; Liu and Strzalkowski, 2012) use bootstrapping to get more patterns.

There are also feature-based classification methods (Freitag, 1998a; Chieu and Ng, 2002; Finn and Kushmerick, 2004; Li et al., 2005; Yu et al., 2005). Apart from the above methods, weakly supervised training (pattern-based and rule-based) of event extraction systems have also been explored (Riloff, 1996; Riloff et al., 1999; Yangarber et al., 2000; Sudo et al., 2003; Stevenson and Greenwood, 2005; Patwardhan and Riloff, 2007; Chambers and Jurafsky, 2011). In some of these systems, human work is needed to delete some nonsense patterns or rules. Other methods (Gu and Cercone, 2006; Patwardhan and Riloff, 2009) consider broader context when deciding on role fillers. Other systems take the whole discourse feature into consideration, such as (Maslennikov and Chua, 2007; Liao and Grishman, 2010; Hong et al., 2011; Huang and Riloff, 2011). Ji and Grishman (2008) even consider topic-related documents, proposing a cross-document method. (Liao and Grishman, 2010; Hong et al., 2011) use a series of global features (for example, the occurrence of one event type lead to the occurrence of another) to improve role assignment and event classification performance. Joint models (Li et al., 2013; Lu and Roth, 2012) are also considered an effective solution. (Li et al., 2013) make full use of the lexical and contextual features to get better results. The semi-CRF based method (Lu and Roth, 2012) trains separate models for each event type, which requires a lot of training data.

The dynamic multi-pooling convolutional neural network (DMCNN) (Chen et al., 2015) is currently the only widely used deep neural network based approach. DMCNN is mainly used to model contextual features. However, DMCNN still does not consider argument-argument interactions.

In summary, most of the above works are either pattern-only or features-only. Moreover, all of these methods consider arguments sepa-

rately while ignoring the relationship between arguments, which is also important for argument identification. Even the joint method (Li et al., 2013) does not model argument relations directly. We use trigger embedding, sentence-level embedding, and pattern features together as our features for trigger classification and design a regularization-based method to solve the two problems.

3 ACE Event Extraction Task

Automatic Content Extraction (ACE) is an event extraction task. It annotates 8 types and 33 subtypes of events. ACE defines the following terminologies:

- Entity: an object or a set of objects in one of the semantic categories of interest
- Entity mention: a reference to an entity, usually a noun phrase (NP)
- Event trigger: the main word which most clearly expresses an event occurrence
- Event arguments: the entity mentions that are involved in an event
- Argument roles: the relation of arguments to the event where they participate, with 35 total possible roles
- Event mention: a phrase or sentence within which an event is described, including trigger and arguments

Given an English document, an event extraction system should identify event triggers with their subtypes and arguments from each sentence. An example is shown in Figure 1. There is an "Attack" event triggered by "tear through" with three arguments. Each argument has a role type such as "Instrument", "Target", etc.

For evaluation, we follow previous works (Ji and Grishman, 2008; Liao and Grishman, 2010; Li et al., 2013) to use the following criteria to determine the correctness of the predicted event mentions.

- A trigger is considered to be correct if and only if its event type and offsets (position in the sentence) can match the reference trigger;
- An argument is correctly identified if and only if its event type and offsets can match any reference arguments;
- An argument is correctly identified and classified if and only if its event type, offsets, and role match any of the reference arguments.

4 Baseline: JET Extractor for Events

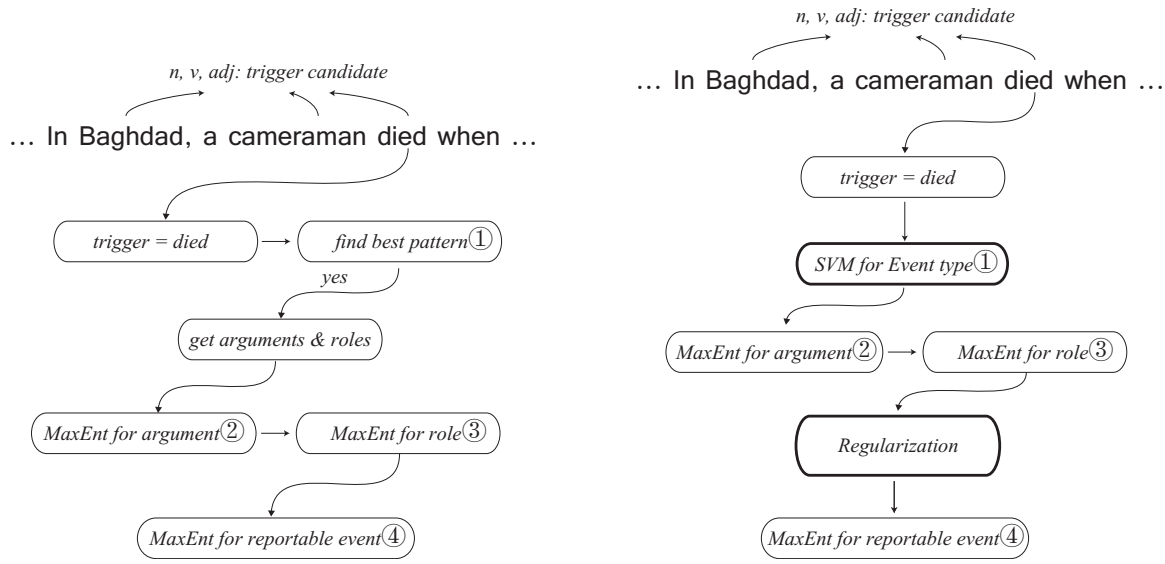
Many previous works take JET as their baseline system, including (Ji and Grishman, 2008), (Liao and Grishman, 2010), (Li et al., 2013). JET extracts events independently for each sentence. This system uses pattern matching to predict trigger and event types, then uses statistical modeling to identify and classify arguments. For each event mention in the training corpus of ACE, the patterns are constructed based on the sequences of constituent heads separating the trigger and arguments. After that, three Maximum Entropy classifiers are trained using lexical features.

- Argument Classifier: to distinguish arguments from non-arguments
- Role Classifier: to label arguments with an argument role
- Reportable-Event Classifier: to determine whether there is a reportable event mentioned (worth being taken as an event mention) according to the trigger, event type, and a set of arguments

Figure 2(a) shows the whole test procedure. In the test procedure, each sentence is scanned for nouns, verbs and adjectives as trigger candidates. When a trigger candidate is found, the system tries to match the context of the trigger against the set of patterns associated with that trigger. If this pattern matching process is successful, the best pattern will assign some of the entity mentions in the sentence as arguments of a potential event mention. Then JET uses the argument classifier to judge if the remaining entity mentions should also be identified. If yes, JET uses the role classifier to assign it a role. Finally, the reportable-event classifier is applied to decide whether this event mention should be reported.

5 Regularization-Based Pattern Balancing Method

Different with JET, as illustrated in Figure 2(b), our work introduces two major improvements: (1) balance the effect of patterns and other features (2)



(a) The flow chart of JET

(b) The flow chart of our approach

Figure 2: The left is the flow chart for JET. The right is the flow chart for our approach. The thick line block is our contribution

use a regularization-based method to make full use of the relation between candidate arguments.

The thick-edge blocks in Figure 2(b) represent our improvements. Since JET only uses patterns when predicting the event type, we use a SVM classifier to decide each candidate trigger's event type (classify the trigger). This classifier uses trigger embedding, sentence-level embedding and pattern features together for balancing. After the outputs of argument and role classifier are calculated, we make use of the argument relationship to regularize for a better result.

5.1 Balancing the Pattern effects

Deciding the event type is the same as classifying an event trigger. JET only uses patterns in this step: for a candidate trigger, we find that the best matched pattern and the corresponding event type are assigned to this trigger. We propose using feature-based methods while not ignoring the effect of patterns. Inspired by progress in representation learning, we use trigger embedding, sentence-level embedding and pattern embedding together as our features.

A pattern example is as follows:

(weapon) tore [through] (building) at (place) $\Rightarrow$ Attack{Roles...}

where each pair of round brackets represents an entity and the word inside is one of the 18 entity types defined by UIUC NER Tool[2]. The word in the square brackets can choose to exist or not. After the right arrow there is an event schema, which can tell us what kind of event this is and which roles each entity should take.

Each pattern has a corresponding event type. A candidate trigger may match more than one pattern so that it has an event type distribution. Assume that there are N_T event types in total, we denote the pattern feature vector (namely, the event type's probability distribution calculated by the trigger's pattern set) as $P_E \in \mathbb{R}^{N_T}$, which is calculated by Eq 1.

$$P_E(i) = \frac{\#(\text{matched patterns of event type } i)}{\#(\text{all matched patterns})} \quad (1)$$

Trigger embeddings are obtained using WORD2VEC[3] with the default "text8" training text data with length 200.

Since all of the NPs are potential roles in the event, they must contain the main information of the event. We extract all the NPs in the sentence and take the average word embedding of these NPs' head word as the sentence-level embedding. For example, in Figure 1, these NPs' head words are bomb, shed, and airport.

Pattern feature vectors, as distributions of event types over patterns, are also composed using

[2]http://cogcomp.cs.illinois.edu/page/software_view/NETagger
[3]http://code.google.com/p/word2vec/

continuous real values, which allows them to be viewed as a kind of pattern embedding and treated similarly to trigger and sentence embedding.

5.2 Capturing the Relationship Between Arguments

We find that there are two typical relations between candidate arguments: (1) positive correlation: if one candidate argument belongs to one event, then the other is more likely to belong to the same event; (2) negative correlation: if one candidate argument belongs to one event, then the other is less likely to belong to the same event.

We calculate a score for all the candidate arguments in a sentence to judge the quality of the argument identification and classification. For capturing the two kinds of relations, we intend to make that (1) the more positive relations the chosen arguments have, the higher the score is; (2) the more negative relations the chosen arguments have, the lower the score is.

For a trigger, if there are n candidate arguments, we set a $n \times n$ matrix C to represent the relationship between arguments. If $C_{i,j} = 1$, then argument i and argument j should belong to the same event. If $C_{i,j} = -1$, then argument i and argument j cannot belong to the same event. We will illustrate how to get matrix C in the next section.

We use a n-dim vector X to represent the identification result of arguments. Each entry of X is 0 or 1. 0 represents "noArg", 1 represents "arg". X can be assigned by maximizing $E(X)$ as defined by Eq 2.

$$\begin{aligned} X &= \underset{X}{\operatorname{argmax}}\, E(X) \\ E(X) &= \lambda_1 X^T C X + \lambda_2 P_{sum}^{arg} \\ &\quad + (1 - \lambda_1 - \lambda_2) P_{sum}^{role} \end{aligned} \tag{2}$$

Here, $X^T C X$ means adding up all the relationship values if the two arguments are identified. Hence, the more the identified arguments are related, the larger the value $X^T C X$ is. P_{sum}^{arg} is the sum of all chosen arguments' probabilities. The probability here is the output of the arguments' maximum entropy classifier. P_{sum}^{role} is the sum of all the classified roles' probabilities. The probability here is the output of the roles' maximum entropy classifier.

Eq 2 shows that while we should identify and classify the candidate arguments with a larger probability, the argument relationship evaluation should also be as large as possible. The arguments should also follow the following constraints. These constraints together with Eq 2 can make the argument identification and classification help each other for a better result.

- Each entity can only take one role
- Each role can belong to one or more entities
- The role assignment must follow the event schema of the corresponding type, which means that only the roles in the event schema can occur in the event mention

We use the Beam Search method to search for the optimal assignment X as is shown in Algorithm 1. The hyperparameters λ_1 and λ_2 can be chosen according to development set.

Input: Argument relationship matrix: C
the argument probabilities required by P_{sum}^{arg}
the role probabilities required by P_{sum}^{role}
Data: K: Beam size
n: Number of candidate arguments
Output: The best assignment X
Set beam $\mathcal{B} \leftarrow [\epsilon]$;
for $i \leftarrow 1 \cdots n$ **do**
  buf$\leftarrow \{z' \circ l | z' \in \mathcal{B}, l \in \{0, 1\}\}$;
  $\mathcal{B} \leftarrow [\epsilon]$;
  while $j \leftarrow 1 \cdots K$ **do**
    $x_{best} = \operatorname{argmax}_{x \in \text{buf}} E(x)$;
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_{best}\}$;
    buf$\leftarrow$buf$-\{x_{best}\}$;
  end
end
Sort $\mathcal{B}$ descendingly according to $E(X)$;
return $\mathcal{B}[0]$;

Algorithm 1: Beam Search decoding algorithm for event extraction. $\circ$ means to concatenate an element to the end of a vector.

5.2.1 Training the Argument Relationship Structure

The argument relationship matrix C is very important in the regularization process. We train a maximum entropy classifier to predict the connection between two entities. We intend to classify the entity pairs into three classes: positive correlation, negative correlation, and unclear correlation. The entity pairs in the ground truth events (in training

data) are used for our training data. We choose the following features:

- TRIGGER: the trigger of the event. The whole model is a pipelined model, so when classifying the argument relationship, the trigger has been identified and classified. So the "trigger" is a feature of the argument relation.
- ENTITY DISTANCE: the distance between the two candidate arguments in the sentence, namely the number of intervening words
- Whether the two candidate arguments occur on the same side of the trigger
- PARENT DEPENDENCY DISTANCE: the distance between the two candidate arguments' parents in the dependency parse tree, namely, the path length.
- PARENT POS: if the two candidate arguments share the same parent, take the common parent's POS tag as a feature
- Whether the two candidate arguments occur on the same side of the common parent if the two candidate arguments share the same parent

For an entity pair, if both of the entities belong to the same event's arguments, we take it as positive example. For each positive example, we randomly exchange one of the entities with an irrelevant entity (an irrelevant entity is in the same sentence as the event, but it is not the event's argument) to get a negative example. In the testing procedure, we predict the relationship between entity i and entity j using the maximum entropy classifier.

When the output of the maximum entropy classifier is around 0.5, it is not easy to figure out whether it is the first relation or the second. We call this kind of information "uncertain information"(unclear correlation). For better performance, we **strengthen** the certain information and weaken the uncertain information. We set two thresholds, if the output of the maximum entropy classifier is larger than 0.8, we set $C_{i,j} = 1$ (positive correlation), if the output is lower than 0.2, we set $C_{i,j} = -1$ (negative correlation), otherwise, we set $C_{i,j} = 0$ (unclear correlation). The strengthen mapping is similar to the hard tanh in neural network. If we do not do this, according to the experiment, the performance cannot beat most of the baselines since the uncertain information has very bad noise.

6 Experiments

6.1 Data

We utilize ACE 2005 data sets as our testbed. As is consistent with previous work, we randomly select 10 newswire texts from ACE 2005 training corpora as our development set, and then conduct blind test on a separate set of 40 ACE 2005 newswire texts. The remaining 529 documents in ACE training corpus are used as the training data.

The training dataset of the argument relationship matrix contains 5826 cases (2904 positive and 2922 negative) which are randomly generated according to the ground truth in the 529 training documents.

6.2 Systems to Compare

We compare our system against the following systems:

- **JET** is the baseline of (Grishman et al., 2005), we report the paper values of this method;
- **Cross-Document** is the method proposed by Ji and Grishman (2008), which uses topic-related documents to help extract events in the current document;
- **Cross-Event** is the method proposed by Liao and Grishman (2010), which uses document-level information to improve the performance of ACE event extraction.
- **Cross-Entity** is the method proposed by Hong et al. (2011), which extracts events using cross-entity inference.
- **Joint** is the method proposed by Li et al. (2013), which extracts events based on structure prediction. It is the best-reported structure-based system.
- **DMCNN** is the method proposed by Chen et al. (2015), which uses a dynamic multi-pooling convolutional neural network to extract events. It is the only neural network based method.

The Cross-Document, Cross-Event and Cross-Entity are all extensions of JET. Among these

Method	Trigger Classification			Argument Identification			Argument Role		
	P	R	F_1	P	R	F_1	P	R	F_1
JET	67.6	53.5	59.7	46.5	37.2	41.3	41.0	32.8	36.5
Cross-Event	68.7	**68.9**	68.8	50.9	49.7	50.3	45.1	44.1	44.6
Cross-Entity	72.9	64.3	68.3	53.4	52.9	53.1	51.6	45.5	48.3
Joint	73.7	62.3	67.5	**69.8**	47.9	56.8	**64.7**	44.4	52.7
DMCNN	**75.6**	63.6	**69.1**	68.8	51.9	59.1	62.2	46.9	53.5
RBPB(JET)	62.3	59.9	61.1	50.4	45.8	48.0	41.9	36.5	39.0
+ ET	66.7	65.9	66.3	60.6	56.7	58.6	49.2	48.3	48.7
+ Regu	67.2	61.7	64.3	62.8	57.5	60.0	52.6	48.4	50.4
+ ET + Regu	70.3	67.5	68.9	63.2	**59.4**	**61.2**	54.1	**53.5**	**53.8**

Table 1: Overall performance with gold-standard entities, timex, and values, the candidate arguments are annotated in ACE 2005. "ET" means the pattern balancing event type classifier, "Regu" means the regularization method

methods, Cross-Event, Cross-Entity, and DMCNN make use of the gold-standard entities, timex, and values annotated in the corpus as the argument candidates. Cross-Document uses the JET system to extract candidate arguments. Li et al. (2013) report the performance with both gold-standard argument candidates and predicted argument candidates. Therefore, we compare our results with methods based on gold argument candidates in Table 1 and methods based on predicted argument candidates in Table 2.

We have done a series of ablation experiments:

- **RBPB(JET)**: Our own implementation of JET
- **RBPB(JET) + ET**: Add pattern balanced event type classifier to RBPB(JET)
- **RBPB(JET) + Regu**: Add regularization mechanism to RBPB(JET)
- **RBPB(JET) + ET + Regu**: Add both pattern balanced event type classifier and regularization mechanism to RBPB(JET)

6.2.1 The Selection of Hyper-parameters

We tune the coefficients λ_1 and λ_2 of Eq 2 on the development set, and finally we set $\lambda_1 = 0.10$ and $\lambda_2 = 0.45$. Figure 3 shows the variation of argument identification's F_1 measure and argument classification's F_1 measure when we fix one parameter and change another. Note that the third coefficient $1 - \lambda_1 - \lambda_2$ must be positive, which is the reason why the curve decreases sharply when λ_2 is fixed and $\lambda_1 > 0.65$. Therefore, Figure 3 illustrates that the robustness of our method is very good, which means if the hyperparameters λ_1, λ_2 are larger or smaller, it will not affect the result very much.

6.3 Experiment Results

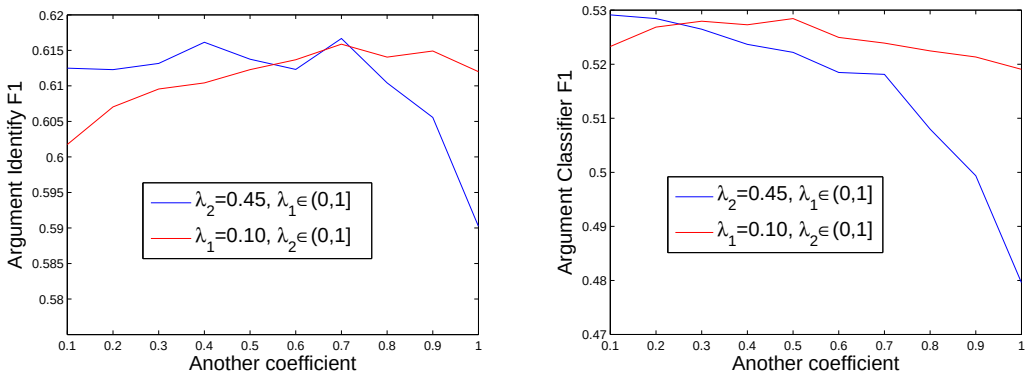
We conduct experiments to answer the following questions. (1) Can pattern balancing lead to a higher performance in trigger classification, argument identification, and classification while retaining the precision value? (2) Can the regularization step improve the performance of argument identification and classification?

Table 1 shows the overall performance on the blind test set. We compare our results with the JET baseline as well as the Cross-Event, Cross-Entity, and joint methods. When adding the event type classifier, in the line titled "+ ET", we see a significant increase in the three measures over the JET baseline in recall. Although our trigger's precision is lower than RBPB(JET), it gains 5.2% improvement on the trigger's F_1 measure, 10.6% improvement on argument identification's F_1 measure and 9.7% improvement on argument classification's F_1 measure. We also test the performance with argument candidates automatically extracted by JET in Table 2, our approach "+ ET" again significantly outperforms the JET baseline. Remarkably, our result is comparable with the Joint model although we only use lexical features.

The line titled "+ Regu" in Table 1 and Table 2 represents the performance when we only use the regularization method. In Table 1, Compared to the four baseline systems, the argument identifi-

Method	Trigger F_1	Arg id F_1	Arg id+cl F_1
JET	59.7	42.5	36.6
Cross-Document	67.3	46.2	42.6
Joint	65.6	-	41.8
RBPB(JET)	60.4	44.3	37.1
+ ET	66.0	47.8	39.7
+ Regu	64.8	54.6	42.0
+ ET + Regu	**67.8**	**55.4**	**43.8**

Table 2: Overall performance with predicted entities, timex, and values, the candidate arguments are extracted by JET. "ET" is the pattern balancing event type classifier, "Regu" is the regularization method



(a) Arg Identify (b) Arg Classify

Figure 3: The trend graph when fix one coefficient and change another

cation's F_1 measure of "+ Regu" is significantly higher. In Table 2, the "+ Regu" again gains a higher F_1 measure than the JET, Cross-Document, joint model baseline and "+ ET".

The complete approach is denoted as "RBPB" in Table 1 and Table 2. Remarkably, our approach performances comparable in trigger classification with the state-of art methods: Cross-Document, Cross-Event, Cross-Entity, Joint model, DMCNN and significantly higher than them in argument identification as well as classification although we did not use the cross-document, cross-event information or any global feature. Therefore, the relationship between argument candidates can indeed contribute to argument identification performance. The event type classifier also contributes a lot in trigger identification & classification. We do the Wilcoxon Signed Rank Test on trigger classification, argument identification and argument classification, all the three have $p < 0.01$.

A more detailed study of the pattern feature's effect is shown in Table 3. We can see that RBPB with both plain feature and pattern feature can gain much better performance than with two kinds of features alone.

Method (RBPB)	Trigger	Arg id	Arg id+cl
+ Plain feature	66.0	60.5	50.4
+ Pattern feature	65.8	60.1	49.2
+ Both	**68.9**	**61.2**	**53.8**

Table 3: The effect (F_1 value) of pattern feature

However, our approach is just a pipeline approach which suffers from error propagation and the argument performance may not affect the trigger too much. We can see from Table 1 that although we use gold argument candidates, the trigger performance is still lower than DMCNN. Another limitation is that our regularization method does not improve the argument classification too much since it only uses constraints to affect roles. Future work may be done to solve these two limitations.

6.4 Analysis of Argument Relationships

The accuracy of the argument relationship maxent classifier is 82.4%. Fig 4 shows an example of the argument relationship matrix, which works

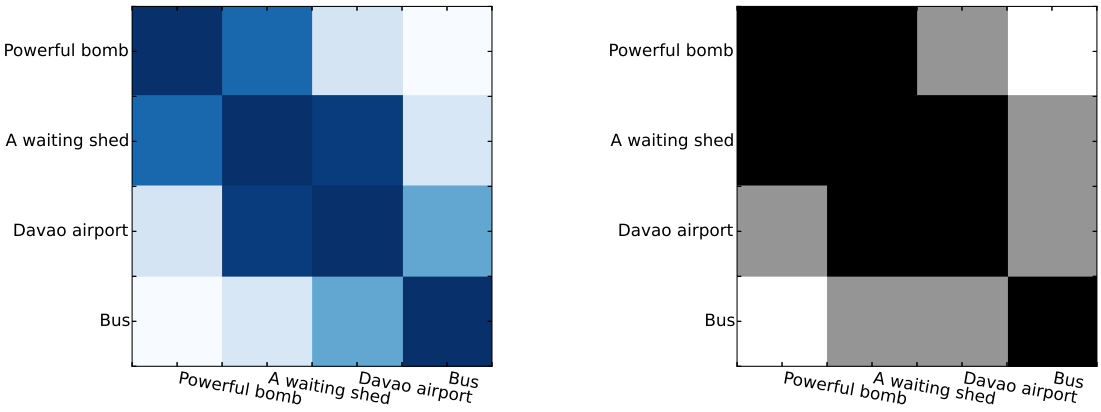


Figure 4: The Argument Relationship Matrix. Left is the origin matrix. Right is the strengthened matrix

for the sentence in Fig 1. In the left part of Fig 4, we can see the argument relationship we capture directly (the darker blue means stronger connection, lighter blue means weaker connection). After strengthening, on the right, the entities with strong connections are classified as positive correlations (the black squares), weak connections are classified as negative correlations (the white squares). Others (the grey squares) are unclear correlations. We can see that positive correlation is between "Powerful bomb" and "A waiting shed" as well as "A waiting shed" and "Davao airport". Therefore, these entities tend to be extracted at the same time. However, "Powerful bomb" and "Bus" has a negative correlation, so they tend not to be extracted at the same time. In practice, the argument probability of "Powerful bomb" and "A waiting shed" are much higher than the other two. Therefore, "Powerful bomb", "A waiting shed" and "Davao airport" are the final extraction results.

7 Conclusion

In this paper, we propose two improvements based on the event extraction baseline JET. We find that JET depends too much on event patterns for event type priori and JET considers each candidate argument separately. However, patterns cannot cover all events and the relationship between candidate arguments may help when identifying arguments. For a trigger, if no pattern can be matched, the event type cannot be assigned and the arguments cannot be correctly identified and classified. Therefore, we develop an event type classifier to assign the event type, using both pattern matching information and other features, which gives our system the capability to deal with failed match cases when using patterns alone.

On the other hand, we train a maximum entropy classifier to predict the relationship between candidate arguments. Then we propose a regularization method to make full use of the argument relationship. Our experiment results show that the regularization method is a significant improvement in argument identification over previous works.

In summary, by using the event type classifier and the regularization method, we have achieved a good performance in which the trigger classification is comparable to state-of-the-art methods, and the argument identification & classification performance is significantly better than state-of-the-art methods. However, we only use sentence-level features and our method is a pipelined approach. Also, the argument classification seems not to be affected too much by the regularization. Future work may be done to integrate our method into a joint approach, use some global feature, which may improve our performance. The code is available at `https://github.com/shalei120/RBPB/tree/master/RBET_release`

Acknowledgements

We would like to thank our three anonymous reviewers for their helpful advice on various aspects of this work. This research was supported by the National Key Basic Research Program of China (No.2014CB340504) and the National Natural Science Foundation of China (No.61375074,61273318). The contact author for this paper is Zhifang Sui.

References

Mary Elaine Califf and Raymond J Mooney. 2003. Bottom-up relational learning of pattern matching rules for information extraction. *The Journal of Machine Learning Research*, 4:177–210.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 976–986. Association for Computational Linguistics.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–176, Beijing, China, July. Association for Computational Linguistics.

Hai Leong Chieu and Hwee Tou Ng. 2002. A maximum entropy approach to information extraction from semi-structured and free text. *AAAI/IAAI*, 2002:786–791.

Fabio Ciravegna et al. 2001. Adaptive information extraction from text by rule induction and generalisation. In *International Joint Conference on Artificial Intelligence*, volume 17, pages 1251–1256. LAWRENCE ERLBAUM ASSOCIATES LTD.

Aidan Finn and Nicholas Kushmerick. 2004. *Multi-level boundary classification for information extraction*. Springer.

Dayne Freitag. 1998a. Multistrategy learning for information extraction. In *ICML*, pages 161–169.

Dayne Freitag. 1998b. Toward general-purpose learning for information extraction. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 404–408. Association for Computational Linguistics.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. Nyus english ace 2005 system description. *ACE*, 5.

Zhenmei Gu and Nick Cercone. 2006. Segment-based hidden markov models for information extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 481–488. Association for Computational Linguistics.

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1127–1136. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2011. Peeling back the layers: detecting event role fillers in secondary contexts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 1137–1147. Association for Computational Linguistics.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped training of event extraction classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295. Association for Computational Linguistics.

Scott B Huffman. 1996. Learning information extraction patterns from examples. In *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, pages 246–260. Springer.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of the 46st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 254–262.

Jun-Tae Kim and Dan I Moldovan. 1993. Acquisition of semantic patterns for information extraction from corpora. In *Artificial Intelligence for Applications, 1993. Proceedings., Ninth Conference on*, pages 171–176. IEEE.

Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. 2005. Using uneven margins svm and perceptron for information extraction. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 72–79. Association for Computational Linguistics.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 73–82, Sofia, Bulgaria, August. Association for Computational Linguistics.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797. Association for Computational Linguistics.

Ting Liu and Tomek Strzalkowski. 2012. Bootstrapping events and relations from text. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 296–305. Association for Computational Linguistics.

Wei Lu and Dan Roth. 2012. Automatic event extraction with structured preference modeling. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 835–844. Association for Computational Linguistics.

Mstislav Maslennikov and Tat-Seng Chua. 2007. A multi-resolution framework for information extraction from free text. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 592. Citeseer.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *EMNLP-CoNLL*, volume 7, pages 717–727. Citeseer.

Siddharth Patwardhan and Ellen Riloff. 2009. A unified model of phrasal and sentential evidence for information extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1*, pages 151–160. Association for Computational Linguistics.

Ellen Riloff et al. 1993. Automatically constructing a dictionary for information extraction tasks. In *AAAI*, pages 811–816.

Ellen Riloff, Rosie Jones, et al. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *AAAI/IAAI*, pages 474–479.

Ellen Riloff. 1996. Automatically generating extraction patterns from untagged text. In *Proceedings of the national conference on artificial intelligence*, pages 1044–1049.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 731–738. Association for Computational Linguistics.

Yusuke Shinyama and Satoshi Sekine. 2006. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 304–311. Association for Computational Linguistics.

Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. 1995. Crystal: Inducing a conceptual dictionary. *arXiv preprint cmp-lg/9505020*.

Mark Stevenson and Mark A Greenwood. 2005. A semantic approach to ie pattern induction. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 379–386. Association for Computational Linguistics.

Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. 2003. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 224–231. Association for Computational Linguistics.

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics-Volume 2*, pages 940–946. Association for Computational Linguistics.

Kun Yu, Gang Guan, and Ming Zhou. 2005. Resume information extraction with cascaded hybrid model. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 499–506. Association for Computational Linguistics.

Neural Network-Based Model for Japanese Predicate Argument Structure Analysis

Tomohide Shibata and **Daisuke Kawahara** and **Sadao Kurohashi**
Graduate School of Informatics, Kyoto University
Yoshida-honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
{shibata, dk, kuro}@i.kyoto-u.ac.jp

Abstract

This paper presents a novel model for Japanese predicate argument structure (PAS) analysis based on a neural network framework. Japanese PAS analysis is challenging due to the tangled characteristics of the Japanese language, such as case disappearance and argument omission. To unravel this problem, we learn selectional preferences from a large raw corpus, and incorporate them into a SOTA PAS analysis model, which considers the consistency of all PASs in a given sentence. We demonstrate that the proposed PAS analysis model significantly outperforms the base SOTA system.

1 Introduction

Research on predicate argument structure (PAS) analysis has been conducted actively these days. The improvement of PAS analysis would benefit many natural language processing (NLP) applications, such as information extraction, summarization, and machine translation.

The target of this work is Japanese PAS analysis. The Japanese language has the following characteristics:

- head final,
- free word order (among arguments), and
- postpositions function as (surface) case markers.

Japanese major surface cases are が (*ga*), を (*wo*), and に (*ni*), which correspond to Japanese postpositions (case markers). We call them nominative case, accusative case, and dative case, respectively. In this paper, we limit our target cases to these three cases. Note that though they are surface cases, they roughly correspond to Arg1, Arg2, and Arg3 of English semantic role labeling based on PropBank.

Japanese PAS analysis has been considered as one of the most difficult basic NLP tasks, due to the following two phenomena.

Case disappearance When a topic marker は (*wa*) is used or a noun is modified by a relative clause, their case markings disappear as in the following examples.[1]

(1) a. ジョンは パンを 食べた。 → ジョンが
John-TOP bread-ACC ate John-NOM
(John ate bread.)

b. パンは ジョンが 食べた。 → パンを
bread-TOP John-NOM ate bread-ACC
(John ate bread.)

(2) a. パンを 食べた ジョン を ... → ジョンが (食べた)
bread-ACC ate John-ACC John-NOM (ate)
(John, who ate bread, ...)

b. ジョンが 食べた パン が ...→ パンを (食べた)
John-NOM ate bread-NOM (ate) bread-ACC
(Bread, which John ate, ...)

In the example sentences (1a) and (1b), since a topic marker は is used, the NOM and ACC case markers disappear. In the example sentences (2a) and (2b), since a noun is modified by a relative clause, the NOM case of "ジョン" (John) for "食べた" (eat) and ACC case of "パン" (bread) for "食べた" disappear.

Argument omission Arguments are very often omitted in Japanese sentences. This phenomenon is totally different from English sentences, where the word order is fixed and pronouns are used con-

[1] In this paper, we use the following abbreviations: NOM (nominative), ACC (accusative), DAT (dative) and TOP (topic marker).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1235–1244,
Berlin, Germany, August 7-12, 2016.

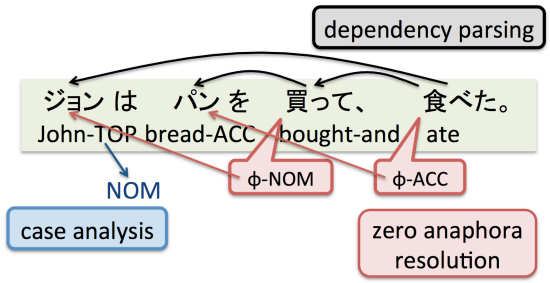


Figure 1: An example of PAS analysis. Input sentence: “ジョンはパンを買って、食べた。” (John bought bread, and ate it.)

sistently. For example, let us compare the following parallel Japanese and English sentences:

(3) a. ジョンは パンを 買って、 食べた。
John-TOP bread-ACC bought ate
b. John bought bread, and ate it.

The dependency parse of (3a) is shown in Figure 1. In general, the first phrase with a topic marker は is treated as modifying the final predicate according to the guidelines of Japanese dependency annotation. As a result, “買って” (bought) has no NOM argument (omitted), and “食べた” (ate) has no ACC argument. Note that “食べた” has an argument “ジョン” (John), but its case does not appear.

In the case of the parallel sentences (4) below, again we can witness the difficulty of Japanese PAS analysis.

(4) a. パンを 買った ジョンは 急いで 食べた。
bread-ACC bought John-TOP hurry ate
b. John who bought bread ate it in a hurry.

Although all the case arguments of the predicates “bought” and “ate” are explicit in (4b), the case of “ジョン” (John) for “買った” (bought) and that for “食べた” (ate) are hidden, and the ACC argument of “食べた” (ate) is omitted in (4a).

Many researchers have been tackling Japanese PAS analysis (Taira et al., 2008; Imamura et al., 2009; Hayashibe et al., 2011; Sasano and Kurohashi, 2011; Hangyo et al., 2013; Ouchi et al., 2015). However, because of the two aforementioned characteristics in Japanese sentences, the accuracy of Japanese PAS analysis for omitted (zero) arguments remains around 40%.

This paper proposes a novel Japanese PAS analysis model based on a neural network (NN) framework, which has been proved to be effective for several NLP tasks recently. To unravel the tangled situation in Japanese, we learn selectional preferences from a large raw corpus, and incorporate them into a SOTA PAS analysis model proposed by Ouchi et al. (2015), which considers the consistency of all PASs in a given sentence. This model is achieved by an NN-based two-stage model that acquires selectional preferences in an unsupervised manner in the first stage and predicts PASs in a supervised manner in the second stage as follows.

1. The most important clue for PAS analysis is selectional preferences, that is, argument prediction from a predicate phrase. For example, how likely the phrase “パンを買った” (bought bread) takes “ジョン” (John) as its NOM argument.

   Such information cannot be learned from a medium-sized PAS annotated corpus with size of the order of ten-thousand sentences; it is necessary to use a huge raw corpus by an unsupervised method. Ouchi et al. (2015) did not utilize such knowledge extracted from a raw corpus. Some work has utilized PMI between a predicate and an argument, or case frames obtained from a raw corpus. However, this is discrete word-based knowledge, not generalized semantic knowledge.

   As the first stage of the method, we learn a prediction score from a predicate phrase to an argument by an NN-based method. The resultant vector representations of predicates and arguments are used as initial vectors for the second stage of the method.

2. In the second stage, we calculate a score that a predicate in a given sentence takes an element in the sentence as an argument using NN framework. We use the prediction score in the first stage as one feature for the second stage NN. The system by Ouchi et al. (2015) used a manually designed feature template to take the interactions of the atomic features into consideration. In the case of an NN framework, no feature template is required, and a hidden layer in an NN can capture the interactions of the atomic features automatically and flexibly.

We demonstrate that the proposed PAS analysis model outperforms the SOTA system by Ouchi et al. (2015).

2 Related Work

Several methods for Japanese PAS analysis have been proposed. The methods can be divided into three types: (i) identifying one case argument independently per predicate (Taira et al., 2008; Imamura et al., 2009; Hayashibe et al., 2011), (ii) identifying all the three case arguments (NOM, ACC, and DAT) simultaneously per predicate (Sasano and Kurohashi, 2011; Hangyo et al., 2013), and (iii) identifying all case arguments of all predicates in a sentence (Ouchi et al., 2015). The third method can capture interactions between predicates and their arguments, and thus performs the best among the three types. This method is adopted as our base model (see Section 3 for details).

Most methods for PAS analysis handle both intra-sentential and inter-sentential zero anaphora. For identifying inter-sentential zero anaphora, an antecedent has to be searched in a broad search space, and the salience of discourse entities has to be captured. Therefore, the task of identifying inter-sentential zero anaphora is more difficult than that of intra-sentential zero anaphora. Thus, Ouchi et al. (2015) and Iida et al. (2015) focused on only intra-sentential zero anaphora. Following this trend, this paper focuses on intra-sentential zero anaphora.

Recently, NN-based approaches have achieved improvement for several NLP tasks. For example, in transition-based parsing, Chen and Manning (2014) proposed an NN-based approach, where the words, POS tags, and dependency labels are first represented by embeddings individually. Then, an NN-based classifier is built to make parsing decisions, where an input layer is a concatenation of embeddings of words, POS tags, and dependency labels. This model has been extended by several studies (Weiss et al., 2015; Dyer et al., 2015; Ballesteros et al., 2015). In semantic role labeling, Zhou and Xu (2015) propose an end-to-end approach using recurrent NN, where an original text is the input, and semantic role labeling is performed without any intermediate syntactic knowledge. Following these approaches, this paper proposes an NN-based PAS method.

3 Base Model

The model proposed by Ouchi et al. (2015) is adopted as our base model (Figure 2). We briefly introduce this base model before describing our proposed model.

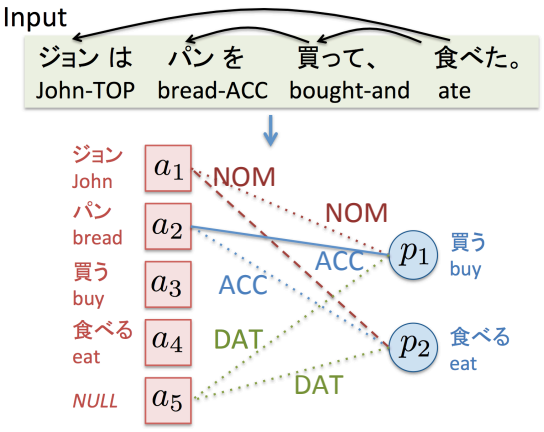


Figure 2: Our base model (Ouchi et al., 2015).

3.1 Predicate-Argument Graph

In this model, for an input sentence, a bipartite graph is constructed, consisting of the set of predicate and argument nodes. This is called Predicate-Argument Graph (PA Graph). A PA graph represents a possible interpretation of the input sentence, including case analysis result and zero anaphora resolution result.

A PA graph is a bipartite graph $\langle A, P, E \rangle$, where A is the node set consisting of candidate arguments, P is the node set consisting of predicates, and E is the set of edges. A PA graph is defined as follows:

$$\begin{aligned} A &= \{a_1, \ldots, a_n, a_{n+1} = \texttt{NULL}\} \\ P &= \{p_1, \ldots, p_m\} \\ E &= \{\langle a, p, c \rangle | deg(p, c) = 1, \\ &\quad \forall a \in A, \forall p \in P, \forall c \in C\} \end{aligned}$$

where n and m represent the number of predicates and arguments, and C denotes the case role set (NOM, ACC, and DAT). An edge $e \in E$ is represented by a tuple $\langle a, p, c \rangle$, indicating the edge with a case role c connecting a candidate argument node a and a predicate node p. $deg(p, c)$ is the number of the edges with a case role c outgoing from a predicate node p. An admissible PA graph satisfies the constraint $deg(p, c) = 1$, which means each predicate node p has only one edge with a case role c. A dummy node a_{n+1} is added, which is defined for the cases where a predicate requires no case argument (e.g. when the predicate node "ある" (exist) connects a NULL node with a case ACC, this means this predicate takes

no ACC argument) or the required argument does not appear in the sentence.

In the bipartite graph shown in Figure 2, the three kinds of edge lines have the meaning as follows:

solid line: the argument node and the predicate node has a dependency relation, and the argument node is followed by a case marking postposition. In this case, these nodes have a relation through its corresponding case marking postposition. Therefore, this edge is fixed.

dashed line: the argument node and the predicate node has a dependency relation, and the argument node is not followed by a case marking postposition. These nodes are likely to have a relation[2], but the case role is unknown. Identifying this case role corresponds to case analysis.

dotted line: the argument node and the predicate node do not have a dependency relation. Identifying this edge and its case role corresponds to zero anaphora resolution.

For an input sentence x, a scoring function $Score(x, y)$ is defined for a candidate graph y, and the PA graph that has the maximum score is searched.

$$\tilde{y} = \underset{y \in G(x)}{\operatorname{argmax}} Score(x, y) \tag{1}$$

where $G(x)$ is a set of admissible PA graphs for the input sentence x. $Score(x, y)$ is defined as follows[3]:

$$\sum_{e \in E(y)} score_l(x, e) + \sum_{e_i, e_j \in E_{pair}(y)} score_g(x, e_i, e_j). \tag{2}$$

$$\begin{aligned} score_l(x, e) &= \boldsymbol{\theta}_l \cdot \boldsymbol{\phi}_l(x, e) \\ score_g(x, e_i, e_j) &= \boldsymbol{\theta}_g \cdot \boldsymbol{\phi}_g(x, e_i, e_j) \end{aligned} \tag{3}$$

where $E(y)$ is the edge set on the candidate graph y, $E_{pair}(y)$ is a set of edge pairs in the edge set $E(y)$, $score_l(x, e)$ and $score_g(x, e_i, e_j)$ represent a local score for the edge e and a global score for the edge pair e_i and e_j, $\boldsymbol{\phi}_l(x, e)$ and $\boldsymbol{\phi}_g(x, e_i, e_j)$ represent local features and global features. While $\boldsymbol{\phi}_l(x, e)$ is defined for each edge e, $\boldsymbol{\phi}_g(x, e_i, e_j)$ is defined for each edge pair e_i, e_j $(i \neq j)$. $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_g$ represent model parameters for local and global features. By using global scores, the interaction between multiple case assignments of multiple predicates can be considered.

[2]For example, in the sentence "今日は 暑い" (today-TOP hot), the predicate "暑い" does not take "今日", which represents time, as an argument. Therefore, these nodes do not always have a relation.

[3]Ouchi et al. (2015) introduce two models: Per-Case Joint Model and All-Cases Joint Model. Since All-Cases Joint Model performed better than Per-Case Joint Model, All-Cases Joint Model is adopted as our base model.

3.2 Inference and Training

Since global features make the inference of finding the maximum scoring PA graph more difficult, the randomized hill-climbing algorithm proposed in (Zhang et al., 2014) is adopted.

Figure 3 describes the pseudo code for hill-climbing algorithm. First, an initial PA graph $y^{(0)}$ is sampled from the set of admissible PA graph $G(x)$. Then, the union Y is constructed from the set of neighboring graphs $NeighborG(y^{(t)})$, which is a set of admissible graphs obtained by changing one edge in $y^{(t)}$, and the current graph $y^{(t)}$. The current graph $y^{(t)}$ is updated to a higher scoring graph $y^{(t+1)}$. This process continues until no more improvement is possible, and finally an optimal graph $\tilde{y}$ can be obtained.

Input: sentence x, parameter $\boldsymbol{\theta}$
Output: a locally optimal PA graph $\tilde{y}$
Sample a PA graph $y^{(0)}$ from $G(x)$
$t \leftarrow 0$
repeat
  $Y \leftarrow NeighborG(y^{(t)}) \bigcup y^{(t)}$
  $y^{(t+1)} \leftarrow \underset{y \in Y}{\operatorname{argmax}} Score(x, y; \boldsymbol{\theta})$
  $t \leftarrow t + 1$
until $y^{(t)} = y^{(t+1)}$
return $\tilde{y} \leftarrow y^{(t)}$

Figure 3: Hill climbing algorithm for obtaining optimal PA graph.

Given N training examples $D = \{(x, \hat{y})\}_k^N$, the model parameter $\boldsymbol{\theta}$ are estimated. $\boldsymbol{\theta}$ is the set of $\boldsymbol{\theta}_l$ and $\boldsymbol{\theta}_g$, and is estimated by averaged perceptron (Collins, 2002) with a max-margin framework (Taskar et al., 2005).

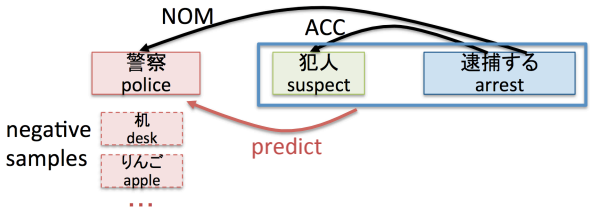


Figure 4: Argument prediction model. In the PAS "警察" (police) NOM "犯人" (suspect) ACC "逮捕" (arrest), "警察" with the NOM case is predicted given the predicate "逮捕" (arrest) and its ACC "犯人" (suspect).

4 Proposed Model

4.1 Argument Prediction Model

No external knowledge is utilized in the base model. One of the most important types of knowledge in PAS analysis is selectional preferences. Sasano and Kurohashi (2011) and Hangyo et al. (2013) extract knowledge of the selectional preferences in the form of case frames from a raw corpus, and the selectional preference score is used as a feature. In this work, *argument prediction model* is trained using a neural network from a raw corpus, in a similar way to Titov and Khoddam (2015) and Hashimoto et al. (2014).

PASs are first extracted from an automatically-parsed raw corpus, and in each PAS, the argument a_i is generated with the following probability $p(a_i|PAS_{-a_i})$:

$$p(a_i|PAS_{-a_i}) = \frac{\exp(\boldsymbol{v}_{a_i}^{\mathrm{T}} W_{a_i}^{\mathrm{T}}(W_{pred}\boldsymbol{v}_{pred} + \sum_{j\neq i} W_{a_j}\boldsymbol{v}_{a_j}))}{Z} \quad (4)$$

where PAS_{-a_i} represents a PAS excluding the target argument a_i, $\boldsymbol{v}_{pred}$, $\boldsymbol{v}_{a_i}$ and $\boldsymbol{v}_{a_j}$ represent embeddings of the predicate, argument a_i and argument a_j, and W_{pred}, W_{a_i}, and W_{a_j} represent transformation matrices for a predicate and an argument a_i and a_j. Z is the partition function.

Figure 4 illustrates the argument prediction model. The PAS "警察" (police) NOM "犯人" (suspect) ACC "逮捕" (arrest)" is extracted from a raw corpus, and the probability of NOM argument "警察" given the predicate "逮捕" and its ACC argument "犯人" is calculated.

All the parameters including predicate/argument embeddings and transformation matrices are trained, so that the likelihood given by Equation (4) is high. Since the denominator of Equation (4) is impractical to be calculated since the number of vocabulary is enormous, negative sampling (Mikolov et al., 2013) is adopted. In the example shown in Figure 4, as for a NOM argument, negative examples, such as "机" (desk) and "りんご" (apple), are drawn from the noise distribution, which is a unigram distribution raised to the 3/4th power.

In each PAS, all the arguments are predicted in turn. All the parameters are updated using stochastic gradient descent.

This model is first trained using the automatic parsing result on a raw corpus, and in performing PAS analysis described in Section 4.2, the score derived from this model is used as a feature.

4.2 Neural Network-Based Score Calculation

In the base model, the score for an edge (local score) or an edge pair (global score) is calculated using the dot product of a sparse high-dimensional feature vector with a model parameter, as shown in Equation (3). In our proposed model, these scores are calculated in a standard neural network with one hidden layer, as shown in Figure 5.

We first describe the calculation of the local score $score_l(x, e)$. A predicate p and an argument a are represented by embeddings (a d dimensional vector) $\boldsymbol{v}_p$ and $\boldsymbol{v}_a \in \mathbb{R}^d$, and $\boldsymbol{v}_{f_l} \in \mathbb{R}^{d_f}$ (d_f represents a dimensional of $\boldsymbol{v}_{f_l}$) represents a feature vector obtained by concatenating the case role between a and p, the argument prediction score obtained from the model described in Section 4.1, and the other atomic features. An input layer is a concatenation of these vectors, and then, a hidden layer $\boldsymbol{h}_l \in \mathbb{R}^{d_h}$ (d_h represents a dimension of the hidden layer) is calculated as follows:

$$\boldsymbol{h}_l = f(W_l^1[\boldsymbol{v}_p; \boldsymbol{v}_a; \boldsymbol{v}_{f_l}]) \quad (5)$$

where f is an element-wise activation function ($tanh$ is used in our experiments), and $W_l^1 \in \mathbb{R}^{d_h(2d+d_h)}$ is a weight matrix (for the local score) from the input layer to the hidden layer. The scalar score in an output layer is then calculated as follows:

$$score_l(x, e) = f(W_l^2 \boldsymbol{h}_l) \quad (6)$$

where $W_l^2 \in \mathbb{R}^{(2d+d_h)\cdot 1}$ is a weight matrix (for the local score) from the hidden layer to the output layer. By calculating the score in this way, all the

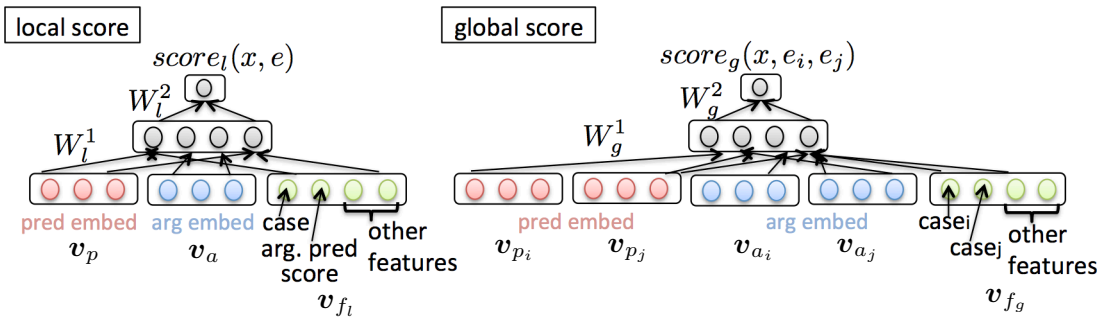


Figure 5: A score calculation in our proposed neural-network based model. The left part and right part represent a local and global score calculation.

combinations of features in the input layer can be considered.

Next we describe the calculation of the global score $score_g(x, e_i, e_j)$. In the base model, the two types of global features are utilized: one is for the two predicates having different arguments, and the other is for the two predicates sharing the same argument. The input layer is a concatenation of involving vectors of predicates/arguments and the other features $\boldsymbol{v}_{f_g}$. For example, when calculating the global score for the two predicates having different arguments, the input layer is a concatenation of the vectors of two predicates and two arguments and $\boldsymbol{v}_{f_g}$.

A hidden layer $\boldsymbol{h}_g$ is calculated as follows:

$$\boldsymbol{h}_g = f(W_g^1[\boldsymbol{v}_{p_i}; \boldsymbol{v}_{p_j}; \boldsymbol{v}_{a_i}; \boldsymbol{v}_{a_j}; \boldsymbol{v}_{f_g}]) \quad (7)$$

where W_g^1 is a weight matrix (for the global score) from the input layer to the hidden layer, $\boldsymbol{v}_{p_i}$ and $\boldsymbol{v}_{a_i}$ are the embeddings of the predicate/argument connected by e_i, and $\boldsymbol{v}_{p_j}$ and $\boldsymbol{v}_{a_j}$ are defined in the same way.

The scalar score in an output layer is then calculated as follows:

$$score_g(x, e_i, e_j) = f(W_g^2 \boldsymbol{h}_g) \quad (8)$$

where W_g^2 is a weight matrix (for the global score) from the hidden layer to the output layer.

4.3 Inference and Training

While inference is the same as the base model, training is slightly different.

In our proposed model, the model parameter $\boldsymbol{\theta}$ consists of the embeddings of predicates/arguments and weight matrices for the local/global score in the neural networks. Our objective is to minimize the following loss function:

case	# of dep arguments	# of zero arguments	total
NOM	1,402	1,431	2,833
ACC	278	113	391
DAT	92	287	379
ALL	1,772	1,831	3,603

Table 1: Test set statistics of the number of arguments.

$$J(\boldsymbol{\theta}) = \sum_k^N l_k(\boldsymbol{\theta}), \quad (9)$$

where

$$l_k(\boldsymbol{\theta}) = \max_{y_k \in G(x)} (Score(x_k, y_k; \boldsymbol{\theta}) - Score(x_k, \hat{y}_k; \boldsymbol{\theta}) + ||y_k - \hat{y}_k||_1), \quad (10)$$

and $||y_k - \hat{y}_k||_1$ denotes the Hamming distance between the gold PA graph $\hat{y}_k$ and a candidate PA graph y_k.

Stochastic gradient descent is used for parameter inference. Derivatives with respect to parameters are taken using backpropagation. Adam (Kingma and Ba, 2014) is adopted as the optimizer.

For initialization of the embeddings of a predicate/argument, the embeddings of the predicate/argument trained by the method described in Section 4.1 are utilized. The weight matrices are randomly initialized.

5 Experiment

5.1 Experimental Setting

The KWDLC (Kyoto University Web Document Leads Corpus) evaluation set (Hangyo et al., 2012) was used for our experiments, because it contains

a wide variety of Web documents, such as news articles and blogs. This evaluation set consists of the first three sentences of 5,000 Web documents. Morphology, named entities, dependencies, PASs, and coreferences were manually annotated.

This evaluation set was divided into 3,694 documents (11,558 sents.) for training, 512 documents (1,585 sents.) for development, and 700 documents (2,195 sents.) for testing. Table 1 shows the statistics of the number of arguments in the test set. While "dep argument" means that the argument and a predicate have a dependency relation, but a specified case marking postposition is hidden (corresponds to "dashed line" in Section 3.1), "zero argument" means that the argument and a predicate do not have a dependency relation (corresponds to "dotted line" in Section 3.1).

Since we want to focus on the accuracy of case analysis and zero anaphora resolution, gold morphological analysis, dependency analysis, and named entities were used.

The sentences having a predicate that takes multiple arguments in the same case role were excluded from training and test examples, since the base model cannot handle this phenomena (it assumes that each predicate has only one argument with one case role). For example, the following sentence,

(5) そんな 面白ネタ 満載な
such funny-material full
日々を 絵と共に
daily life-ACC picture-with
お届けします。,
report
(I report my daily life full of such funny materials along with pictures.)

where the predicate "お届けします" (report) takes both "日々" (daily life) and "絵" (picture) as ACC case arguments, was excluded from training and testing. About 200 sentences (corresponding to about 1.5% of the whole evaluation set) were excluded.

In this evaluation set, zero exophora, which is a phenomenon that a referent does not appear in a document, is annotated. Among five types of zero exophora, the two major types, "author" and "reader," are adopted, and the others are discarded. To consider "author" and "reader" as a referent, the two special nodes, AUTHOR and READER, are added as well as a NULL node in a PA graph of the base model. When the argument predication score is calculated for "author" or "reader," because its lemma does not appear in a document, for each noun in the following noun list of "author"/"reader" (Hangyo et al., 2013), the argument prediction score is calculated, and the maximum score is used as a feature.

- author: "私" (I), "我々" (we), "僕" (I), "弊社" (our company), · · ·
- reader: "あなた" (you), "客" (customer), "君" (you), "皆様"(you all), · · ·

In the argument prediction model training described in Section 4.1, a Japanese Web corpus consisting of 10M sentences was used. We preformed syntactic parsing with a publicly available Japanese parser, KNP[4]. The number of negative samples was 5, and the number of epochs was 10.

In the model training described in Section 4.3, the dimensions of both embeddings for predicates/arguments and hidden layer were set to 100. The number of epochs was set to 20, following the base model.

5.2 Result

We compared the following three methods:

- Baseline (Ouchi et al., 2015)
- Proposed model w/o arg. prediction score: in the PAS analysis model, the feature derived from the argument prediction model was not utilized. The embeddings of a predicate/argument were randomly initialized. This method corresponds to adopting the NN-based score calculation in the base model.
- Proposed model w/ arg. prediction score: the feature derived from the argument prediction model was utilized, and the embeddings of a predicate/argument were initialized with those obtained in the argument prediction model learning.

The performances of case analysis and zero anaphora resolution were evaluated by micro-averaged precision, recall, and F-measure. The precision, recall, and F-measure were averaged

[4]http://nlp.ist.i.kyoto-u.ac.jp/index.php?KNP

case	method	case analysis			zero anaphora		
		P	R	F	P	R	F
NOM	Baseline	0.880	0.868	0.874	**0.693**	0.377	0.488
	Proposed model w/o arg. prediction score	0.927	0.917	0.922	0.559	0.532	0.545
	Proposed model w arg. prediction score	**0.946**	**0.936**	**0.941**	0.568	**0.586**	**0.577**
ACC	Baseline	0.433	0.374	0.402	0.000	0.000	0.000
	Proposed model w/o arg. prediction score	0.805	0.553	0.656	0.151	0.060	0.085
	Proposed model w/ arg. prediction score	**0.890**	**0.658**	**0.756**	**0.297**	**0.124**	**0.173**
DAT	Baseline	0.224	**0.359**	0.276	0.531	0.059	0.107
	Proposed model w/o arg. prediction score	0.512	0.104	0.173	0.535	0.242	0.332
	Proposed model w/ arg. prediction score	**0.834**	0.185	**0.300**	**0.622**	**0.273**	**0.378**
ALL	Baseline	0.765	0.764	0.765	**0.686**	0.304	0.421
	Proposed model w/o arg. prediction score	0.908	0.818	0.860	0.544	0.458	0.497
	Proposed model w/ arg. prediction score	**0.937**	**0.853**	**0.893**	0.563	**0.509**	**0.534**

Table 2: Experimental results on the KWDLC corpus.

over 5 runs. Table 2 shows our experimental results. Our proposed method outperformed the baseline method by about 11 absolute points in F-measure. The comparison of "Proposed model w/o arg. prediction score" with the baseline showed that the neural network-based approach was effective, and the comparison of "Proposed model w/ arg. prediction score" with "Proposed model w/o arg. prediction score" showed that our arg. prediction model was also effective.

The following is improved by adding an argument prediction score.

(6) 久しぶりに　　パートですけど、
after a long time part-time job
働き始めて　新しい 一歩を
begin to work new　　step-ACC
踏み出しました。
step forward
(It's my first part-time job in a long time. I begin to work, and make a new step.)

While in the base model, the NOM arguments of the predicate "働き始める" (begin to work) and "踏み出す" (step forward) were wrongly classified as NULL, by adding an argument prediction score, they were correctly identified as "author."

The phenomenon "case disappearance" occurs in other languages such as Korean, and the phenomenon "argument omission" occurs in other languages such as Korean, Hindi, Chinese, and Spanish. We believe that our neural network approach to the argument prediction and the calculation of the local and global scores is also effective for such languages.

5.3 Error Analysis

Errors in our proposed model are listed below:

- Recall for ACC and DAT in both case analysis and zero anaphora resolution is low.

  One reason is that since the number of the ACC and DAT arguments is smaller than that of the NOM argument, the system tends to assign the ACC and DAT arguments with NULL. Another reason is that since this paper focuses on intra-sentential zero anaphora, the NULL arguments include arguments that appear in previous sentences as well as the case where a predicate takes no argument, which makes the training for NULL arguments difficult. We are planing to tackle with inter-sentential zero anaphora resolution.

- The distinction of "author" from NULL fails.

  (7) 肉を　　焼く だけが
  meat-ACC roast-only-NOM
  BBQ じゃない！
  BBQ-(COPULA)
  (Roasting meat isn't all in BBQ!)

  Although the NOM argument of the predicate "焼く" (roast) is "author," our proposed model wrongly classified it as NULL. Hangyo et al. (2013) identify mentions referring to an author or reader in a document, and utilize this result in the zero anaphora resolu-

tion. We plan to incorporate the author/reader identification into our model.

6 Conclusion

In this paper we presented a novel model for Japanese PAS analysis based on neural network framework. We learned selectional preferences from a large raw corpus, and incorporated them into a PAS analysis model, which considers the consistency of all PASs in a given sentence. In our experiments, we demonstrated that the proposed PAS analysis model significantly outperformed the base SOTA model.

In the future, we plan to extend our model to incorporate coreference resolution and inter-sentential zero anaphora resolution.

Acknowledgments

This work is supported by CREST, Japan Science and Technology Agency.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359, Lisbon, Portugal, September. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2012. Building a diverse document leads corpus annotated with semantic relations. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 535–544, Bali,Indonesia, November. Faculty of Computer Science, Universitas Indonesia.

Masatsugu Hangyo, Daisuke Kawahara, and Sadao Kurohashi. 2013. Japanese zero reference resolution considering exophora and author/reader mentions. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 924–934, Seattle, Washington, USA, October. Association for Computational Linguistics.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1544–1555, Doha, Qatar, October. Association for Computational Linguistics.

Yuta Hayashibe, Mamoru Komachi, and Yuji Matsumoto. 2011. Japanese predicate argument structure analysis exploiting argument position and type. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 201–209, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Ryu Iida, Kentaro Torisawa, Chikara Hashimoto, Jong-Hoon Oh, and Julien Kloetzer. 2015. Intra-sentential zero anaphora resolution using subject sharing recognition. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2179–2189, Lisbon, Portugal, September. Association for Computational Linguistics.

Kenji Imamura, Kuniko Saito, and Tomoko Izumi. 2009. Discriminative approach to predicate-argument structure analysis with zero-anaphora resolution. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, pages 85–88, Suntec, Singapore, August. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Hiroki Ouchi, Hiroyuki Shindo, Kevin Duh, and Yuji Matsumoto. 2015. Joint case argument identification for Japanese predicate argument structure analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*,

pages 961–970, Beijing, China, July. Association for Computational Linguistics.

Ryohei Sasano and Sadao Kurohashi. 2011. A discriminative approach to Japanese zero anaphora resolution with large-scale lexicalized case frames. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 758–766, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Hirotoshi Taira, Sanae Fujita, and Masaaki Nagata. 2008. A Japanese predicate argument structure analysis using decision lists. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 523–532, Honolulu, Hawaii, October. Association for Computational Linguistics.

Ben Taskar, Vassil Chatalbashev, Daphne Koller, and Carlos Guestrin. 2005. Learning structured prediction models: A large margin approach. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 896–903, New York, NY, USA. ACM.

Ivan Titov and Ehsan Khoddam. 2015. Unsupervised induction of semantic roles within a reconstruction-error minimization framework. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1–10, Denver, Colorado, May–June. Association for Computational Linguistics.

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

Yuan Zhang, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2014. Greed is good if randomized: New inference for dependency parsing. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1013–1024, Doha, Qatar, October. Association for Computational Linguistics.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China, July. Association for Computational Linguistics.

Addressing Limited Data for Textual Entailment Across Domains

Chaitanya Shivade* **Preethi Raghavan**† and **Siddharth Patwardhan**†

*Department of Computer Science and Engineering,
The Ohio State University,
Columbus, OH 43210
shivade@cse.ohio-state.edu

†IBM T. J. Watson Research Center,
1101 Kitchawan Road,
Yorktown Heights, NY 10598
{praghav,siddharth}@us.ibm.com

Abstract

We seek to address the lack of labeled data (and high cost of annotation) for textual entailment in some domains. To that end, we first create (for experimental purposes) an entailment dataset for the clinical domain, and a highly competitive supervised entailment system, ENT, that is effective (out of the box) on two domains. We then explore self-training and active learning strategies to address the lack of labeled data. With self-training, we successfully exploit unlabeled data to improve over ENT by 15% F-score on the newswire domain, and 13% F-score on clinical data. On the other hand, our active learning experiments demonstrate that we can match (and even beat) ENT using only 6.6% of the training data in the clinical domain, and only 5.8% of the training data in the newswire domain.

1 Introduction

Textual entailment is the task of automatically determining whether a natural language *hypothesis* can be inferred from a given piece of natural language *text*. The RTE challenges (Bentivogli et al., 2009; Bentivogli et al., 2011) have spurred considerable research in textual entailment over newswire data. This, along with the availability of large-scale datasets labeled with entailment information (Bowman et al., 2015), has resulted in a variety of approaches for textual *entailment recognition*.

A variation of this task, dubbed textual *entailment search*, has been the focus of RTE-5 and subsequent challenges, where the goal is to find all sentences in a corpus that entail a given hypothesis. The mindshare created by those challenges and the availability of the datasets has spurred many creative solutions to this problem. However, the evaluations have been restricted primarily to these datasets, which are in the newswire domain. Thus, much of the existing state-of-the-art research has focused on solutions that are effective in this domain.

It is easy to see though, that entailment search has potential applications in other domains too. For instance, in the clinical domain we imagine entailment search can be applied for clinical trial matching as one example. Inclusion criteria for a clinical trial (for e.g., *patient is a smoker*) become the hypotheses, and the patient's electronic health records are the text for entailment search. Clearly, an effective textual entailment search system could possibly one day fully automate clinical trial matching.

Developing an entailment system that works well in the clinical domain and, thus, automates this matching process, requires lots of labeled data, which is extremely scant in the clinical domain. Generating such a dataset is tedious and costly, primarily because it requires medical domain expertise. Moreover, there are always privacy concerns in releasing such a dataset to the community. Taking this into consideration, we investigate the problem of textual entailment in a low-resource setting.

We begin by creating a dataset in the clinical domain, and a supervised entailment system that

*This work was conducted during an internship at IBM

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1245–1255,
Berlin, Germany, August 7-12, 2016.

is competitive on multiple domains – newswire as well as clinical. We then present our work on self-training and active learning to address the lack of a large-scale labeled dataset. Our self-training system results in significant gains in performance on clinical (+13% F-score) and on newswire (+15% F-score) data. Further, we show that active learning with uncertainty sampling reduces the number of required annotations for the entailment search task by more than 90% in both domains.

2 Related work

Recognizing Textual Entailment (RTE) shared tasks (Dagan et al., 2013) conducted annually from 2006 up until 2011 have been the primary drivers of textual entailment research in recent years. Initially the task was defined as that of *entailment recognition*. RTE-5 (Bentivogli et al., 2009) then introduced the task of *entailment search* as a pilot. Subsequently, RTE-6 (Bentivogli et al., 2010) and RTE-7 (Bentivogli et al., 2011) featured entailment search as the primary task, but constrained the search space to only those candidate sentences that were first retrieved by Lucene, an open source search engine[1]. Based on the 80% recall from Lucene in RTE-5, the organizers of RTE-6 and RTE-7 deemed this filter to be an appropriate compromise between the size of the search space and the cost and complexity of the human annotation task.

Annotating data for these tasks has remained a challenge since they were defined in the RTE challenges. Successful approaches for entailment (Mirkin et al., 2009; Jia et al., 2010; Tsuchida and Ishikawa, 2011) have relied on annotated data to either train classifiers, or to develop rules for detecting entailing sentences. Operating under the assumption that more labeled data would improve system performance, some researchers have sought to augment their training data with automatically or semi-automatically obtained labeled pairs (Burger and Ferro, 2005; Hickl et al., 2006; Hickl and Bensley, 2007; Zanzotto and Pennacchiotti, 2010; Celikyilmaz et al., 2009).

Burger and Ferro (2005) automatically create an entailment recognition corpus using the news headline and the first paragraph of a news article as near-paraphrases. Their approach has an estimated accuracy of 70% on a held out set of 500 pairs. The primary limitation of the approach is that it only generates positive training examples. Hickl et al. (2006) improves upon this work by including negative examples selected using heuristic rules (e.g., sentences connected by *although*, *otherwise*, and *but*). On RTE-2 their method achieves accuracy improvements of upto 10%. However, Hickl and Bensley (2007) achieves only a 1% accuracy improvement on RTE-3 using the same method, suggesting that it is not always as beneficial.

Recent work by Bowman et al. (2015) describes a method for generating large scale annotated datasets, viz., the Stanford Natural Language Inference (SNLI) Corpus, for the problem of entailment recognition. They use Amazon Mechanical Turk to very inexpensively produce a large entailment annotated data set from image captions.

Zanzotto and Pennacchiotti (2010) create an entailment corpus using Wikipedia data. They hand-annotate original Wikipedia entries, and their associated revisions for entailment recognition. Using a previously published system for RTE (Zanzotto and Moschitti, 2006), they show that their expanded corpus does not result in improvement for RTE-1, RTE-2 or RTE-3.

Similarly, Celikyilmaz et al. (2009) address the lack of labeled data by semi-automatically creating an entailment corpus, which they use within their question answering system. They reuse text-hypothesis pairs from RTE challenges in addition to manually annotated pairs from a newswire corpus (with pairs for annotation obtained through a Lucene search over the corpus).

Note that all of the above research on expanding the labeled data for entailment has focused on *entailment recognition*. Our focus in this paper is on improving *entailment search* by exploiting unlabeled data with self-training and active learning.

3 Datasets

In this section, we describe the data sets from two domains, *newswire* and *clinical*, that we use in the development and evaluation of our work.

3.1 Newswire Domain

For the newswire domain, we use entailment search data from the PASCAL RTE-5, RTE-6 and RTE-7 challenges (Bentivogli et al., 2009; Bentivogli et al., 2010; Bentivogli et al., 2011). The dataset consists of a corpus of news documents, along with a set of hypotheses. The hypotheses come from a separate summarization task, where

[1] http://lucene.apache.org

Dataset	Size	Entailing
Newswire-train	20,104	810 (4.0%)
Newswire-dev	35,927	1,842 (5.1%)
Newswire-test	17,280	800 (4.6%)
Newswire-unlabeled	43,485	-
Clinical-train	7,026	293 (4.1%)
Clinical-dev	8,092	324 (4.0%)
Clinical-test	10,466	596 (5.6%)
Clinical-unlabeled	623,600	-

Table 1: Summary of datasets

```
**NAME[XX (YY) ZZ] has no liver
problems.
PAST MEDICAL HISTORY
1.  Htn
Well controlled
2.  Diabetes mellitus
On regular dose of insulin.

FAMILY HISTORY:
Father with T2DM age unknown
```

Figure 1: Excerpt from a sample clinical note

the summary sentences about a news story (given a topic) were manually created by human annotators. These summary sentences are used as hypotheses in the dataset. Entailment annotations are then provided for a subset of sentences from the document corpus, based on a Lucene filter for each hypothesis.

In this work, we use the RTE-5 development data to train our system (*Newswire-train*), RTE-5 test data for evaluation of our systems (*Newswire-test*), and we use the combined RTE-6 development and test data for our system development and parameter estimation (*Newswire-dev*). We use all of the development and test data from RTE-7, without the human annotation labels, as our unlabeled data (*Newswire-unlabeled*) for self-training and active learning experiments. A summary of the newswire data is shown in Table 1.

3.2 Clinical Domain

There are no public datasets available for textual entailment *search* in the clinical domain. In creating this dataset, we imagine a real-world clinical situation where hypotheses are facts about a patient that a physician seeing the patient might want to learn (e.g., *The patient underwent a surgical procedure within the last three months.*). The unstructured notes in the patients electronic medical record (EMR) is the text against which a system would determine the entailment status of the given hypotheses.

Observe that the aforementioned real-world clinical scenario is very closely related to a question answering problem, where instead of hypotheses a physician may pose natural language questions seeking information about the patient (e.g., *Has this patient undergone a surgical procedure within the past three months?*). Answers to such questions are words, phrases or passages from the patient's EMR. Since we have access to a patient-specific question answering dataset over EMRs[2] (henceforth, referred to as the QA dataset), we use it here as our starting point in constructing the clinical domain textual entailment dataset.

Given a question answering dataset, how might one go about creating a dataset on textual entailment? We follow a methodology similar to that of RTE-1 through RTE-5 for entailment set derived from question answering data. The text corpus in our entailment dataset is the set of de-identified patient records associated with the QA dataset. To generate hypotheses, human annotators converted questions into multiple assertive sentences, which is somewhat similar to what was done in the first five RTE challenges (RTE-1 through RTE-5). For a given question, the human annotators plugged in clinically-plausible answers to convert the question into a statement that may or may not be true about a given patient. Table 2 shows example hypotheses and their source questions. Note that this procedure for hypothesis generation diverges slightly from the RTE procedure, where answers from a question answering system were plugged into the questions to produce assertive sentences.

To generate entailment annotations, we paired a hypothesis with every sentence in a subset of clinical notes of the EHR, and asked human annotators to determine if the note sentence enabled them to conclude an entailment relationship with the hypothesis. For example, the text: *"The appearance is felt to be classic for early MS."* entails the hypothesis: *"She has multiple sclerosis"*. While in the RTE procedure, a Lucene search was used as a filter to limit the number of hypothesis-sentence pairs that are annotated, in our clinical dataset we

[2]a publication describing the question-answering dataset is currently under review at another venue

Question	Hypotheses
When was the patient diagnosed with dermatomyositis?	`The patient was diagnosed with dermatomyositis two years ago.`
Any creatinine elevation?	`Creatinine is elevated.`
	`Creatinine is normal.`
Why were xrays done on the forearm and hand?	`Xrays were done on the forearm and hand for suspected fracture.`

Table 2: Example question → hypotheses mappings

limit the number of annotations by pairing each hypothesis only with sentences from EMR notes containing an answer to the original question in the QA dataset.

The entailment annotations were generated by two medical students with the help of the annotations generated for QA. 11 medical students created our QA dataset of 5696 questions over 71 patient records, of which 1747 questions have corresponding answers. This was generated intermittently over a period of 11 months. Given the QA dataset, the time taken to generate entailment annotations includes conversion of questions to hypotheses, and annotating entailment. While conversion of questions to hypotheses took approx. 2 hours for 20 questions, generating about 3000 hypothesis and text pairs took approx. 16 hours.

At the end of this process, we had a total of 243 hypotheses annotated against sentences from 380 clinical notes, to generate 25,584 text-hypothesis pairs. We split this into train, development and test sets, summarized in Table 1. Although we have a fairly limited number of labeled text-hypothesis pairs, we do have a large number of patient health records (besides the ones in the annotated set). We generated unlabeled data in the clinical domain, by pairing the hypotheses from our training data with sentences from a set of randomly sampled subset of health records outside of the annotated data.

Datasets for the textual entailment search task are highly skewed towards the non-entailment class. Note that our clinical data, while smaller in size than the newswire data, maintains a similar class imbalance.

4 Supervised Entailment System

We begin by defining, in this section, our supervised entailment system (called ENT) that is used as the basis of our self-training and active learning experiments. Our system draws upon characteristics and features of systems that have previously been successful in the RTE challenges in the newswire domain. We further enhance this system with new features targeting the clinical domain. The purpose of this section is to demonstrate, through an experimental comparison with other entailment systems, that ENT is competitive on *both* domains, and is a reasonable supervised system to use in our investigations into self-training and active learning.

4.1 System Description

Top systems (Tsuchida and Ishikawa, 2011; Mirkin et al., 2009) in the RTE challenges have used various types of passage matching approaches in combination with machine learning for entailment. We follow along these lines, and design a classifier-based entailment system. For every text-hypothesis pair in the dataset we extract a feature vector representative of that pair. Then, using the training data, we train a classifier to make entailment decisions on unseen examples. In our system, we employ a logistic regression with ridge estimator (the Weka implementation (Hall et al., 2009)), powered by a variety of passage matching features described below.

Underlying many of our passage match features is a more fine-grained notion of "term match". *Term matchers* are a set of algorithms that attempt to match tokens (including multi-word tokens, such as *New York* or *heart attack*) across a pair of passages. One of the simplest examples of these is *exact string matcher*. A token in one text passage that matches exactly, character-for-character, with a token in another text passage would be considered a term match by this simple term matcher. However, these term matchers could be more sophisticated and match pairs of terms that are synonyms, or paraphrases, or

Exact	String match, ignore case
Multi-word	Overlapping terms in multi-word token
Head	String match head of multi-word token
Wikipedia	Wikipedia redirects and disamb. pages
Morphology	Derivational morphology, e.g. *archaeological* → *archaeology*
Date+Time	Match normalized dates and times
Verb resource	Match verbs using WordNet, Moby thesaurus, manual resources
UMLS	Medical concept match using UMLS
Translation	Affix-rule-based translation of medical terms to layman terms

Table 3: ENT term matchers

equivalent to one another according to other criteria. ENT employs a series of term matchers listed in Table 3. Each of these may also produce a confidence score for every match they find. Because we are working with clinical data, we added some medical domain term matchers as well – using UMLS (Bodenreider, 2004) and a rule-based "translator" of medical terms to layman terms[3].

Listed below are all of our features used in the ENT's classifier. Most passage match features aggregate the output of the term matchers along various linguistic dimensions – lexical, syntactic, semantic, and document/passage characteristics.

Lexical: This set includes a feature aggregating *exact string matches* across text-hypothesis, one aggregating *all term matchers*, a feature counting *skip-bigram matches* (using all matchers), a measure of *matched term coverage of text* (ratio of matched terms to unmatched terms). Additionally, we have some medical domain features, viz. *UMLS concept overlap*, and a measure of *UMLS-based similarity* (Shivade et al., 2015; Pedersen et al., 2007) using the UMLS::Similarity tool (McInnes et al., 2009).

Syntactic: Following the lead of several approaches textual entailment (Wang and Zhang, 2009; Mirkin et al., 2009; Kouylekov and Negri, 2010) we have a features measuring the *similarity of parse trees*. Our rule-based syntactic parser (McCord, 1989) produces dependency parses the text-hypothesis pair, whose nodes are aligned using all of the term matchers. The tree match feature is an aggregation of the aligned subgraphs in the tree (somewhat similar to a tree kernel (Moschitti, 2004)).

Semantic: We apply open domain as well as medical entity and relation detectors (Wang et al., 2011; Wang et al., 2012) to the texts, and post features measuring *overlap in detected entities* and *overlap in the detected relations* across the text-hypothesis pair. We also have a rule-based semantic frame detector for a "medical finding" frame (patient presenting with symptom or disease). We post a feature that aggregates matched *elements of detected frames*.

Passage Characteristics: Clinical notes typically have a structure and the content is often organized in sections (e.g. *History of Illness* followed by *Physical Examination* and ending with *Assessment and Plan*). We identified the section in which each note sentence was located and used them as features in the classifier. Clinical notes are also classified into many different categories (e.g., *discharge summary*, *radiology report*, etc.), which we generate features from. We also generate several features capturing the "readability" of the text segments – *parse failure*, *list detector*, *number of verbs*, *word capitalization*, *no punctuation* and *sentence size*. We also have *a measure of passage topic relevance* based on medical concepts in the pair of texts.

4.2 System Performance

To compare effectiveness of ENT on the entailment task, we chose two publicly available systems – EDITS and TIE – for comparison. Both these system are available under the Excitement Open Platform (EOP), an initiative (Magnini et al., 2014) to make tools for textual entailment freely available[4] to the NLP community. EDITS (Edit Distance Textual Entailment Suite) by Kouylekov and Negri (2010) is an open source textual entailment system that uses a set of rules and resources to perform "edit" operations on the text to convert it into the hypothesis. There are costs associated with the operations, and an overall cost is computed for the text-hypothesis pair, which determines the decision for that pair. This system has placed third (out of eight teams) in RTE-5, and seventh (out of thirteen teams) in RTE-7. The Textual Inference Engine (TIE) (Wang and Zhang, 2009) is a maximum entropy based entailment system relying on predicate argument structure matching. While this system did not partici-

[3]Rules for medical term translator were derived from http://www.globalrph.com/medterm.htm

[4]http://hltfbk.github.io/Excitement-Open-Platform/

System	Newswire			Clinical		
	Precision	Recall	F-score	Precision	Recall	F-score
Lucene	0.47	0.48	**0.47***	0.16	0.22	0.19
EDITS	0.22	0.57	0.32	0.23	0.21	0.20
TIE	0.66	0.21	0.31	0.43	0.01	0.02
ENT	0.77	0.26	0.39	0.42	0.15	**0.23***

Table 4: System performance on test data (* indicates statistical significance)

pate in the RTE challenges, it has been shown to be effective on the RTE datasets. In our experiments, we trained the EDITS system optimizing for F-score (the default optimization criterion is accuracy) and TIE with its default settings. We also used a Lucene baseline similar to the one used in RTE-5, RTE-6 and RTE-7 entailment challenges.

We trained the systems on the training set of each domain and tested on the test set. The Lucene baseline considers the first N sentences (where N is 5, 10, 15 or 20) top-ranked by the search engine to be entailing the hypothesis. The configuration with the top 10 sentences performed the best, and is reported in the results. Note that this baseline is a strong one, and none of the systems participating in RTE-5 could beat it.

Table 4 summarizes the system performance on newswire and clinical data. We observe that systems that did well on RTE datasets, were mediocre on the clinical dataset. We did not, however, put any effort into adaption of TIE and EDITS to the clinical data. So the mediocre performance on clinical is understandable. It is interesting to see though that ENT did well (comparatively) on both domains.

We note that our problem setting is most similar to the RTE-5 entailment search task. Of the 20 runs across eight teams that participated in RTE-5, the median F-Score was 0.30 and the best system (Mirkin et al., 2009) achieved an F-Score of 0.46. EDITS and TIE perform slightly above the median and ENT (with 0.39 F-score) would have ranked third in the challenge.

The performance of all systems on the clinical data is noticeably low as compared to the newswire data. An obvious difference in the two domains is the training data size (see Table 1). However, obtaining annotations for textual entailment search is expensive, particularly in the clinical domain. The remaining sections present our investigations into self-training and active learning, to overcome the lack of training data.

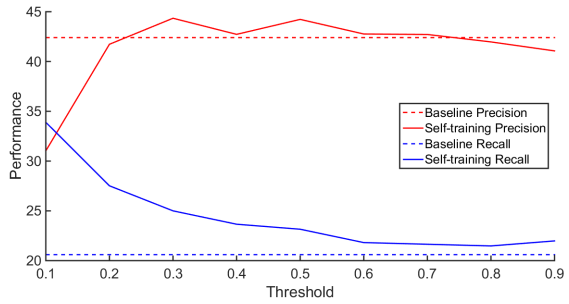
5 Self-Training

Our goal is to exploit unlabeled data, with the hope of augmenting the limited annotated data in a given domain. Self-training is a method that has been successfully used to address limited training data on many NLP tasks, such as parsing (McClosky et al., 2006), information extraction (Huang and Riloff, 2012; Patwardhan and Riloff, 2007), word sense disambiguation (Mihalcea, 2004), etc. Self-training iteratively increases the size of the training set, by automatically assigning labels to unlabeled examples, using a model trained in a previous iteration of the self-training regime.

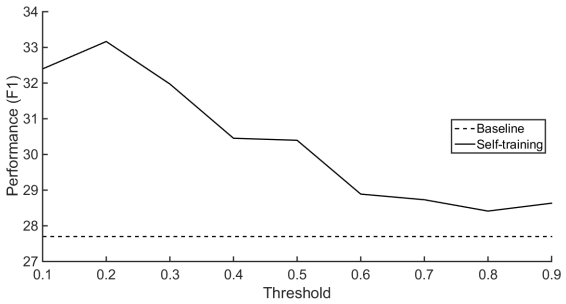
For our newswire and clinical datasets, using the set of unlabeled text-hypothesis pairs U, we ran the following training regime: A model was created using the training data L_n, and applied it to the unlabeled data U. From U, all such pairs that were classified by the model as entailing pairs with high confidence (above a threshold τ) were added to the labeled training data L_n to generate L_{n+1}. Non-entailing pairs were ignored. A new model is trained on data L_{n+1}, and the above process repeated iteratively, until a stopping criteria is reached (in our case, all pairs from U are exhausted).

The threshold τ determines the confidence of our model for a text-hypothesis pair being classified to the entailment class. This threshold was tuned by varying it incrementally from 0.1 to 0.9 in steps of 0.1. The best τ was determined on the development set, and chosen for the self-training system. Figure 2 shows the effect of τ on the development data.

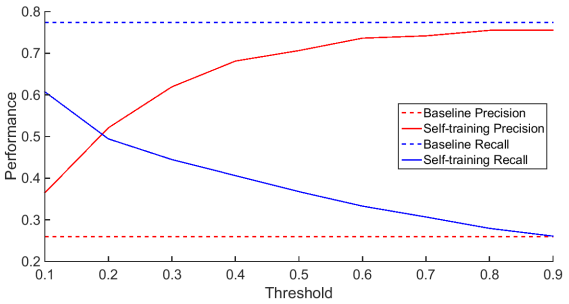
As such, we see that the F-score of the self-trained model is always above that of the baseline ENT system. The F-score increases upto a peak of 0.33 at threshold τ of 0.2 before dropping at higher thresholds. Using this tuned threshold on test set, the comparitive performance on the test set is outlined in Table 5. We observe an F-score



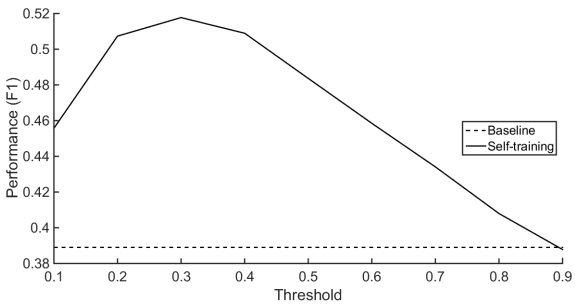
(a) Precision and recall for clinical data



(b) F-Score for clinical data



(c) Precision and recall for newswire data



(d) F-Score for newswire data

Figure 2: Self-training on development data

System	Newswire			Clinical		
	Precision	**Recall**	**F-score**	**Precision**	**Recall**	**F-score**
ENT	0.77	0.26	0.39	0.42	0.15	0.23
ENT + Self-Training	0.62	0.48	**0.54***	0.34	0.39	**0.36***

Table 5: Self-training results on test data (* indicates statistical significance)

of 0.36, which is significantly greater than that of the vanilla ENT system (0.23).

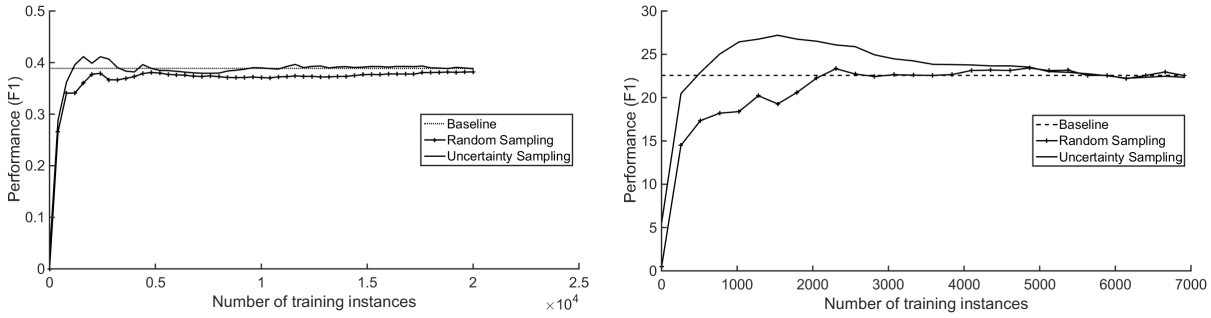
The effect of the threshold on performance correlates with the number of instances added to the training set. When the threshold is low, there are more instances being added (10,799 at threshold of 0.1) into the training set. Therefore, recall is likely to benefit, since the model is exposed to a larger variety of text-hypothesis pairs. However, the precision is low since noisy pairs are likely to be added. When the threshold is high, fewer instances are added (316 at threshold of 0.9). These are the ones that the model is most certain about, suggesting that these are likely to be less noisy. Therefore, the precision is comparatively high.

We also ran our self-training approach on the Newswire datasets. We observed similar variations in performance with newswire data as with the clinical data. At threshold of 0.9, fewer instances (49) are added to the training set from the unlabeled data, while a large number of instances (2,861) are added at a lower threshold τ of 0.1. The best performance (F-score of 0.52) was obtained at threshold of 0.3, on the development set.

This threshold also resulted in the best performance (0.54) on the test set. Similar to the clinical domain, precision increased but recall decreased as the threshold increased. Again, it is evident from Table 5 that gains obtained from self-training are due to recall. It should be noted that the self-trained system achieves an F-score of 0.54 – substantially better than the best performing system of Mirkin et al. (2009) (F-score, 0.46) in RTE-5.

6 Active Learning

Active learning is a popular training paradigm in machine learning (Settles, 2012) where a learning agent interacts with its environment in acquiring a training set, rather than passively receiving independent samples from an underlying distribution. This is especially pertinent in the clinical domain, where input from a medical professional should be sought only when really necessary, because of the high cost of such input. The purpose of exploring



(a) Newswire Data

(b) Clinical Data

Figure 3: Learning curves for uncertainty sampling and random sampling on test data

this paradigm is to achieve the best possible generalization performance at the lowest cost.

Active learning is an iterative process, and typically works as follows: a model M is trained using a minimal training dataset L. A query framework is used to identify an instance from an unlabeled set U that, if added to L, will result in maximum expected benefit. Gold standard annotations are obtained for this instance and added to the original training set L to generate a new training set L'. In the next iteration, a new model M' is trained using L' and used to identify the next most beneficial instance for the training set L'. This is repeated until a stopping criterion is met. This approach is often *simulated* using a training dataset L of reasonable size. The initial model M is created using a subset A of L. Further, instead of querying a large unlabeled set U, the remaining training data $(L - A)$ is treated as an unlabeled dataset and queried for the most beneficial addition.

We carried out active learning in this setting using a querying framework known as *uncertainty sampling* (Lewis and Gale, 1994). Here, the model M trained using A, queries the instances in $(L-A)$ for instance(s) it is least certain for a prediction label. For probabilistic classifiers the most uncertain instance is the one where posterior probability for a given class is nearest to 0.5. To estimate the effectiveness of this framework, it is always compared with a *random sampling* framework, where random instances from the training data are incrementally added to the model.

Starting with a model trained using a single randomly chosen instance, we carried out active learning using uncertainty sampling, adding one instance at a time. After the addition of each instance, the model was retrained and tested on a held out set. To minimize the effect of randomization associated with the first instance, we repeated the experiment ten times and averaged the performance scores across the ten runs.

Following previous work (Settles and Craven, 2008; Reichart et al., 2008) we evaluate active learning using learning curves on the test set. Figure 3 shows the learning curves for newswire and clinical data.

On clinical data, uncertainty sampling achieves a performance equal to the baseline ENT with only 470 instances. With random sampling, over 2,200 instances are required. The active learner matches the performance of the ENT with only 6.6% of training data. Newswire shows a similar trend, with both sampling strategies outperforming ENT, using less than half the training, and uncertainty sampling learning faster than random. While uncertainty sampling matches ENT F-score with only 1,169 instances, random sampling requires 2,305. Here, the active learner matches ENT performance using only 5.8% of the training data.

7 Effect of Class Distribution

After analyzing our experimental results, we considered that one possible explanation for the improvements over baseline ENT could plausibly be because of changes in the class distribution. From Table 1, we observe that the distribution of classes in both domains is highly skewed (only 4-5% positive instances). Self-training and active learning dramatically change the class distribution in training. To assess the effect of class distribution changes on performance, we ran additional experiments, described here.

We first investigated sub-sampling (Japkowicz, 2000) the training data to address class imbalance. This includes down-sampling the majority class or up-sampling the minority class until the classes are

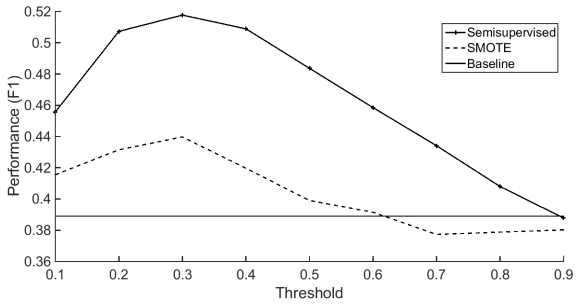


Figure 4: Comparison of SMOTE and self-training (on newswire development set)

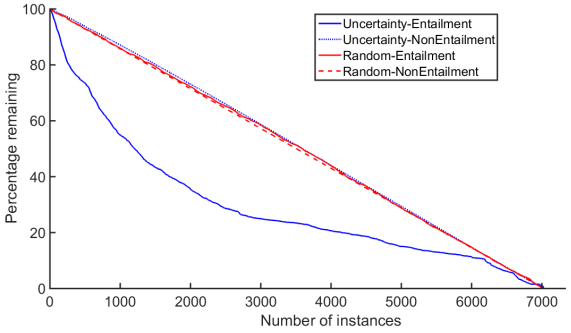


Figure 5: Comparison of sampling strategies for active learning (on newswire development set)

balanced. We found no significant gains over the vanilla ENT baseline with both strategies. Specifically, down-sampling resulted in gains of only 0.002 and 0.001 F-score and up-sampling resulted in a drop of 0.011 and 0.013 F-score on clinical-dev and newswire-dev, respectively.

Another approach to addressing class imbalance is to apply Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al., 2002). SMOTE creates instances of the minority class by taking a minority class sample and introducing synthetic examples between its k nearest neighbors. Using SMOTE on newswire and clinical datasets resulted in improvements over baseline ENT in both domains. The improvements using self-training, however, are significantly higher than SMOTE. Figure 4 shows a comparison of SMOTE and self-training on newswire data, where equal number of instances are added to the training set by both techniques.

Finally, for active learning, we consider random sampling as a competing approach to uncertainty sampling. Figure 5 illustrates the percentage of positive and negative instances that get included in the training set for both sampling strategies, as active learning proceeds. The blue solid line shows that positive instances are *consumed* faster than the negative instances with uncertainty sampling. Thus, a higher percentage of positive instances (that approximately equals the number of negative instances getting added) get added and this helps maintain a balanced class distribution.

Once the positive instances are exhausted, more negative instances are added, resulting in some class imbalance that hurts performance (even though more training data is being added overall). In contrast, random sampling does not change the class balance, as it *consumes* a proportional number of positive and negative instances (resulting in more negative than positive instances). The plot indicates that when using uncertainty sampling 80% of the positive examples are added to the training set with less than 50% of the data. This also explains how the active learner matches the performance of the model using the entire labeled set, but with fewer training examples.

8 Conclusion

We explored the problem of textual entailment search in two domains – newswire and clinical – and focused a spotlight on the cost of obtaining labeled data in certain domains. In the process, we first created an entailment dataset for the clinical domain, and a highly competitive supervised entailment system, called ENT, which is effective (out of the box) on two domains. We then explored two strategies – self-training and active learning – to address the lack of labeled data, and observed some interesting results. Our self-training system substantially improved over ENT, achieving an F-score gain of 15% on newswire and 13% on clinical, using only additional unlabeled data. On the other hand, our active learning experiments demonstrated that we could match (and even beat) the baseline ENT system with only 6.6% of the training data in the clinical domain, and only 5.8% of the training data in the newswire domain.

Acknowledgments

We thank our in-house medical expert, Jennifer Liang, for guidance on the data annotation task, our medical annotators for annotating clinical data for us, and Murthy Devarakonda for valuable insights during the project. We also thank Eric Fosler-Lussier and Albert M. Lai for their help in conceptualizing this work.

References

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. 2009. The Fifth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, MD.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2010. The Sixth PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the Third Text Analysis Conference*, Gaithersburg, MD.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2011. The Seventh PASCAL Recognizing Textual Entailment Challenge. In *Proceedings of the Fourth Text Analysis Conference*, Gaithersburg, MD.

Olivier Bodenreider. 2004. The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. *Nucleic Acids Research*, 32(Database Issue):D267–D270.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal.

John Burger and Lisa Ferro. 2005. Generating an Entailment Corpus from News Headlines. In *Proceedings of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 49–54, Ann Arbor, MI.

Asli Celikyilmaz, Marcus Thint, and Zhiheng Huang. 2009. A Graph-based Semi-Supervised Learning for Question-Answering. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 719–727, Singapore.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research*, 16:321–357.

Ido Dagan, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. Recognizing Textual Entailment: Models and Applications. *Synthesis Lectures on Human Language Technologies*, 6(4):1–220.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software. *ACM SIGKDD Explorations Newsletter*, 11(1):10.

Andrew Hickl and Jeremy Bensley. 2007. A Discourse Commitment-based Framework for Recognizing Textual Entailment. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 171–176, Prague, Czech Republic.

Andrew Hickl, Jeremy Bensley, John Williams, Kirk Roberts, Bryan Rink, and Ying Shi. 2006. Recognizing Textual Entailment with LCC's GROUNDHOG System. In *The Second PASCAL Recognizing Textual Entailment Challenge: Proceedings of the Challenges Workshop*, Venice, Italy.

Ruihong Huang and Ellen Riloff. 2012. Bootstrapped Training of Event Extraction Classifiers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 286–295, Avignon, France.

Nathalie Japkowicz. 2000. Learning from Imbalanced Data Sets: A Comparison of Various Strategies. In *Proceedings of the AAAI Workshop on Learning from Imbalanced Data Sets*, pages 10–15, Austin, TX.

Houping Jia, Xiaojiang Huang, Tengfei Ma, Xiaojun Wan, and Jianguo Xiao. 2010. PKUTM Participation at TAC 2010 RTE and Summarization Track. In *Proceedings of the Third Text Analysis Conference*, Gaithersburg, MD.

Milen Kouylekov and Matteo Negri. 2010. An Open-Source Package for Recognizing Textual Entailment. In *Proceedings of the ACL 2010 System Demonstrations*, Uppsala, Sweden.

David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, Ireland.

Bernardo Magnini, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Neumann Guenter, Tae-Gil Noh, Sebastian Padó, Asher Stern, and Omer Levy. 2014. The Excitement Open Platform for Textual Inferences. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 43–48, Baltimore, MD.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective Self-Training for Parsing. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 152–159, New York City, NY.

Michael McCord. 1989. Slot Grammar: A System for Simpler Construction of Practical Natural Language Grammars. In *Proceedings of the International Symposium on Natural Language and Logic*, pages 118–145, Hamburg, Germany.

Bridget T McInnes, Ted Pedersen, and Serguei V. S. Pakhomov. 2009. UMLS-Interface and UMLS-Similarity : Open Source Software for Measuring Paths and Semantic Similarity. In *Proceedings of the Annual Symposium of the American Medical Informatics Association*, San Francisco, CA.

Rada Mihalcea. 2004. Co-Training and Self-Training for Word Sense Disambiguation. In *Proceedings of the Eighth Conference on Natural Language Learning*, pages 33–40, Boston, MA.

Shachar Mirkin, Roy Bar-Haim, Jonathan Berant, Ido Dagan, Eyal Shnarch, Asher Stern, and Idan Szpektor. 2009. Addressing Discourse and Document Structure in the RTE Search Task. In *Proceedings of the Second Text Analysis Conference*, Gaithersburg, MD.

Alessandro Moschitti. 2004. A Study on Convolution Kernels for Shallow Statistic Parsing. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Barcelona, Spain.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 717–727, Prague, Czech Republic.

Ted Pedersen, Serguei V. S. Pakhomov, Siddharth Patwardhan, and Christopher G Chute. 2007. Measures of Semantic Similarity and Relatedness in the Biomedical Domain. *Journal of Biomedical Informatics*, 40(3):288–99.

Roi Reichart, Katrin Tomanek, Udo Hahn, and Ari Rappoport. 2008. Multi-Task Active Learning for Linguistic Annotations. In *Proceedings of ACL-08: HLT*, pages 861–869, Columbus, OH.

Burr Settles and Mark Craven. 2008. An Analysis of Active Learning Strategies for Sequence Labeling Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1070–1079, Honolulu, HI.

Burr Settles. 2012. Active Learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Chaitanya Shivade, Courtney Hebert, Marcelo Loptegui, Marie-Catherine de Marneffe, Eric Fosler-Lussier, and Albert M. Lai. 2015. Textual Inference for Eligibility Criteria Resolution in Clinical trials. *Journal of Biomedical Informatics*, 58:S211–S218.

Masaaki Tsuchida and Kai Ishikawa. 2011. IKOMA at TAC2011 : A Method for Recognizing Textual Entailment using Lexical-level and Sentence Structure-level Features. In *Proceedings of the Fourth Text Analysis Conference*, Gaithersburg, MD.

Rui Wang and Yi Zhang. 2009. Recognizing Textual Relatedness with Predicate-Argument Structures. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 784–792, Singapore.

Chang Wang, James Fan, Aditya Kalyanpur, and David Gondek. 2011. Relation Extraction with Relation Topics. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1426–1436, Edinburgh, UK.

Chang Wang, Aditya Kalyanpur, James Fan, Branimir K. Boguraev, and David Gondek. 2012. Relation Extraction and Scoring in DeepQA. *IBM Journal of Research and Development*, 56(3.4):9:1–9:12.

Fabio M. Zanzotto and Alessandro Moschitti. 2006. Automatic Learning of Textual Entailments with Cross-Pair Similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 401–408, Sydney, Australia.

Fabio M. Zanzotto and Marco Pennacchiotti. 2010. Expanding Textual Entailment Corpora from Wikipedia using Co-training. In *Proceedings of the 2nd Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, pages 28–36, Beijing, China.

Annotating and Predicting Non-Restrictive Noun Phrase Modifications

Gabriel Stanovsky **Ido Dagan**
Computer Science Department, Bar-Ilan University
gabriel.satanovsky@gmail.com
dagan@cs.biu.ac.il

Abstract

The distinction between restrictive and non-restrictive modification in noun phrases is a well studied subject in linguistics. Automatically identifying non-restrictive modifiers can provide NLP applications with shorter, more salient arguments, which were found beneficial by several recent works. While previous work showed that restrictiveness can be annotated with high agreement, no large scale corpus was created, hindering the development of suitable classification algorithms. In this work we devise a novel crowdsourcing annotation methodology, and an accompanying large scale corpus. Then, we present a robust automated system which identifies non-restrictive modifiers, notably improving over prior methods.

1 Introduction

Linguistic literature provides a large body of research distinguishing between two types of modifiers within noun phrases: (1) *Restrictive modifiers*, which constitute an integral part of the entity referenced by the NP, e.g., the underlined modifier in "*She wore the necklace that her mother gave her*", versus (2) Non-restrictive modifiers, which provide an additional or parenthetical information on an already definite entity, e.g., "*The speaker thanked president Obama who just came into the room*" (Huddleston et al., 2002; Fabb, 1990; Umbach, 2006).

The distinction between the two types is semantic in nature and relies heavily on the context of the NP. Evidently, many syntactic constructions can appear in both restrictive and non-restrictive uses. While the previous examples were of relative clauses, Figure 1 demonstrates this distinction in various other syntactic constructions.

Identifying and removing non-restrictive modifiers yields shorter NP arguments, which proved beneficial in many NLP tasks. In the context of abstractive summarization (Ganesan et al., 2010) or sentence compression (Knight and Marcu, 2002), non-restrictive modifiers can be removed to shorten sentences, while restrictive modification should be preserved.

Further, recent work in information extraction showed that shorter arguments can be beneficial for downstream tasks. Angeli et al. (2015) built an Open-IE system which focuses on shorter argument spans, and demonstrated its usefulness in a state-of-the-art Knowledge Base Population system. Stanovsky et al. (2015) compared the performance of several off-the-shelf analyzers in different semantic tasks. Most relevant to this work is the comparison between Open-IE and Semantic Role Labeling (Carreras and Màrquez, 2005). Specifically, they suggest that SRL's longer arguments introduce noise which hurts performance for downstream tasks.

Finally, in question answering, omitting non-restrictive modification can assist in providing more concise answers, or in matching between multiple answer occurrences.

Despite these benefits, there is currently no consistent large scale annotation of restrictiveness, which hinders the development of automatic tools for its classification. In prior art in this field, Dornescu et al. (2014) used trained annotators to mark restrictiveness in a large corpus. Although they reached good agreement levels in restrictiveness annotation, their corpus suffered from inconsistencies, since it conflated restrictiveness annotation with inconsistent modifier span annotation.

The contributions of this work are twofold. Primarily, we propose a novel crowdsroucing anno-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1256–1265,
Berlin, Germany, August 7-12, 2016.

tation methodology which decouples the binary (restrictive / non-restrictive) distinction from the modifier span annotation (Section 3). Following this methodology, in Section 4 we present a large scale annotated corpus, which will allow further research into the automatic identification of non-restrictive modification.

Additionally, we developed a strong automatic classifier, which learns from our new corpus (Section 5). This classifier uses new linguistically motivated features which are robust enough to perform well over automatically predicted parse trees.

The corpus and the automatic classifier are both made publicly available.[1]

While there is still much room for improvement, especially in some of the harder, more context-dependent, cases (most notably, prepositional and adjectival modifiers), our system provides an applicable means for identifying non-restrictive modification in a realistic NLP setting.

2 Background

In this section we cover relevant literature from several domains. In Section 2.1 we discuss the established linguistic distinction between restrictive and non-restrictive modification. Following, in Section 2.2 we discuss previous NLP work on annotating and identifying this distinction. Finally, in Section 2.3 we briefly describe the recent QA-SRL annotation paradigm (He et al., 2015), which we utilize in Section 3 as part of our annotation scheme.

2.1 Non-Restrictive Modification

Throughout the paper we follow Huddleston et al.'s (2002) well-known distinction between two types of NP modifiers: (1) *Restrictive modifiers*, for which the content of the modifier is an integral part of the meaning of the containing NP, and, in contrast, (2) *Non-restrictive modifiers*, that present a separate, parenthetical unit of information about the NP.

While some syntactic modifiers (such as determiners or genitives) are always restrictive, others are known to appear in both restrictive as well as non-restrictive uses, depending on semantics and context (Huddleston et al., 2002; Fabb, 1990; Umbach, 2006). Among these are relative clauses, adjectival, prepositional, non-finite, and verbal modifiers. See Figure 1 for examples of different syntactic constructions appearing in both restrictive as well as non-restrictive contexts.

(RC1) The necklace *that her mother gave her*$^{+}$ is in the safe.
(RC2) The governor disagreed with the U.S ambassador to China *who seemed nervous*$^{-}$.
(NF1) People *living near the site*$^{+}$ will have to be evacuated.
(NF2) sheriff Arthur Lester, *standing against the wall*$^{-}$, looked tired.
(PP1) The kid *from New York*$^{+}$ won the lottery.
(PP2) The assassination of Franz Ferdinand *from Austria*$^{-}$ started WWI.
(AD1) The *good*$^{+}$ boys won.
(AD2) The water level rose a *good*$^{-}$ 12 inches.

Figure 1: Restrictive (marked in red and a plus sign) and non-restrictive (marked in blue and a minus sign) examples in different syntactic constructions, see elaboration in Section 2. Examples index: RC - Relative clause, NF - Non-finite clauses (Huddleston et al. [p. 1265]), PP - Prepositional modifiers, AD - Adjectival modifiers (Huddleston et al. [p. 528]).

For example, for *relative clause*, Huddleston et al. [p. 1058] identifies both restrictive as well as non-restrictive uses (for which they use the terms *integrated* and *supplementary*, respectively). In the sentence marked (RC1), the highlighted relative clause is restrictive, distinguishing the necklace being referred to from other necklaces, while in sentence (RC2), the relative clause does not pick an entity from a larger set, but instead presents separate information about an already specified definite entity.

2.2 Non-Restrictive Modification in NLP

Syntactic and semantic annotations generally avoid the distinction between restrictive and non-restrictive modification (referred here as "restrictiveness" annotation).

The syntactic annotation of the Penn TreeBank (Marcus et al., 1993) and its common conversion to dependency trees (e.g., (de Marneffe and Manning, 2008)) do not differentiate the cases discussed above, providing the same syntactic structure for the semantically different instances. See Figure 2 for an example.

Furthermore, prominent semantic annotations,

[1] http://www.cs.biu.ac.il/~nlp/resources/downloads

such as PropBank (Palmer et al., 2005), AMR (Banarescu et al., 2013), CCG (Hockenmaier and Steedman, 2007), or FrameNet (Baker et al., 1998), also avoid this distinction. For example, PropBank does not differentiate between such modifiers, treating both types of modification as an integral part of an argument NP.

Two recent works have focused on automatically identifying non-restrictive modifications. Honnibal et al. (2010) added simple automated restrictiveness annotations to NP-modifiers in the CCGbank (Hockenmaier and Steedman, 2007). Following a writing style and grammar rule, a modifier was judged as non-restrictive if and only if it was preceded by a comma.[2] This annotation was not intrinsically evaluated, as it was carried as part of an extrinsic evaluation of a statistical parser.

Having similar goals to ours, Dornescu et al. (2014) sets the prior art at annotating and predicting non-restrictive modification. In the annotation phase, each of their trained annotators was asked to (1) Mark spans of words in the sentence as forming an NP modifier, and (2) Mark each span they annotated in (1) as either restrictive or non-restrictive, and specify its type from a predefined list (e.g., relative clause, adjectival modifier, etc.).

Their inter-annotator agreement on the first task (modifier span) was low, reaching pairwise F1 score of only 54.9%, possibly due to problems in the annotation procedure, as acknowledged by the authors. The second part of the annotation achieved better agreement levels, reaching kappa of 0.78 (substantial agreement) for type annotation and 0.51 (moderate agreement) for restrictiveness annotation.[3]

Following the creation of the annotated dataset, they developed rule based and machine learning classifiers. All of their classifiers performed only at about 47% F1, at least partly due to the inconsistencies in span annotation discussed above.

To conclude this survey, although an effort was made by Dorenscu et al. (2014), there is currently no available consistent corpus annotated with non-restrictive modifiers.

[2]Notice that this is indeed the case in some of the non-restrictive examples in Figure 1.

[3]Note that the agreement for the first task is reported in F1 while the second task is reported in Cohen's kappa.

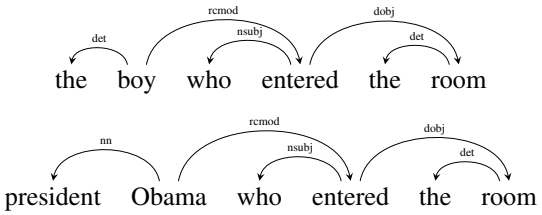


Figure 2: Restrictive (top) and non-restrictive (bottom) NP modifications receive the same representation in dependency trees. See Section 2.2.

2.3 QA-SRL

Traditional Semantic Role Labeling (SRL) (Carreras and Màrquez, 2005) is typically perceived as answering **argument role questions**, such as *who, what, to whom, when*, or *where*, regarding a target predicate. For instance, PropBank's ARG0 for the predicate **say** answers the question "*who* **said** *something?*".

QA-SRL (He et al., 2015) suggests that answering explicit role questions is an intuitive means to solicit predicate-argument structures from non-expert annotators. Annotators are presented with a sentence in which a target predicate[4] was marked, and are requested to annotate argument role questions, phrased using a restricted grammar, and corresponding answers.

For example, given the sentence "*President Obama who flew to Russia called the vice president*" and the target predicate **called**, an annotator can intuitively provide the following QA pairs: (1) *Who called?* **President Obama** and (2) *Whom did someone call?* **the vice president**.

In order to assess the validity of their annotation scheme, He et al. annotated a large sample of the PropBank corpus (1241 sentences) with QA-SRL, and showed high agreement with PropBank over this sample. In the following sections we make use of these explicit role questions for annotating non-restrictive modifiers.

3 Annotation Methodology

As mentioned in the Introduction, the first goal of this work is to assemble a large and consistent corpus, annotated with non-restrictive modifications. In this section, we present a crowdsourcing methodology which allows us to generate such corpus in a cost-effective manner (Section 3.2). As a preliminary step, we conducted a smaller scale

[4]Currently consisting of automatically annotated verbs.

expert annotation (Section 3.1), which will serve as a gold standard with which to test the crowsdsourced annotations.

3.1 Expert Annotation

Two researchers, with linguistics and NLP education, were presented with a sample of 219 modifiers of NPs in 100 sentences,[5] and were asked to annotate each modifier as either restrictive or non-restrictive, according to the linguistic definition presented in Section 2. Prior to annotating the expert dataset, the annotators discussed the process and resolved conflicts on a development set of 20 modifiers.

The annotators agreement was found to be high, reaching agreement on 93.5% of the instances, and κ of 84.2% . An analysis of the few disagreements found that the deviations between the annotators stem from semantic ambiguities, where two legitimate readings of the sentence led to disagreeing annotations. For example, in "*sympathetic fans have* **sent** *Ms. Shere copies of her recipes clipped from magazines over the years*", one annotator read the underlined modifier clause as restrictive, identifying particular recipes, while the second annotator read the modifier as non-restrictive, adding supplementary information on the sent recipes.

Finally, we compose the expert annotation dataset from the 207 modifiers agreed upon by both annotators. In the next section we use this dataset to evaluate the quality of our crowdsourced annotations.

3.2 Crowdsourcing Annotation Process

In our scheme, each annotation instance assigns a binary label (restrictive or non-restrictive) to a 4-tuple (s, v, p, m) – where m is a modifier of the noun phrase p, which is an argument of a verbal predicate v, in a sentence s. We incorporate v in our scheme in order to provide non-trained annotators with an argument role question (discussed in 2.3), as elaborated below.[6]

Consider, for example, the sentence s – "*the speaker* **thanked** *[President Obama who just entered the room]*". We want to annotate the restrictiveness value of the relative clause m (underlined), which modifies the matrix noun phrase p (bracketed), which is in turn an argument of a governing predicate v (in bold).

Our annotation procedure does not require the annotator to be familiar with the formal linguistic definition of restrictiveness. Instead, we use binary question-answering (true / false questions) as an intuitive formulation of non-restrictive modification. We present annotators with the argument role question pertaining to the argument NP, and ask whether this NP *without* the modifier gives the same answer to the argument role question as the original NP did.

In our example, an annotator is presented with the argument role question "*whom did someone thank?*" (which is answered by p), and is asked to decide whether the reduced NP, "*President Obama*", provides the same answer to the question as the full NP does. If the answer is positive (as in this case), we consider the modifier to be *non-restrictive*, otherwise we consider it to be *restrictive*.

As an example for the restrictive case, consider "*she* **wore** *[the necklace that her mother gave her]*", and the respective argument role-question "*what did someone wear?*". In this case, as opposed to the previous example, the reduced NP ("*the necklace*") does not refer to the same entity as the original NP, since we lose the specific identity of the necklace which was worn.

The intuition for this process arises from the linguistic definition for modifier restrictiveness. Namely, a restrictive modifier is defined as an integral part of the NP, and a non-restrictive modifier as providing supplementary or additional information about it. Therefore, in the restrictive case, omitting the modifier would necessarily change the meaning of the answer, while in the non-restrictive case, omitting it would not change the entity referenced by the full NP, and would therefore provide the same answer to the argument role question.

4 Corpus

In this section we describe the creation of a consistent human-annotated restrictiveness corpus, using the annotation process described in the previous section. We show this corpus to be of high quality by comparing it with the independent expert annotation. In Section 5 we use this corpus to train and test several automatic classifiers.

[5] These were taken at random from the development partition of the corpus described in Section 4.

[6] Our annotation currently covers the most common case of NPs which serve as arguments of verbal predicates.

Modifier Type	Identified By	#	Non-Restrictive	Agreement	
				κ	%
Adjectival	*pos* = JJ	684	41.36%	74.70	87.36
Prepositional	*pos* = IN / TMP / LOC	693	36.22%	61.65	85.10
Appositive	*rel* = APPO / PRN	342	73.68%	60.29	80.00
Non-Finite	*rel* = TO	279	68.82%	71.04	86.48
Verbal	*pos* = VB and not relative clause	150	69.33%	100	100
Relative clause	*pos* = VB and child *pos* = WP	43	79.07%	100	100
Total	**-**	**2191**	**51.12%**	**73.79**	**87.00**

Table 1: Corpus statistics by modifier types, which were identified by part of speech (*pos*) and dependency label (*rel*) (Section 4.1). The number of instances (#) and non-restrictiveness percentage refer to the full crowdsourced annotation. Agreement (Cohen's κ and percent of matching instances) is reported for the expert-annotated data (Section 4.2), between the expert and crowdsourced annotations.

4.1 Data Collection

We use the dataset which He et al. (2015) annotated with Question-Answer pairs (discussed in Section 2.3), and keep their train / dev / test split into 744 / 249 / 248 sentences, respectively. This conveniently allows us to link between argument NPs and their corresponding argument role question needed for our annotation process, as described in previous section.

This dataset is composed of 1241 sentences from the CoNLL 2009 English dataset (Hajič et al., 2009), which consists of newswire text annotated by the Penn TreeBank (Marcus et al., 1993), PropBank (Palmer et al., 2005), and NomBank (Meyers et al., 2004), and converted into dependency grammar by (Johansson and Nugues, 2008).

As mentioned in Section 3.2, each of our annotation instances is composed of a sentence s, a verbal predicate v, a noun phrase p, and a modifier m. We extract each such possible tuple from the set of sentences in the following automatic manner:

1. Identify a verb v in the gold dependency tree.
2. Follow its outgoing dependency arcs to a noun phrase argument p (a dependent of v with a nominal part of speech).
3. Find m, a modifying clause of p which might be non-restrictive, according to the rules described in Table 1, under the "Identified By" column. This filters out modifiers which are always restrictive, such as determiners or genitives, following (Huddleston et al., 2002), as discussed in Section 2. Notice that this automatic modifier detection decouples the span annotation from the restrictiveness annotation, which was a source for inconsistencies in Dornescu et al's annotation (Section 2.2).

This automatic process yields a dataset of 2191 modifiers of 1930 NPs in 1241 sentences. We note that our collection process ensures that the corpus correlates with the syntactic dependency annotation of the CoNLL 2009 shared task, and can therefore be seen as an augmentation of its modifier labels to include restrictiveness annotations.

In order to find the corresponding argument role question, we follow the process carried by He et al.; An argument NP is matched to an annotated Question-Answer pair if the NP head is within the annotated answer span. Following this matching process yields a match for 1840 of the NPs.

For the remaining 90 NPs we manually compose an argument role question by looking at the governing predicate and its argument NP. For example, given the sentence "*[The son of an immigrant stonemason of Slovenian descent] was* **raised** *in a small borough outside Ebensburg*", the predicate **raised** and the bracketed NP argument, we produce the argument role question "*Who was raised?*".

The corpus category distribution is depicted in Table 1, under column labeled "#". In later sections we report agreement and performance across these categories to produce finer grained analyses.

4.2 Crowdsourcing Annotation

We use Amazon Mechanical Turk[7] to annotate the 2191 modifiers for restrictiveness, according to the

[7] https://www.mturk.com

process defined in Section 3.2. Each modifier was given to 5 annotators, and the final tag was assigned by majority vote. We used the development set to refine the guidelines, task presentation, and the number of annotators.

Each annotator was paid 5c for the annotation of an NP, which in average provided 1.16 modifiers. This sets the average price for obtaining a single modifier annotation at $5 \cdot \frac{5}{1.16} = 21.5$c.

The agreement with the 217 NP modifiers annotated by the experts (Section 3.1) and percentage of positive (non-restrictive) examples per category can be found in Table 1, in the columns labeled "agreement". The labels are generally balanced, with 51.12% non-restrictive modifiers in the entire dataset (varying between 36.22% for prepositional modifiers and 79.07% for relative clauses).

Overall, the crowdsourced annotation reached good agreement levels with our expert annotation, achieving 73.79 κ score (substantial agreement). The lowest agreement levels were found on prepositional and appositive modifiers (61.65% and 60.29%).[8] Indeed, as discussed in Section 2, these are often subtle decisions which rely heavily on context. For example, the following instances were disagreed upon between our expert annotation and the crowdsourced annotation: In "*[Charles LaBella , the assistant U.S. attorney prosecuting the Marcos case], did n't* **return** *phone calls seeking comment*" (an appositive example), the experts annotated the underlined modifier as non-restrictive, while the crowdsource annotation marked it as restrictive. Inversely, in "*The amendment* **prompted** *[an ironic protest] from Mr. Thurmond*", the experts annotated the adjectival modifier as restrictive, while the crowdsource annotation tagged it as non-restrictive.

5 Predicting Non-Restrictive Modification

In this section we present an automatic system which: (1) Identifies NP modifiers in a dependency parser's output (as shown in Table 1, column "Identified By") and (2) Uses a CRF model to classify each modifier as either restrictive or non-restrictive, based on the features listed in Table 2, and elaborated below.[9]

5.1 Baselines

We begin by replicating the algorithms in the two prior works discussed in Section 2.2. This allows us to test their performance consistently against our new human annotated dataset.

Replicating (Honnibal et al., 2010) They annotated a modifier as restrictive if and only if it was preceded with a comma. We re-implement this baseline and classify all of the modifiers in the test set according to this simple property.

Replicating (Dornescu et al., 2014) Their best performing ML-based algorithm[10] uses the supervised CRFsuite classifier (Okazaki, 2007) over "standard features used in chunking, such as word form, lemma and part of speech tags". Replicating their baseline, we extract the list of features detailed in Table 2 (in the row labeled "chunking features").

5.2 Our Classifier

In addition to Dornescu et al.'s generic chunking framework, we also extract features which were identified in the linguistic literature as indicative for non-restrictive modifications. These features are then used in the CRFsuite classifier (the same CRF classifier used by Donescu et al.) to make the binary decision. The following paragraphs elaborate on the motivation for each of the features.

Enclosing commas We extend Honnibal's et al.'s classification method as a binary feature which marks whether the clause is both preceded and terminated with a comma. This follows from a well-known writing style and grammar rule which indicates that non-restrictive clausal modifiers should be enclosed with a comma.

Governing relative In the linguistic literature, it was posited that the word introducing a clausal modifier (termed *relative*) is an indication for the restrictiveness of the subordinate clause. For example, Huddleston. et al. (2002) [p. 1059] analyzes the word "that" as generally introducing a restrictive modifier, while a wh-pronoun is

[8] While linguistic literature generally regards appositives as non-restrictive, some of the appositions marked in the dependency conversion are in fact misclassified coordinations, which explains why some of them were marked as restrictive.

[9] We use our new crowdsourced corpus to train our model as well as the baseline model.

[10] They also implement a rule-based method, named DAPR, which, when combined with the described ML approach surpasses their ML algorithm by ∼1.5% increase in F1. We could not find a publicly available implementation of this method.

more likely to introduce non-restrictive modification. We therefore extract features of the word which governs the relative, such as the surface form, its lemma, POS tag, and more. The full list is shown under "Governing relative" in Table 2.

Named entities As illustrated throughout the paper, modifiers of named entities tend to be non-restrictive. We run the Stanford Named Entity Recognizer (NER) (Finkel et al., 2005) and introduce a feature indicating the type of named entity (PERSON, ORG or LOC), where applicable.

Lexical word embeddings We include the pre-trained word embeddings of the modifier's head word, calculated by (Mikolov et al., 2013). These distributional features help the classifier associate between similar words (for example, if "good" is non-restrictive in some contexts, it is likely that "fine" is also non-restrictive within similar contexts).

Modifier type We add the automatically identified modifier type as a feature, to associate certain features as indicative for certain types of modifiers (e.g., enclosing commas might be good indicators for relative clause, while word embeddings can be specifically helpful for adjectival modifiers).

System	Feature Type	Description
Honnibal et al.	Preceding comma	w[-1] == ,
This paper	Chunking features (Dornescu et al.)	feats[head-1] feats[head] feats[head+1]
	Enclosing commas	true iff the clause is preceded and terminated with commas
	Governing relative	feats[parent-1] feats[parent] feats[parent+1]
	Prepositions	feats[pobj-1] feats[pobj] feats[pobj+1]
	NER	PERSON, ORGANIZATION, LOCATION
	Lexical word embeddings	Mikolov et al's 300-dimensional continuous word embeddings
	Modifier type	one of the types described in Table 1

Table 2: Features used for classification in each of the systems as described in Section 5. *head* - head of the modifier in the dependency tree. *parent* - parent of *head* in the dependency tree. *pobj* - object of the preposition, in case of prepositional *head*. *feats[i]* refers to extracting the following features from the word i: POS tag, lemma, is title, is all lower case, is all upper case, is beginning / end of sentence.

6 Evaluation

We use the QA-SRL test section (containing 412 NP modifiers) to evaluate each of the systems described in Section 5 on gold and predicted trees, both provided in the CoNLL 2009 dataset (the predicted dependency relations were obtained using MaltParser (Nivre et al., 2007)). The gold setting allows us to test the performance of the systems without accumulating parser errors. In addition, it allows us to partition and analyze our dataset according to the gold modifier type. The predicted setting, on the other hand, allows us to evaluate our classifier in a real-world application scenario, given automatic parsing output.

6.1 Gold Trees

The results for each of the systems across our categories on the gold trees are shown in Table 3. Note that we regard non-restrictive modification as positive examples, and restrictive modification as negative examples. This is in line with the applicative goal of reducing argument span by removing non-restrictive modifiers, discussed in the Introduction. Switching the labels does not significantly change the numbers, since the corpus is relatively well balanced between the two labels (as can be seen in Table 1). Following are several observations based on an error analysis of these results.

Prepositional and adjectival modifiers are harder to predict All systems had more difficulties in classifying both of these categories. This reiterates the relatively lower agreement for these categories between the crowdsource and expert annotation, discussed in Section 4.2.

For clausal modifiers, preceding commas are good in precision but poor for recall As can be seen in Honnibal et al.'s columns, a preceding comma is a good indication for a non-restrictive clausal modifier (all categories excluding adjectival or verbal modifiers), but classifying solely by its existence misses many of the non-restrictive instances.

Modifier Type	#	Precision			Recall			F1		
		Honnibal	Dornescu	Our	Honnibal	Dornescu	Our	Honnibal	Dornescu	Our
Prepositional	135	.83	.67	.69	.1	.16	.41	.18	.26	.51
Adjectival	111	.33	.38	.59	.06	.06	.21	.11	.11	.31
Appositive	78	.77	.81	.82	.34	.93	.98	.47	.87	.89
Non-Finite	55	.77	.63	.64	.29	.97	.97	.42	.76	.77
Verbal	20	0	.75	.75	0	1	1	0	.86	.86
Relative clause	13	1	.85	.85	.27	1	1	.43	.92	.92
Total	412	.72	.72	**.73**	.19	.58	**.68**	.3	.64	**.72**

Table 3: Test set performance of the 3 different systems described in Sections 5 and 6 on gold trees from the CoNLL 2009 dataset, across the different categories defined in Section 4.

Features		**P**	**R**	**F1**
All		.73	.68	.72
Baseline	- comma	.72	.68	.70
	- chunking	.72	.66	.69
New	- governing relative	.74	.61	.67
	- prepositions	.73	.67	.70
	- word embeddings	.72	.69	.71
	- NER	.71	.68	.70
	- mod type	.74	.66	.70

Table 4: Feature ablation tests on gold trees. Each row specifies a different feature set – "All" specifies the entire feature set from Table 2, while each subsequent line removes one type of features.

(Dornescu et al., 2014) performs better on our dataset Their method achieves much better results on our dataset (compare 64% overall F1 on our dataset with their reported 45.29% F1 on their dataset). This speaks both for their method as a valid signal for restrictiveness annotation, as well as for the improved consistency of our dataset.

Our system improves recall Overall, our system significantly outperforms both baselines by more than 8% gain in F1 score. Specifically, the numbers show clearly that we improve recall in the frequent categories of prepositional and adjectival modifiers. Furthermore, the results of an ablation test on our features (shown in Table 4) show that chunking and governing relative features provide the highest individual impact.

6.2 Predicted Trees

To test our classifier in a realistic setting we evaluate its performance on predicted dependency trees. To obtain the candidate modifiers, we use the same extractor presented in previous sections, applied on the predicted trees in the test section of the CoNLL 2009 dataset. We then apply the models trained on the gold train set of the same corpus.

System	**P**	**R**	**F1**
Candidate Extraction	.91	.93	.92
Honnibal	.71	.18	.29
Dornescu	.68	.53	.59
Our	**.69**	**.63**	**.66**

Table 5: Restrictiveness results (bottom three lines) on predicted trees. The top line (Candidate Extraction) measures the percent of correct modifiers identified in the predicted trees (shared across all of the classifiers). See Section 6.2.

For evaluation, we use the gold labels and compute (1) precision – the percent of predicted non-restrictive modifiers which match a gold non-restrictive modifier, and (2) recall – the percent of gold non-restrictive modifiers which match a predicted non-restrictive modifier. Note that this metric is strict, conflating both parser errors with our classifier's errors. The results are shown in Table 5.

The first line in the table measures the performance of the modifier extractor module on the predicted trees. A predicted modifier is considered correct if it agrees with a gold modifier on both its syntactic head as well as its span. The modifier extractor module is shared across all classifiers, as discussed in Section 5, and its performance on the predicted trees imposes an upper bound on all the classifiers.

Both our and Dornescu's classifiers drop 5-6 points in F1, keeping the differences observed on the gold trees, while Honnibal et al.'s simple comma-based classifier is less sensitive to parser errors, dropping only one point in F1.

This small drop stems from our classifiers largely relying only on the modifier head and its span for feature computation, generally ignoring

parsing errors within the modifier subtree.

7 Conclusions and Future Work

We presented an end-to-end framework for restrictiveness annotation, including a novel QA-SRL based crowdsourcing methodology and a first consistent human-annotated corpus. Furthermore, we presented a linguistically motivated classifier, surpassing the previous baseline by 8% gain in F1.

Future work can use our annotated corpus to develop classifiers that deal better with prepositional and adjectival modifiers, which require deeper semantic analysis.

Acknowledgments

We would like to thank Michael Elhadad and Yoav Goldberg for fruitful discussions, and the anonymous reviewers for their helpful comments.

This work was supported in part by grants from the MAGNET program of the Israeli Office of the Chief Scientist (OCS), the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Gabor Angeli, Melvin Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of ACL*, pages 86–90. Association for Computational Linguistics.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proceedings of CONLL*, pages 152–164.

Marie-Catherine de Marneffe and Christopher D Manning. 2008. The stanford typed dependencies representation. In *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Iustin Dornescu, Richard Evans, and Constantin Orasan. 2014. Relative clause extraction for syntactic simplification. *COLING 2014*, page 1.

Nigel Fabb. 1990. The difference between english restrictive and nonrestrictive relative clauses. *Journal of linguistics*, 26(01):57–77.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*, pages 340–348. Association for Computational Linguistics.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18. Association for Computational Linguistics.

Luheng He, Mike Lewis, and Luke Zettlemoyer. 2015. Question-answer driven semantic role labeling: Using natural language to annotate natural language. In *the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. In *Computational Linguistics*.

Matthew Honnibal, James R Curran, and Johan Bos. 2010. Rebanking ccgbank for improved np interpretation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 207–215. Association for Computational Linguistics.

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

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic-semantic analysis with propbank and nombank. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning*, pages 183–187. Association for Computational Linguistics.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

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

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. The nombank project: An interim report. In *HLT-NAACL 2004 workshop: Frontiers in corpus annotation*, pages 24–31.

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

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

Naoaki Okazaki. 2007. Crfsuite: a fast implementation of conditional random fields (crfs).

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational linguistics*, 31(1):71–106.

Gabriel Stanovsky, Ido Dagan, and Mausam. 2015. Open IE as an intermediate structure for semantic tasks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL 2015)*.

Carla Umbach. 2006. Non-restrictive modification and backgrounding. In *Proceedings of the Ninth Symposium on Logic and Language*, pages 152–159.

Bilingual Segmented Topic Model

Akihiro Tamura and **Eiichiro Sumita**
National Institute of Information and Communications Technology
3-5 Hikaridai, Seika-cho, Soraku-gun, Kyoto, JAPAN
{akihiro.tamura, eiichiro.sumita}@nict.go.jp

Abstract

This study proposes the bilingual segmented topic model (BiSTM), which hierarchically models documents by treating each document as a set of segments, e.g., sections. While previous bilingual topic models, such as bilingual latent Dirichlet allocation (BiLDA) (Mimno et al., 2009; Ni et al., 2009), consider only cross-lingual alignments between entire documents, the proposed model considers cross-lingual alignments between segments in addition to document-level alignments and assigns the same topic distribution to aligned segments. This study also presents a method for simultaneously inferring latent topics and segmentation boundaries, incorporating unsupervised topic segmentation (Du et al., 2013) into BiSTM. Experimental results show that the proposed model significantly outperforms BiLDA in terms of perplexity and demonstrates improved performance in translation pair extraction (up to +0.083 extraction accuracy).

1 Introduction

Probabilistic topic models, such as probabilistic latent semantic analysis (PLSA) (Hofmann, 1999) and latent Dirichlet allocation (LDA) (Blei et al., 2003), are generative models for documents that have been used as unsupervised frameworks to discover latent topics in document collections without prior knowledge. These topic models were originally applied to monolingual data; however, various recent studies have proposed the use of probabilistic topic models in multilingual settings[1], where latent topics are shared across multiple languages. These models have improved several multilingual tasks, such as translation pair extraction and cross-lingual text classification (see the survey paper by Vulić et al. (2015) for details).

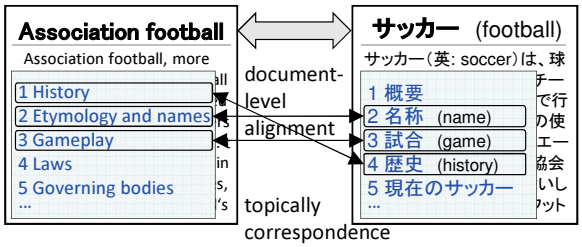


Figure 1: Wikipedia Article Example

Most multilingual topic models, including bilingual LDA (BiLDA) (Mimno et al., 2009; Ni et al., 2009), model a document-aligned comparable corpus, such as a collection of Wikipedia articles, where aligned documents are topically similar but are not direct translations[2]. In particular, these models assume that the documents in each tuple share the same topic distribution and that each cross-lingual topic has a language-specific word distribution.

Existing multilingual topic models consider only document-level alignments. However, most documents are hierarchically structured, i.e., a document comprises segments (e.g., sections and paragraphs) that can be aligned across languages. Figure 1 shows a Wikipedia article example, which contains a set of sections. Sections 1, 2, and 3 in the English article correspond topically to sections 4, 2, and 3 in the Japanese counterpart, re-

[1] In this work, we deal with a bilingual setting, but our approach can be extended straightforwardly to apply to more than two languages.

[2] In this study, we focus on models for a document-aligned comparable corpus. We describe other types of multilingual topic models and their limitations in Section 7.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1266–1276,
Berlin, Germany, August 7-12, 2016.

spectively. To date, such segment-level alignments have been ignored; however, we consider that such corresponding segments must share the same topic distribution.

Du et al. (2010) have shown that segment-level topics and their dependencies can improve modeling accuracy in a monolingual setting. Based on that research, we expect that segment-level topics can also be useful for modeling multilingual data.

This study proposes a bilingual segmented topic model (BiSTM) that extends BiLDA to capture segment-level alignments through a hierarchical structure. In particular, BiSTM considers each document as a set of segments and models a document as a document-segment-word structure. The topic distribution of each segment (per-segment topic distribution) is generated using a Pitman–Yor process (PYP) (Pitman and Yor, 1997), in which the base measure is the topic distribution of the related document (per-document topic distribution). In addition, BiSTM introduces a binary variable that indicates whether two segments in different languages are aligned. If two segments are aligned, their per-segment topic distributions are shared; if they are not aligned, they are independently generated.

BiSTM leverages existing segments from a given segmentation. However, a segmentation is not always given, and a given segmentation might not be optimal for statistical modeling. Therefore, this study also presents a model, BiSTM+TS, that incorporates unsupervised topic segmentation into BiSTM. BiSTM+TS integrates point-wise boundary sampling into BiSTM in a manner similar to that proposed by Du et al. (2013) and infers segmentation boundaries and latent topics jointly.

Experiments using an English–Japanese and English–French Wikipedia corpus show that the proposed models (BiSTM and BiSTM+TS) significantly outperform the standard bilingual topic model (BiLDA) in terms of perplexity, and that they improve performance in translation extraction (up to +0.083 top 1 accuracy). The experiments also reveal that BiSTM+TS is comparable to BiSTM, which uses manually provided segmentation, i.e., section boundaries in Wikipedia articles.

2 Bilingual LDA

This section describes the BiLDA model (Mimno et al., 2009; Ni et al., 2009), which we take as our baseline. BiLDA is a bilingual extension of basic monolingual LDA (Blei et al., 2003) for a document-aligned comparable corpus. While monolingual LDA assumes that each document has its own topic distribution, BiLDA assumes that aligned documents share the same topic distribution and discovers latent cross-lingual topics.

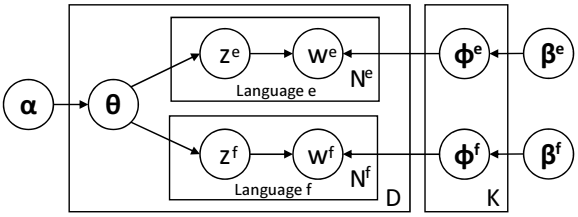


Figure 2: Graphical Model of BiLDA

Algorithm 1 Generative Process of BiLDA

1: **for** each topic $k \in \{1, ..., K\}$ **do**
2: **for** each language $l \in \{e, f\}$ **do**
3: choose $\boldsymbol{\phi}^{\boldsymbol{l}}_{\boldsymbol{k}} \sim$ Dirichlet($\boldsymbol{\beta}^{\boldsymbol{l}}$)
4: **end for**
5: **end for**
6: **for** each document pair d_i ($i \in \{1, ..., D\}$) **do**
7: choose $\boldsymbol{\theta}_{\boldsymbol{i}} \sim$ Dirichlet($\boldsymbol{\alpha}$)
8: **for** each language $l \in \{e, f\}$ **do**
9: **for** each word w^l_{im} ($m \in \{1, ..., N^l_i\}$) **do**
10: choose $z^l_{im} \sim$ Multinomial($\boldsymbol{\theta}_{\boldsymbol{i}}$)
11: choose $w^l_{im} \sim p(w^l_{im}|z^l_{im}, \boldsymbol{\phi}^{\boldsymbol{l}})$
12: **end for**
13: **end for**
14: **end for**

Algorithm 1 and Figure 2 show the generative process and graphical model, respectively, of BiLDA. BiLDA models a document-aligned comparable corpus, i.e., a set of D document pairs in two languages, e and f. Each document pair d_i ($i \in \{1, ..., D\}$) comprises aligned documents in the language e and f: d_i=(d^e_i, d^f_i). BiLDA assumes that each topic $k \in \{1, ..., K\}$ comprises the set of a discrete distribution over words for each language. Each language-specific per-topic word distribution $\boldsymbol{\phi}^{\boldsymbol{l}}_{\boldsymbol{k}}$ ($l \in \{e, f\}$) is drawn from a Dirichlet distribution with the prior $\boldsymbol{\beta}^{\boldsymbol{l}}$ (Steps 1-5). To generate a document pair d_i, the per-document topic distribution $\boldsymbol{\theta}_{\boldsymbol{i}}$ is first drawn from a Dirichlet distribution with the prior $\boldsymbol{\alpha}$ (Step 7). Thus, aligned documents d^e_i and d^f_i share the same topic distribution. Then, for each word at $m \in \{1, ..., N^l_i\}$ in document d^l_i in language l, a latent topic assignment z^l_{im} is drawn from a multinomial

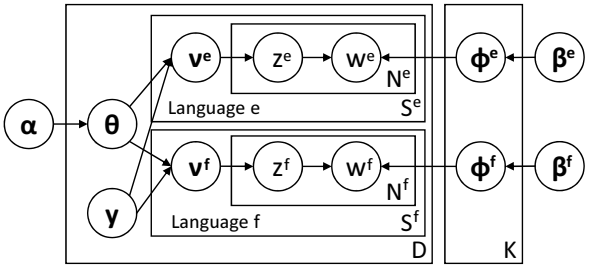


Figure 3: Graphical Model of BiSTM

Algorithm 2 Generative Process of BiSTM

1: **for** each topic $k \in \{1, ..., K\}$ **do**
2: **for** each language $l \in \{e, f\}$ **do**
3: choose $\boldsymbol{\phi_k^l} \sim$ Dirichlet($\boldsymbol{\beta^l}$)
4: **end for**
5: **end for**
6: **for** each document pair d_i ($i \in \{1, ..., D\}$) **do**
7: choose $\boldsymbol{\theta_i} \sim$ Dirichlet($\boldsymbol{\alpha}$)
8: **if** $\boldsymbol{y_i}$ are not given **then**
9: choose $\gamma_i \sim$ Beta(η_0, η_1)
10: choose $\boldsymbol{y_i} \sim$ Bernoulli(γ_i)
11: **end if**
12: generate aligned segment sets $\boldsymbol{AS_i}$ = genAS($\boldsymbol{y_i}$)
13: **for** each set $\boldsymbol{AS_{ig}}$ ($g \in \{1, ..., |\boldsymbol{AS_i}|\}$) **do**
14: choose $\boldsymbol{\nu_{ig}} \sim$ PYP($a, b, \boldsymbol{\theta_i}$)
15: **end for**
16: **for** each language $l \in \{e, f\}$ **do**
17: **for** each segment s_{ij}^l ($j \in \{1, ..., S_i^l\}$) **do**
18: get index of s_{ij}^l in $\boldsymbol{AS_i}$: g =get_idx($\boldsymbol{AS_i}$,s_{ij}^l)
19: **for** each word w_{ijm}^l ($m \in \{1, ..., N_{ij}^l\}$) **do**
20: choose $z_{ijm}^l \sim$ Multinomial($\boldsymbol{\nu_{ig}}$)
21: choose $w_{ijm}^l \sim p(w_{ijm}^l | z_{ijm}^l, \boldsymbol{\phi^l})$
22: **end for**
23: **end for**
24: **end for**
25: **end for**

distribution with the prior $\boldsymbol{\theta_i}$ (Step 10). Later, a word w_{im}^l is drawn from a probability distribution $p(w_{im}^l | z_{im}^l, \boldsymbol{\phi^l})$ given the topic z_{im}^l (Step 11).

3 Bilingual Segmented Topic Model

Here, we describe BiSTM, which extends BiLDA to capture segment-level alignments. Algorithm 2 and Figure 3 show the generative process and graphical model, respectively, of BiSTM. As can be seen in Figure 3, BiSTM introduces a segment-level layer between the document- and word-level layers in both languages. In other words, per-segment topic distributions for each language, $\boldsymbol{\nu^e}$ and $\boldsymbol{\nu^f}$, are introduced between per-document topic distributions $\boldsymbol{\theta}$ and topic assignments for words, $\boldsymbol{z^e}$ and $\boldsymbol{z^f}$. In addition, BiSTM incorporates binary variables $\boldsymbol{y}$ to represent segment-level alignments.

Each document d_i^l in a pair of aligned documents d_i is divided into S_i^l segments: $d_i^l = \bigcup_{j=1}^{S_i^l} s_{ij}^l$. BiSTM makes the same assumption for per-topic word distributions as BiLDA, i.e., $\boldsymbol{\phi_k^l}$ are language-specific and drawn from Dirichlet distributions (Steps 1-5).

In the generative process for a document pair d_i, the per-document topic distribution $\boldsymbol{\theta_i}$ is first drawn in the same way as in BiLDA (Step 7). Thus, in BiSTM, each document pair shares the same topic distribution.

Then, if segment-level alignments are not given, $\boldsymbol{y_i}$ are generated (Steps 8-11). We assume that each document pair d_i has a probability γ_i that indicates comparability between segments across languages. γ_i is drawn from a Beta distribution with the priors η_0 and η_1 (Step 9). Then, each of $\boldsymbol{y_i}$ is drawn from a Bernoulli distribution with the prior γ_i (Step 10). Here, $y_{ijj'}$ = 1 if and only if s_{ij}^e and $s_{ij'}^f$ are aligned; otherwise, $y_{ijj'}$ = 0. Note that if segment-level alignments are observed, then Steps 8-11 are skipped. Later, a set of aligned segment sets $\boldsymbol{AS_i}$ is generated based on $\boldsymbol{y_i}$ (Step 12). For example, given $d_i^e = \{s_{i1}^e, s_{i2}^e\}, d_i^f = \{s_{i1}^f, s_{i2}^f, s_{i3}^f\}$, y_{i11} and y_{i12} are 1, and the other y's are 0, $\boldsymbol{AS_i} = \{\boldsymbol{AS_{i1}} = \{s_{i1}^e, s_{i1}^f, s_{i2}^f\}, \boldsymbol{AS_{i2}} = \{s_{i2}^e\}, \boldsymbol{AS_{i3}} = \{s_{i3}^f\}\}$ is generated in Step 12. Then, for each aligned segment set $\boldsymbol{AS_{ig}}$ ($g \in \{1, ..., |\boldsymbol{AS_i}|\}$), the per-segment topic distribution $\boldsymbol{\nu_{ig}}$ is obtained from a Pitman–Yor process with the base measure $\boldsymbol{\theta_i}$, the concentration parameter a, and the discount parameter b (Step 14). Through Steps 12-15, aligned segments indicated by $\boldsymbol{y}$ share the same per-segment topic distribution. For instance, s_{i1}^e, s_{i1}^f, and s_{i2}^f have the same topic distribution $\boldsymbol{\nu_{i1}} \sim$ PYP(a, b, $\boldsymbol{\theta_i}$) in the above example.

Then, for each word at $m \in \{1, ..., N_{ij}^l\}$ in segment s_{ij}^l in document d_i^l in language l, a latent topic assignment z_{ijm}^l is drawn from a multinomial distribution with the prior $\boldsymbol{\nu_{ig}}$ (Step 20), where g denotes the index of the element set of $\boldsymbol{AS_i}$ that includes the segment s_{ij}^l, e.g., g for s_{i2}^f is 1. Subsequently, a word w_{ijm}^l is drawn based on the assigned topic z_{ijm}^l and the language-specific per-topic word distribution $\boldsymbol{\phi^l}$ in the same manner as in BiLDA (Step 21).

t_{igk}	Table count of topic k in the CRP for aligned segment set g in document pair i.
$\boldsymbol{t_{ig}}$	K-dimensional vector, where k-th value is t_{igk}.
$t_{ig\cdot}$	Total table count in aligned segment set g in document pair i, i.e., $\sum_k t_{igk}$.
n_{igk}	Total number of words with topic k in aligned segment set g in document pair i.
$n_{ig\cdot}$	Total number of words in aligned segment set g in document pair i, i.e., $\sum_k n_{igk}$.
M^l_{kw}	Total number of word w with topic k in language l.
$\boldsymbol{M^l_k}$	$\lvert\boldsymbol{W^l}\rvert$-dimensional vector, where w-th value is M^l_{kw}.

Table 1: Statistics used in our Inference

3.1 Inference for BiSTM

In inference, we find the set of latent variables $\boldsymbol{\theta}$, $\boldsymbol{\nu}$, $\boldsymbol{z}$, and $\boldsymbol{\phi}$ that maximizes their posterior probability given the model parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and observations $\boldsymbol{w}$, $\boldsymbol{y}$, i.e., $p(\boldsymbol{\theta}, \boldsymbol{\nu}, \boldsymbol{z}, \boldsymbol{\phi} | \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{w}, \boldsymbol{y})$. Here, a language-dependent variable without a superscript denotes both of the variable in language e and that in f, e.g., $\boldsymbol{z} = \{\boldsymbol{z^e}, \boldsymbol{z^f}\}$. Unfortunately, as in other probabilistic topic models, such as LDA and BiLDA, we cannot compute this posterior using an exact inference method. This section presents an approximation method for BiSTM based on blocked Gibbs sampling, inspired by Du et al. (2013).

In our inference, the hierarchy in BiSTM, i.e., the generation of $\boldsymbol{\nu}$ and $\boldsymbol{z}$, is explained by the Chinese restaurant process (CRP), through which the parameters $\boldsymbol{\theta}$, $\boldsymbol{\nu}$, and $\boldsymbol{\phi}$ are integrated out, and the statistics on table counts in the CRP, $\boldsymbol{t}$, are introduced. Table 1 lists all statistics used in our inference, where $\boldsymbol{W^l}$ denotes a vocabulary set in language l. Moreover, to accelerate convergence, we introduce an auxiliary binary variable δ^l_{ijm} for w^l_{ijm}, indicating whether w^l_{ijm} is the first customer on a table ($\delta^l_{ijm} = 1$) or not ($\delta^l_{ijm} = 0$), and t_{igk} is computed based on $\boldsymbol{\delta}$ in the same manner as in Chen et al. (2011): $t_{igk} = \sum_{s^l_{ij} \in \boldsymbol{AS_{ig}}} \sum_{m=1}^{N^l_{ij}} \delta^l_{ijm} I(z^l_{ijm} = k)$, where $I(x)$ is a function that returns 1 if the condition x is true and 0 otherwise.

Our inference groups z^l_{ijm} and δ^l_{ijm} (each group is called a "block") and jointly samples them. Moreover, if $\boldsymbol{y}$ is not observed, our inference alternates two different kinds of blocks, $(z^l_{ijm}, \delta^l_{ijm})$ and $y_{ijj'}$. In each sampling, individual variables are resampled, conditioned on all other variables. In the following, we describe each sampling stage.

Sampling $(\boldsymbol{z}, \boldsymbol{\delta})$:

The joint posterior distribution of $\boldsymbol{z}$, $\boldsymbol{w}$, and $\boldsymbol{\delta}$ is induced in a manner similar to that in Du et al. (2010; 2013): $p(\boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\delta} | \boldsymbol{\alpha}, \boldsymbol{\beta}, a, b, \boldsymbol{y})$

$$= \prod_{i=1}^{D} \left(\frac{\mathrm{Beta}_K(\boldsymbol{\alpha} + \sum_{\boldsymbol{AS_i}} \boldsymbol{t_{ig}})}{\mathrm{Beta}_K(\boldsymbol{\alpha})} \prod_{\boldsymbol{AS_i}} \left(\frac{(b|a)_{t_{ig\cdot}}}{(b)_{n_{ig\cdot}}} \prod_{k=1}^{K} S\left(n_{igk}, t_{igk}, a\right) \binom{n_{igk}}{t_{igk}}^{-1} \right) \right)$$

$$\prod_{k=1}^{K} \left(\frac{\mathrm{Beta}_{W^e}(\boldsymbol{\beta^e} + \boldsymbol{M^e_k})}{\mathrm{Beta}_{W^e}(\boldsymbol{\beta^e})} \frac{\mathrm{Beta}_{W^f}(\boldsymbol{\beta^f} + \boldsymbol{M^f_k})}{\mathrm{Beta}_{W^f}(\boldsymbol{\beta^f})} \right),$$

where $\mathrm{Beta}_K(\cdot)$ and $\mathrm{Beta}_{W^l}(\cdot)$ are K- and $|\boldsymbol{W^l}|$-dimensional beta functions, respectively, $(b|a)_n$ is the Pochhammer symbol[3], and $(b)_n$ is given by $(b|1)_n$. $S(n, m, a)$ is a generalized Stirling number of the second kind (Hsu and Shiue, 1998), which is given by the linear recursion $S(n+1, m, a) = S(n, m-1, a) + (n - ma) S(n, m, a)$. To reduce computational cost, the Stirling numbers are preliminarily calculated in a logarithm format (Buntine and Hutter, 2012). Then, the cached values are used in our sampling.

The joint conditional distributions of z^l_{ijm} and δ^l_{ijm} are obtained from the above joint distribution using Bayes' rule: $p(z^l_{ijm} = k, \delta^l_{ijm} = 1 | \boldsymbol{z}^{-z^l_{ijm}}, \boldsymbol{w}, \boldsymbol{\delta}^{-\delta^l_{ijm}}, \boldsymbol{\alpha}, \boldsymbol{\beta}, a, b, \boldsymbol{y})$

$$= \frac{\beta^l_{w^l_{ijm}} + M^l_{kw^l_{ijm}}}{\sum_{w \in \boldsymbol{W^l}} (\beta^l_w + M^l_{kw})} \frac{\alpha_k + \sum_{\boldsymbol{AS_i}} t_{igk}}{\sum_{k=1}^{K} (\alpha_k + \sum_{\boldsymbol{AS_i}} t_{igk})} \frac{b + a t_{ig'\cdot}}{b + n_{ig'\cdot}} \frac{S(n_{ig'k} + 1, t_{ig'k} + 1, a)}{S(n_{ig'k}, t_{ig'k}, a)} \frac{t_{ig'k} + 1}{n_{ig'k} + 1},$$

$p(z^l_{ijm} = k, \delta^l_{ijm} = 0 | \boldsymbol{z}^{-z^l_{ijm}}, \boldsymbol{w}, \boldsymbol{\delta}^{-\delta^l_{ijm}}, \boldsymbol{\alpha}, \boldsymbol{\beta}, a, b, \boldsymbol{y})$

$$= \frac{\beta^l_{w^l_{ijm}} + M^l_{kw^l_{ijm}}}{\sum_{w \in \boldsymbol{W^l}} (\beta^l_w + M^l_{kw})} \frac{1}{b + n_{ig'\cdot}} \frac{S(n_{ig'k} + 1, t_{ig'k}, a)}{S(n_{ig'k}, t_{ig'k}, a)} \frac{n_{ig'k} + 1 - t_{ig'k}}{n_{ig'k} + 1},$$

where s^l_{ij} is included in $\boldsymbol{AS_{ig'}}$.

Sampling $\boldsymbol{y}$:

In our inference, each aligned segment set corresponds to a restaurant in the CRP. We regard the sampling of $y_{ijj'}$ as the choice of splitting or merging restaurant(s) in a manner similar to that

[3] $(b|a)_n = \prod_{t=0}^{n-1} (b + ta)$.

in the sampling of segmentation boundaries in Du et al. (2013). In particular, if $y_{ijj'} = 0$, then one aligned segment set $\boldsymbol{AS_m}$ is split into two aligned segment sets $\boldsymbol{AS_l}$ and $\boldsymbol{AS_r}$, where $\boldsymbol{AS_l}$, $\boldsymbol{AS_r}$, and $\boldsymbol{AS_m}$ include s^e_{ij}, $s^f_{ij'}$, and both, respectively. If $y_{ijj'} = 1$, then $\boldsymbol{AS_l}$ and $\boldsymbol{AS_r}$ are merged to $\boldsymbol{AS_m}$. For simplicity, our inference specifies $\boldsymbol{AS_l}$ and $\boldsymbol{AS_r}$ based on the current $\boldsymbol{y}$ as follows: if $AS_i(s^e_{ij}) = AS_i(s^f_{ij'})$, then $\boldsymbol{AS_l} = \{s^e_{ij}\} \cup AS^f_i(s^e_{ij}) \setminus \{s^f_{ij'}\}$ and $\boldsymbol{AS_r} = \{s^f_{ij'}\} \cup AS^e_i(s^f_{ij'}) \setminus \{s^e_{ij}\}$; otherwise, $\boldsymbol{AS_l} = AS_i(s^e_{ij})$ and $\boldsymbol{AS_r} = AS_i(s^f_{ij'})$. Here, $AS_i(j)$ is the element set of $\boldsymbol{AS_i}$ that includes the segment j, and $AS^l_i(j)$ is the set of segments in language l included in $AS_i(j)$. For example, in the example in Section 3, $AS_i(s^f_{i1}) = \boldsymbol{AS_{i1}} = \{s^e_{i1}, s^f_{i1}, s^f_{i2}\}$, $AS^e_i(s^f_{i1}) = \{s^e_{i1}\}$, and $AS^f_i(s^f_{i1}) = \{s^f_{i1}, s^f_{i2}\}$. In addition, if $y_{i11} = 0$, then $\boldsymbol{AS_m} = \{s^e_{i1}, s^f_{i1}, s^f_{i2}\}$ is split into $\boldsymbol{AS_l} = \{s^e_{i1}\} \cup AS^f_i(s^e_{i1}) \setminus \{s^f_{i1}\} = \{s^e_{i1}, s^f_{i2}\}$ and $\boldsymbol{AS_r} = \{s^f_{i1}\} \cup AS^e_i(s^f_{i1}) \setminus \{s^e_{i1}\} = \{s^f_{i1}\}$. If $y_{i23} = 1$, then $\boldsymbol{AS_l} = AS_i(s^e_{i2}) = \{s^e_{i2}\}$ and $\boldsymbol{AS_r} = AS_i(s^f_{i3}) = \{s^f_{i3}\}$ are merged to $\boldsymbol{AS_m} = \{s^e_{i2}, s^f_{i3}\}$.

The conditional distributions of $y_{ijj'}$ are as follows:

$$p(y_{ijj'} = 0|\boldsymbol{y}^{-y_{ijj'}}, \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{\delta}, \boldsymbol{\alpha}, a, b, \eta_0, \eta_1)$$
$$\propto \frac{\eta_0 + c_{i0}}{\eta_0 + \eta_1 + c_{i0} + c_{i1}} \mathrm{Beta}_K\left(\boldsymbol{\alpha} + \sum_{\boldsymbol{AS_i}} \boldsymbol{t_{ig}}\right)$$
$$\prod_{g \in \{\boldsymbol{AS_l}, \boldsymbol{AS_r}\}} \frac{(b|a)_{t_{ig\cdot}}}{(b)_{n_{ig\cdot}}} \prod_{k=1}^{K} S(n_{igk}, t_{igk}, a),$$
$$p(y_{ijj'} = 1|\boldsymbol{y}^{-y_{ijj'}}, \boldsymbol{z}, \boldsymbol{w}, \boldsymbol{t} \setminus \mathbb{T}, \boldsymbol{\alpha}, a, b, \eta_0, \eta_1)$$
$$\propto \sum_{\mathbb{T}} \left(\frac{\eta_1 + c_{i1}}{\eta_0 + \eta_1 + c_{i0} + c_{i1}} \mathrm{Beta}_K\left(\boldsymbol{\alpha} + \sum_{\boldsymbol{AS_i}} \boldsymbol{t_{ig}}\right) \right.$$
$$\left. \frac{(b|a)_{t_{i,\boldsymbol{AS_m},\cdot}}}{(b)_{n_{i,\boldsymbol{AS_m},\cdot}}} \prod_{k=1}^{K} S(n_{i,\boldsymbol{AS_m},k}, t_{i,\boldsymbol{AS_m},k}, a) \right),$$

where $\mathbb{T}$ is the set of t_{igk} such that for either or both of $\boldsymbol{AS_l}$ and $\boldsymbol{AS_r}$, $t_{igk} = 1$. c_{i0} and c_{i1} are the total number of y_i's whose values are 0 and that of y_i's whose values are 1, respectively. Note that we change y_i's that relate to the selected action (merging or splitting), in addition to $y_{ijj'}$ to maintain consistency between $\boldsymbol{y}$ and the aligned segment sets.

Inference of $\boldsymbol{\theta}$, $\boldsymbol{\nu}$, $\boldsymbol{\phi}$:

Although our inference does not directly estimate $\boldsymbol{\theta}$, $\boldsymbol{\nu}$, and $\boldsymbol{\phi}$, these variables can be inferred from the following posterior expected values via sampling:

$$\hat{\theta}_{ik} = \mathbb{E}_{\boldsymbol{z_i}, \boldsymbol{t_i}|\boldsymbol{w_i}, \boldsymbol{\alpha}, \boldsymbol{\beta}, a, b, \boldsymbol{y}}\left[\frac{\alpha_k + \sum_{\boldsymbol{AS_i}} t_{igk}}{\sum_{k=1}^{K}(\alpha_k + \sum_{\boldsymbol{AS_i}} t_{igk})}\right],$$
$$\hat{\nu}_{igk} = \mathbb{E}_{\boldsymbol{z_i}, \boldsymbol{t_i}|\boldsymbol{w_i}, \boldsymbol{\alpha}, \boldsymbol{\beta}, a, b, \boldsymbol{y}}\left[\frac{n_{igk} - at_{igk}}{b + n_{ig\cdot}} + \theta_{ik}\frac{at_{ig\cdot} + b}{b + n_{ig\cdot}}\right],$$
$$\hat{\phi}^l_{kw} = \mathbb{E}_{\boldsymbol{z}, \boldsymbol{t}|\boldsymbol{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}, a, b, \boldsymbol{y}}\left[\frac{\beta^l_w + M^l_{kw}}{\sum_{w' \in \boldsymbol{W}^l}(\beta^l_{w'} + M^l_{kw'})}\right].$$

Algorithm 3 Generative Process for Segments

1: **for** each document d^l_i ($i \in \{1, ..., D\}$) **do**
2: choose $\pi^l_i \sim$ Beta(λ_0,λ_1)
3: **for** each passage u^l_{ih} ($h \in \{1, ..., U^l_i\}$) **do**
4: choose $\rho^l_{ih} \sim$ Bernoulli(π^l_i)
5: **end for**
6: $\boldsymbol{s^l_i}$ = concatenate($\boldsymbol{u^l_i}$, $\boldsymbol{\rho^l_i}$)
7: **end for**

4 Integration of Topic Segmentation into BiSTM (BiSTM+TS)

To infer segmentation boundaries simultaneously with cross-lingual topics, we integrate the unsupervised Bayesian topic segmentation method proposed by Du et al. (2013) into the proposed BiSTM (BiSTM+TS).

We assume that each segment is a sequence of topically-related passages. In particular, we consider a sentence as a passage. Our segmentation model defines a segment in document d^l_i by a boundary indicator variable ρ^l_{ih} for each passage u^l_{ih} ($h \in \{1, ..., U^l_i\}$); ρ^l_{ih} is 1 if there is a boundary after passage u^l_{ih} (otherwise 0). For example, $\boldsymbol{\rho^l_i} = (0, 1, 0, 0, 1)$ indicates that the document d^l_i comprises the two segments $\{u^l_{i1}, u^l_{i2}\}$ and $\{u^l_{i3}, u^l_{i4}, u^l_{i5}\}$.

Algorithm 3 shows the generative process for segments. The generative process of BiSTM+TS inserts Algorithm 3 between Steps 7 and 8 of Algorithm 2. Note that two documents $(d^e_i, d^f_i) \in d_i$ are segmented independently. BiSTM+TS assumes that each document d^l_i has its own topic shift probability π^l_i. For each document d^l_i, π^l_i is first drawn from a Beta distribution with the priors λ_0 and λ_1 (Step 2). Then, for each passage u^l_{ih} ($h \in \{1, ..., U^l_i\}$), ρ^l_{ih} is drawn from a Bernoulli distribution with the prior π^l_i (Step 4). Finally, segments $\boldsymbol{s^l_i}$ are generated by concatenating passages based on $\boldsymbol{\rho^l_i}$ (Step 6).

4.1 Inference for BiSTM+TS

Our inference for BiSTM+TS alternates three different kinds of blocks, sampling of $\boldsymbol{\rho}$ and samplings for BiSTM (($\boldsymbol{z}$, $\boldsymbol{\delta}$) and $\boldsymbol{y}$). The conditional distribution of $\boldsymbol{\rho}$ comprises the Gibbs probability for splitting one segment s_m into two segments s_r and s_l by placing the boundary after u^l_{ih} ($\rho^l_{ih} = 1$) and that for merging s_r and s_l to s_m by removing the boundary after u^l_{ih} ($\rho^l_{ih} = 0$).

These probabilities are estimated in the same manner as the conditional probabilities of $y_{ijj'}$, where $\boldsymbol{y}$ ($y_{ijj'} = 0, 1$), $\boldsymbol{AS_l}$, $\boldsymbol{AS_r}$, $\boldsymbol{AS_m}$, η_0, and η_1 are replaced with $\boldsymbol{\rho}$ ($\rho^l_{ih} = 1, 0$), s_l, s_r, s_m, λ_1, and λ_0, respectively, and the statistics t and n are summed for every segment rather than for every aligned segment set (see Equation (6) and (9) in Du et al. (2013)).

Our inference assumes that sampling $\boldsymbol{\rho}$ does not depend on aligned segments in the other language, i.e., $\boldsymbol{y}$[4]. After splitting or merging, we set the y's of s_m, s_l, and s_r as follows: if s_m is split into s_l and s_r, then $AS(s_l) = AS(s_m)$ and $AS(s_r) = AS(s_m)$; if s_l and s_r are merged to s_m, then $AS(s_m) = AS(s_l) \cup AS(s_r)$.

5 Experiment

We evaluated the proposed models in terms of perplexity and performance in translation pair extraction, which is a well-known application that uses a bilingual topic model. We used a document-aligned comparable corpus comprising 3,995 document pairs, each of which is a Japanese Wikipedia article in the Kyoto Wiki Corpus[5] and its corresponding English Wikipedia article[6]. Note that the English articles were collected from the English Wikipedia database dump (2 June 2015)[7] based on inter-language links, even though the original Kyoto Wiki corpus is a parallel corpus, in which each sentence in the Japanese articles is manually translated into English. Thus, our experimental data is not a parallel corpus. We extracted texts from the collected English articles using an open-source script[8]. All Japanese and English texts were segmented using MeCab[9] and TreeTagger[10] (Schmid, 1994), respectively. Then, function words were removed, and the remaining words were lemmatized to reduce data sparsity.

For translation extraction experiments, we automatically created a gold-standard translation set according to Liu et al. (2013). We first computed $p(w^e|w^f)$ and $p(w^f|w^e)$ by running IBM Model 4 on the original Kyoto Wiki corpus, which is a parallel corpus, using GIZA++ (Och and Ney, 2003), and then extracted word pairs $(\hat{w^e}, \hat{w^f})$ that satisfy both of the following conditions: $\hat{w^e} = \text{argmax}_{w^e} p(w^e|w^f = \hat{w^f})$ and $\hat{w^f} = \text{argmax}_{w^f} p(w^f|w^e = \hat{w^e})$. Finally, we eliminated word pairs that do not appear in the document pairs in the document-aligned comparable corpus. We used all 7,930 Japanese words in the resulting gold-standard set as the evaluation input.

5.1 Competing Methods

We compared the proposed models (BiSTM and BiSTM+TS) with a standard bilingual topic model (BiLDA). BiSTM considers each section in Wikipedia articles as a segment. Note that alignments between sections are not given in our experimental data. Thus, $\boldsymbol{y}$ is inferred in both BiSTM and BiSTM+TS.

As in the proposed models, BiLDA was trained using Gibbs sampling (Mimno et al., 2009; Ni et al., 2009; Vulić et al., 2015). In the training of each model, each variable was first initialized. Here, z^l_{ijm} is randomly initialized to an integer between 1 and K, and each of δ^l_{ijm}, $y_{ijj'}$, and ρ^l_{ih} is randomly initialized to 0 or 1. We then performed 10,000 Gibbs iterations. We used the symmetric prior $\alpha_k = 50/K$ and $\beta^l_w = 0.01$ over $\boldsymbol{\theta}$ and $\boldsymbol{\phi}^l$, respectively, in accordance with Vulić et al. (2011). The hyperparameters a, b, λ_0, and λ_1 were set to 0.2, 10, 0.1, and 0.1, respectively, in accordance with Du et al. (2010; 2013). Both η_0 and η_1 were set to 0.2 as a result of preliminary experiments. We used several values of K to measure the impact of topic size: we used $K = 100$ and $K = 400$ in accordance with Liu et al. (2013) in addition to the suggested value $K = 2,000$ in Vulić et al. (2011).

In the translation extraction experiments,

[4] We leave a bilingual extension of the topic segmentation, i.e., incorporation of $\boldsymbol{y}$, for future work.

[5] http://alaginrc.nict.go.jp/WikiCorpus/index_E.html

[6] We filtered out the Japanese articles that do not have corresponding English articles.

[7] http://dumps.wikimedia.org/enwiki/

[8] https://github.com/attardi/wikiextractor/

[9] http://taku910.github.io/mecab/

[10] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

Model	K=100	K=400	K=2,000
BiLDA	693.6	530.7	479.9
BiSTM	520.1	429.3	394.6
BiSTM+TS	537.5	445.3	411.8

Table 2: Test Set Perplexity

we used two translation extraction methods, i.e., *Cue* (Vulić et al., 2011) and *Liu* (Liu et al., 2013). Both methods first infer cross-lingual topics for words using a bilingual topic model (BiLDA/BiSTM/BiSTM+TS) and then extract word pairs (w^e, w^f) with a high value of the probability $p(w^e|w^f)$ defined by the inferred topics. *Cue* calculates $p(w^e|w^f) = \sum_{k=1}^{K} p(w^e|k)p(k|w^f)$, where $p(k|w) \propto \frac{p(w|k)}{\sum_{k=1}^{K} p(w|k)}$ and $p(w|k) = \phi_{kw}$. Liu first converts a document-aligned comparable corpus into a topic-aligned parallel corpus according to the topics of words and computes $p(w^e|w^f, k)$ by running IBM Model 1 on the parallel corpus. Liu then calculates $p(w^e|w^f) = \sum_{k=1}^{K} p(w^e|w^f, k)p(k|w^f)$. Hereafter, a bilingual topic model used in an extraction method is shown in parentheses, e.g., *Cue*(BiLDA) denotes *Cue* with BiLDA.

5.2 Experimental Results

We evaluated the predictive performance of each model by computing the test set perplexity based on 5-fold cross validation. A lower perplexity indicates better generalization performance. Table 2 shows the perplexity of each model. As can be seen, BiSTM and BiSTM+TS are better than BiLDA in terms of perplexity.

We measured the performance of translation extraction with top N accuracy (ACC_N), the number of test words whose top N translation candidates contain a correct translation over the total number of test words (7,930). Table 3 summarizes ACC_1 and ACC_{10} for each model. As can be seen, *Cue*/*Liu*(BiSTM) and *Cue*/*Liu*(BiSTM+TS) significantly outperform *Cue*/*Liu*(BiLDA) ($p < 0.01$ in the sign test). This indicates that BiSTM and BiSTM+TS improve the performance of translation extraction for both the *Cue* and *Liu* methods by assigning more suitable topics.

Both experiments prove that capturing segment-level alignments is effective for modeling bilingual data. In addition, these experiments show that BiSTM+TS is comparable with BiSTM, indicating that the proposed model could yield a significant benefit even if the boundaries of segments are unknown.

ACC_1			
Method	K=100	K=400	K=2,000
Cue(BiLDA)	0.024	0.056	0.101
Cue(BiSTM)	0.055	0.112	0.184
Cue(BiSTM+TS)	0.052	0.107	0.176
Liu(BiLDA)	0.206	0.345	0.426
Liu(BiSTM)	0.287	0.414	0.479
Liu(BiSTM+TS)	0.283	0.406	0.467
ACC_{10}			
Method	K=100	K=400	K=2,000
Cue(BiLDA)	0.093	0.170	0.281
Cue(BiSTM)	0.218	0.286	0.410
Cue(BiSTM+TS)	0.196	0.274	0.398
Liu(BiLDA)	0.463	0.550	0.603
Liu(BiSTM)	0.531	0.625	0.671
Liu(BiSTM+TS)	0.536	0.612	0.667

Table 3: Performance of Translation Extraction

	Reference $y = 1$	Reference $y = 0$
Inference $y = 1$	195	174
Inference $y = 0$	43	1132

Table 4: Distribution of Segment-level Alignments

Tables 2 and 3 show that a larger topic size yields better performance for each model. Furthermore, *Liu* outperforms *Cue* regardless of the choice of bilingual topic models, which is consistent with previously reported results (Liu et al., 2013). The results of our experiments demonstrate that the proposed models have the same tendencies as BiLDA.

6 Discussion

6.1 Inferred Segment-level Alignments

We created a reference set to evaluate segment-level alignments $\boldsymbol{y}$ inferred by BiSTM (K=2,000). We randomly selected 100 document pairs from the comparable corpus and then manually identified cross-lingual alignments between sections. Table 4 shows the distribution of inferred $\boldsymbol{y}$ values and that of $\boldsymbol{y}$ values in the reference set. As can be seen, the accuracy of $\boldsymbol{y}$ is 0.859 (1,327/1,544).

The majority of false negatives (121/174) are sections that are not parallel but correspond partially. An example is the alignment between the

Model	Japanese article	English article
BiSTM	4.8	2.9
BiSTM+TS	10.6	4.1

Table 5: Average Number of Segments

Japanese section "history" and the English section "Bujutsu (old type of Budo)" in the "Budo (a Japanese martial art)" article pair, where a part of the English section "Bujutsu" is described in the Japanese section "history." Such errors might not necessarily have a negative effect, because partial alignments can be useful.

6.2 Inferred Segmentation Boundaries

This section compares segment boundaries inferred by BiSTM+TS (K=2,000) with section boundaries in the original articles, which have been referred to by BiSTM. The recall of BiSTM+TS for the original section boundaries is 0.727. This indicates that the unsupervised segmentation in BiSTM+TS finds drastic topical changes, i.e., section boundaries, with high recall.

Table 5 shows the average number of segments per article for each model. As can be seen, BiSTM+TS divides an article into segments smaller than the original sections. This seems to be reasonable, because some original sections include multiple topics. However, Tables 2 and 3 show that inferred boundaries do not work better than section boundaries. One reason for that is that some errors are caused by a sparseness problem, when BiSTM+TS separates an article into extremely fine-grained segments. In addition, Table 5 reveals that BiSTM+TS increases the gap between languages. Thus, segmentation with a comparable granularity between languages might be favorable for the proposed models.

6.3 Effectiveness for an English–French Wikipedia Corpus

We evaluated BiLDA, BiSTM, and BiSTM+TS in terms of perplexity and performance in translation extraction on an English–French Wikipedia corpus to verify the effectiveness of the proposed models for language pairs other than English–Japanese. The settings, e.g., parameters, for each model are the same as in Section 5. Note that we report only the performances of each model with $K = 2,000$, because all models achieved the best performances when $K = 2,000$.

Model	Test Set Perplexity	
BiLDA	439.1	
BiSTM	379.4	
BiSTM+TS	396.6	
Model	ACC_1	ACC_{10}
Cue(BiLDA)	0.219	0.556
Cue(BiSTM)	0.275	0.580
Cue(BiSTM+TS)	0.257	0.582
Liu(BiLDA)	0.715	0.838
Liu(BiSTM)	0.742	0.859
Liu(BiSTM+TS)	0.732	0.852

Table 6: Performance on an English–French Wikipedia Corpus ($K = 2,000$)

We collected French articles that correspond to the English articles used in the experiments in Section 5, from the French Wikipedia database dump (2 June 2015) based on inter-language links. As a result, our English–French corpus comprises 3,159 document pairs. The French articles were preprocessed in the same manner as the English articles: text extraction using the open-source script, segmentation using TreeTagger, removal of function words, and lemmatization.

We created a gold-standard translation set for translation extraction experiments using Google Translate service[11] in a manner similar to that in Gouws et al. (2015) and Coulmance et al. (2015), translating the French words in our corpus using Google Translate, and then eliminating word pairs that do not appear in the document pairs in our corpus. We used the top 1,000 most frequent French words in the resulting gold-standard set as the evaluation input.

Table 6 summarizes ACC_1, ACC_{10}, and perplexity. It shows that the proposed models are effective also for the English–French Wikipedia corpus. BiSTM and BiSTM+TS outperform BiLDA in terms of perplexity and performance of translation extraction, and BiSTM+TS works well even if the boundaries of segments are unknown.

7 Related Work

Multilingual topic models other than BiLDA (Section 2) have been proposed for document-aligned comparable corpora. Fukumasu et al. (2012) applied SwitchLDA (Newman et al., 2006) and Correspondence LDA (Blei and Jordan, 2003), which

[11] http://translate.google.com/

were originally intended to work with multimodal data, such as annotated image data, to modeling multilingual text data. They also proposed a symmetric version of Correspondence LDA. Platt et al. (2010) projected monolingual models based on PLSA or Principal Component Analysis into a shared multilingual space with the constraint that document pairs must map to similar locations. Hu et al. (2014) proposed a multilingual tree-based topic model that uses a hierarchical bilingual dictionary in addition to document alignments. Note that these models do not consider segment-level alignments.

There are several multilingual topic models tailored for data other than a document-aligned comparable corpus, including bilingual topic models for word alignment and machine translation on parallel sentence pairs (Zhao and Xing, 2006; Zhao and Xing, 2008). Some models have mined multilingual topics from unaligned text data by bridging the gap between different languages using a bilingual dictionary (Jagarlamudi and Daumé III, 2010; Zhang et al., 2010; Negi, 2011). Boyd-Graber and Blei (2009) used parallel sentences in combination with a bilingual dictionary. However, these models have the drawback that they require a parallel corpus or a bilingual dictionary in advance, which cannot be obtained for some language pairs or domains.

In a monolingual setting, some topic models that consider segment-level topics have been proposed. Du et al. (2010) considered a document as a set of segments and generated each per-segment topic distribution from the topic distribution of the related document through a Pitman–Yor process. Others have considered a document as a sequence of segments. Cheng et al. (2009) reflected the underlying sequences of segments' topics by positing a permutation distribution over a document. Wang et al. (2011) modeled topical sequences in documents with a latent first-order Markov chain, and Du et al. (2012) generated each per-segment topic distribution from the topic distribution of its document and that of its previous segment. Note that none of these models have been extended to a multilingual setting.

8 Conclusions

In this paper, we proposed BiSTM, which models a document hierarchically and deals with segment-level alignments. BiSTM assigns the same topic distribution to both aligned documents and aligned segments. We also presented an extended model, BiSTM+TS, that infers segmentation boundaries in addition to latent topics by incorporating unsupervised topic segmentation (Du et al., 2013). Our experimental results show that capturing segment-level alignments improves perplexity and translation extraction performance, and that BiSTM+TS yields a significant benefit even if the boundaries of segments are not given.

This paper presented an extension to BiLDA, but hierarchical structures can also be incorporated into other bilingual topic models (Section 7). As future work, we would like to verify the effectiveness of the proposed models for other datasets or other cross-lingual tasks, such as cross-lingual document classification (Ni et al., 2009; Platt et al., 2010; Ni et al., 2011; Smet et al., 2011) and cross-lingual information retrieval (Vulić et al., 2013).

Acknowledgments

We thank Atsushi Fujita for valuable comments on earlier versions of this manuscript.

References

David M. Blei and Michael I. Jordan. 2003. Modeling Annotated Data. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pages 127–134.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David M. Blei. 2009. Multilingual Topic Models for Unaligned Text. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 75–82.

Wray Buntine and Marcus Hutter. 2012. A Bayesian View of the Poisson-Dirichlet Process. `http://arxiv.org/pdf/1007.0296.pdf`.

Harr Chen, S.R.K. Branavan, Regina Barzilay, and David R. Karger. 2009. Global Models of Document Structure using Latent Permutations. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 371–379.

Changyou Chen, Lan Du, and Wray Buntine. 2011. Sampling Table Configurations for the Hierarchical Poisson-Dirichlet Process. In *Proceedings of the European Conference on Machine Learning and*

Principles and Practice of Knowledge Discovery in Databases 2011, pages 296–311.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, Fast Cross-lingual Word-embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1109–1113.

Lan Du, Wray Buntine, and Huidong Jin. 2010. A Segmented Topic Model Based on the Two-parameter Poisson-Dirichlet Process. *Machine Learning*, 81(1):5–19.

Lan Du, Wray Buntine, and Huidong Jin. 2012. Modelling Sequential Text with an Adaptive Topic Model. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 535–545.

Lan Du, Wray Buntine, and Mark Johnson. 2013. Topic Segmentation with a Structured Topic Model. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 190–200.

Kosuke Fukumasu, Koji Eguchi, and Eric P. Xing. 2012. Symmetric Correspondence Topic Models for Multilingual Text Analysis. In *Advances in Neural Information Processing Systems 25*, pages 1286–1294.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 748–756.

Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57.

Leetsch C. Hsu and Peter Jau-Shyong Shiue. 1998. A Unified Approach to Generalized Stirling Numbers. *Advances in Applied Mathematics*, 20(3):366–384.

Yuening Hu, Ke Zhai, Vladimir Eidelman, and Jordan Boyd-Graber. 2014. Polylingual Tree-Based Topic Models for Translation Domain Adaptation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1166–1176.

Jagadeesh Jagarlamudi and Hal Daumé III. 2010. Extracting Multilingual Topics from Unaligned Comparable Corpora. In *Proceedings of the 32nd European Conference on Advances in Information Retrieval*, pages 444–456.

Xiaodong Liu, Kevin Duh, and Yuji Matsumoto. 2013. Topic Models + Word Alignment = A Flexible Framework for Extracting Bilingual Dictionary from Comparable Corpus. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 212–221.

David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual Topic Models. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 880–889.

Sumit Negi. 2011. Mining Bilingual Topic Hierarchies from Unaligned Text. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 992–1000.

David Newman, Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2006. Statistical Entity-topic Models. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 680–686.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2009. Mining Multilingual Topics from Wikipedia. In *Proceedings of the 18th International World Wide Web Conference*, pages 1155–1156.

Xiaochuan Ni, Jian-Tao Sun, Jian Hu, and Zheng Chen. 2011. Cross Lingual Text Classification by Mining Multilingual Topics from Wikipedia. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 375–384.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29:19–51.

Jim Pitman and Marc Yor. 1997. The Two-Parameter Poisson-Dirichlet Distribution Derived from a Stable Subordinator. *The Annals of Probability*, 25(2):855–900.

John Platt, Kristina Toutanova, and Wen tau Yih. 2010. Translingual Document Representations from Discriminative Projections. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 251–261.

Helmut Schmid. 1994. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *Proceedings of the International Conference on New Methods in Language Processing*, pages 44–49.

Wim De Smet, Jie Tang, and Marie-Francine Moens. 2011. Knowledge Transfer Across Multilingual Corpora via Latent Topics. In *Proceedings of the 15th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 549–560.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2011. Identifying Word Translations from Comparable Corpora Using Latent Topic Models. In *Proceedings of the 49th Annual Meeting of the Associ-*

ation for Computational Linguistics: Human Language Technologies, pages 479–484.

Ivan Vulić, Wim De Smet, and Marie-Francine Moens. 2013. Cross-Language Information Retrieval Models Based on Latent Topic Models Trained with Document-Aligned Comparable Corpora. *Information Retrieval*, 16(3):331–368.

Ivan Vulić, Wim De Smet, Jie Tang, and Marie-Francine Moens. 2015. Probabilistic Topic Modeling in Multilingual Settings: An Overview of Its Methodology and Applications. *Information Processing & Management*, 51(1):111–147.

Hongning Wang, Duo Zhang, and ChengXiang Zhai. 2011. Structural Topic Model for Latent Topical Structure Analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1526–1535.

Duo Zhang, Qiaozhu Mei, and ChengXiang Zhai. 2010. Cross-Lingual Latent Topic Extraction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1128–1137.

Bing Zhao and Eric P. Xing. 2006. BiTAM: Bilingual Topic AdMixture Models for Word Alignment. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 969–976.

Bing Zhao and Eric P. Xing. 2008. HM-BiTAM: Bilingual Topic Exploration, Word Alignment, and Translation. In *Advances in Neural Information Processing Systems 20*, pages 1689–1696.

Learning Semantically and Additively Compositional Distributional Representations

Ran Tian and **Naoaki Okazaki** and **Kentaro Inui**
Tohoku University, Japan
{tianran, okazaki, inui}@ecei.tohoku.ac.jp

Abstract

This paper connects a vector-based composition model to a formal semantics, the Dependency-based Compositional Semantics (DCS). We show theoretical evidence that the vector compositions in our model conform to the logic of DCS. Experimentally, we show that vector-based composition brings a strong ability to calculate similar phrases as similar vectors, achieving near state-of-the-art on a wide range of phrase similarity tasks and relation classification; meanwhile, DCS can guide building vectors for structured queries that can be directly executed. We evaluate this utility on sentence completion task and report a new state-of-the-art.

1 Introduction

A major goal of semantic processing is to map natural language utterances to representations that facilitate calculation of meanings, execution of commands, and/or inference of knowledge. Formal semantics supports such representations by defining words as some functional units and combining them via a specific logic. A simple and illustrative example is the Dependency-based Compositional Semantics (DCS) (Liang et al., 2013). DCS composes meanings from denotations of words (i.e. sets of things to which the words apply); say, the denotations of the concept `drug` and the event `ban` is shown in Figure 1b, where `drug` is a list of drug names and `ban` is a list of the subject-complement pairs in any *ban* event; then, a list of *banned drugs* can be constructed by first taking the COMP column of all records in `ban` (projection "π_{COMP}"), and then intersecting the results with `drug` (intersection "$\cap$"). This procedure defined how words can be combined to form a meaning. Better yet, the procedure can be concisely illustrated by the DCS tree of "*banned drugs*" (Figure 1a), which is similar to a dependency tree but possesses precise procedural and logical meaning (Section 2). DCS has been shown useful in question answering (Liang et al., 2013) and textual entailment recognition (Tian et al., 2014).

Orthogonal to the formal semantics of DCS, distributional vector representations are useful in capturing lexical semantics of words (Turney and Pantel, 2010; Levy et al., 2015), and progress is made in combining the word vectors to form meanings of phrases/sentences (Mitchell and Lapata, 2010; Baroni and Zamparelli, 2010; Grefenstette and Sadrzadeh, 2011; Socher et al., 2012; Paperno et al., 2014; Hashimoto et al., 2014). However, less effort is devoted to finding a link between vector-based compositions and the composition operations in any formal semantics. We believe that if a link can be found, then symbolic formulas in the formal semantics will be realized by vectors composed from word embeddings, such that similar things are realized by similar vectors; meanwhile, vectors will acquire formal meanings that can directly be used in execution or inference process. Still, to find a link is challenging because any vector compositions that realize such a link must conform to the logic of the formal semantics.

In this paper, we establish a link between DCS and certain vector compositions, achieving a *vector-based DCS* by replacing denotations of words with word vectors, and realizing the composition operations such as intersection and projection as addition and linear mapping, respectively. For example, to construct a vector for "*banned drugs*", one takes the word vector $\mathbf{v}_{ban}$ and multiply it by a matrix M_{COMP}, corresponding to the projection π_{COMP}; then, one adds the result to the word vector $\mathbf{v}_{drug}$ to realize the intersection operation (Figure 1c). We provide a method to train the

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1277–1287,
Berlin, Germany, August 7-12, 2016.

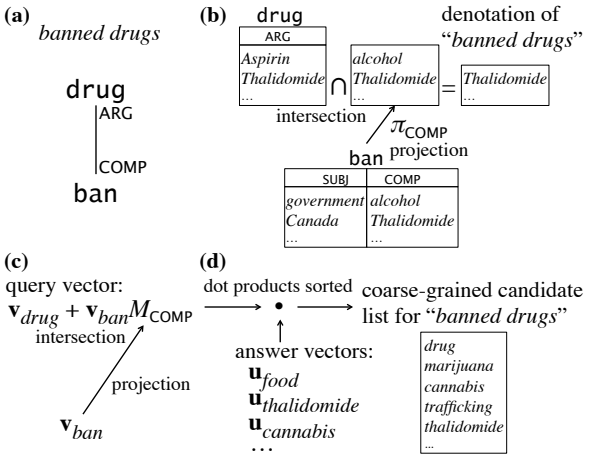


Figure 1: (a) The DCS tree of "*banned drugs*", which controls (b) the calculation of its denotation. In this paper, we learn word vectors and matrices such that (c) the same calculation is realized in distributional semantics. The constructed query vector can be used to (d) retrieve a list of coarse-grained candidate answers to that query.

word vectors and linear mappings (i.e. matrices) jointly from unlabeled corpora.

The rationale for our model is as follows. First, recent research has shown that additive composition of word vectors is an approximation to the situation where two words have overlapping context (Tian et al., 2015); therefore, it is suitable to implement an "and" or intersection operation (Section 3). We design our model such that the resulted distributional representations are expected to have additive compositionality. Second, when intersection is realized as addition, it is natural to implement projection as linear mapping, as suggested by the logical interactions between the two operations (Section 3). Experimentally, we show that vectors and matrices learned by our model exhibit favorable characteristics as compared with vectors trained by GloVe (Pennington et al., 2014) or those learned from syntactic dependencies (Section 5.1). Finally, additive composition brings our model a strong ability to calculate similar vectors for similar phrases, whereas syntactic-semantic roles (e.g. SUBJ, COMP) can be distinguished by different projection matrices (e.g. M_{SUBJ}, M_{COMP}). We achieve near state-of-the-art performance on a wide range of phrase similarity tasks (Section 5.2) and relation classification (Section 5.3).

Furthermore, we show that a vector as constructed above for "*banned drugs*" can be used as a *query vector* to retrieve a coarse-grained candidate list of banned drugs, by sorting its dot products with *answer vectors* that are also learned by our model (Figure 1d). This is due to the ability of our approach to provide a language model that can find likely words to fill in the blanks such as "__ *is a banned drug*" or "*the drug* __ *is banned by* ...". A highlight is the calculation being done as if a query is "executed" by the DCS tree of "*banned drugs*". We quantitatively evaluate this utility on sentence completion task (Zweig et al., 2012) and report a new state-of-the-art (Section 5.4).

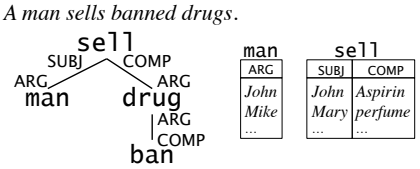


Figure 2: DCS tree for a sentence

2 DCS Trees

DCS composes meanings from denotations, or sets of things to which words apply. A "thing" (i.e. element of a denotation) is represented by a tuple of features of the form Field=*Value*, with a fixed inventory of fields. For example, a denotation `ban` might be a set of tuples $\texttt{ban} = \{(\mathsf{SUBJ}{=}\textit{Canada}, \mathsf{COMP}{=}\textit{Thalidomide}), \ldots\}$, in which each tuple records participants of a banning event (e.g. Canada banning Thalidomide).

Operations are applied to sets of things to generate new denotations, for modeling semantic composition. An example is the intersection of `pet` and `fish` giving the denotation of "*pet fish*". Another necessary operation is projection; by π_{N} we mean a function mapping a tuple to its value of the field N. For example, $\pi_{\mathsf{COMP}}(\texttt{ban})$ is the value set of the COMP fields in `ban`, which consists of banned objects (i.e. $\{\textit{Thalidomide}, \ldots\}$). In this paper, we assume a field ARG to be names of things representing themselves, hence for example $\pi_{\mathsf{ARG}}(\texttt{drug})$ is the set of names of drugs.

For a value set V, we also consider inverse image $\pi_{\mathsf{N}}^{-1}(V) := \{x \mid \pi_{\mathsf{N}}(x) \in V\}$. For example,

$$\texttt{D}_1 := \pi_{\mathsf{SUBJ}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{man}))$$

consists of all tuples of the form $(\mathsf{SUBJ}{=}x, \ldots)$, where x is a man's name (i.e. $x \in \pi_{\mathsf{ARG}}(\texttt{man})$). Thus, $\texttt{sell} \cap \texttt{D}_1$ denotes men's selling events (i.e. $\{(\mathsf{SUBJ}{=}\textit{John}, \mathsf{COMP}{=}\textit{Aspirin}), \ldots\}$ as in Figure 2). Similarly, the denotation of "*banned*

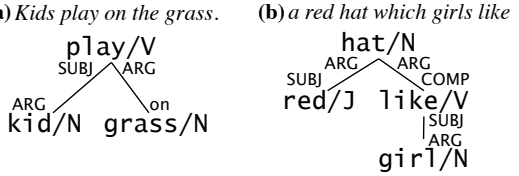


Figure 3: DCS trees in this work

drugs" as in Figure 1b is formally written as

$$\mathtt{D}_2 := \mathtt{drug} \cap \pi^{-1}_{\mathsf{ARG}}(\pi_{\mathsf{COMP}}(\mathtt{ban})),$$

Hence the following denotation

$$\mathtt{D}_3 := \mathtt{sell} \cap \mathtt{D}_1 \cap \pi^{-1}_{\mathsf{COMP}}(\pi_{\mathsf{ARG}}(\mathtt{D}_2))$$

consists of selling events such that the SUBJ is a man and the COMP is a banned drug.

The calculation above can proceed in a recursive manner controlled by DCS trees. The DCS tree for the sentence "*a man sells banned drugs*" is shown in Figure 2. Formally, a DCS tree is defined as a rooted tree in which nodes are denotations of content words and edges are labeled by fields at each ends. Assume a node $\mathtt{x}$ has children $\mathtt{y}_1, \ldots, \mathtt{y}_n$, and the edges $(\mathtt{x}, \mathtt{y}_1), \ldots, (\mathtt{x}, \mathtt{y}_n)$ are labeled by $(\mathsf{P}_1, \mathsf{L}_1), \ldots, (\mathsf{P}_n, \mathsf{L}_n)$, respectively. Then, the denotation $[\![\mathtt{x}]\!]$ of the subtree rooted at $\mathtt{x}$ is recursively calculated as

$$[\![\mathtt{x}]\!] := \mathtt{x} \cap \bigcap_{i=1}^{n} \pi^{-1}_{\mathsf{P}_i}(\pi_{\mathsf{L}_i}([\![\mathtt{y}_i]\!])). \qquad (1)$$

As a result, the denotation of the DCS tree in Figure 2 is the denotation $\mathtt{D}_3$ of "*a man sells banned drugs*" as calculated above. DCS can be further extended to handle phenomena such as quantifiers or superlatives (Liang et al., 2013; Tian et al., 2014). In this paper, we focus on the basic version, but note that it is already expressive enough to at least partially capture the meanings of a large portion of phrases and sentences.

DCS trees can be learned from question-answer pairs and a given database of denotations (Liang et al., 2013), or they can be extracted from dependency trees if no database is specified, by taking advantage of the observation that DCS trees are similar to dependency trees (Tian et al., 2014). We use the latter approach, obtaining DCS trees by rule-based conversion from universal dependency (UD) trees (McDonald et al., 2013). Therefore, nodes in a DCS tree are content words in a UD tree, which are in the form of lemma-POS pairs (Figure 3). The inventory of fields is designed to be ARG, SUBJ, COMP, and all prepositions. Prepositions are unlike content words which denote sets of things, but act as relations which we treat similarly as SUBJ and COMP. For example, a prepositional phrase attached to a verb (e.g. *play on the grass*) is treated as in Figure 3a. The presence of two field labels on each edge of a DCS tree makes it convenient for modeling semantics in several cases, such as a relative clause (Figure 3b).

3 Vector-based DCS

For any content word w, we use a query vector $\mathbf{v}_w$ to model its denotation, and an answer vector $\mathbf{u}_w$ to model a prototypical element in that denotation. Query vector $\mathbf{v}$ and answer vector $\mathbf{u}$ are learned such that $\exp(\mathbf{v} \cdot \mathbf{u})$ is proportional to the probability of $\mathbf{u}$ answering the query $\mathbf{v}$. The learning source is a collection of DCS trees, based on the idea that the DCS tree of a declarative sentence usually has non-empty denotation. For example, "*kids play*" means there exists some kid who plays. Consequently, some element in the play denotation belongs to $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$, and some element in the kid denotation belongs to $\pi^{-1}_{\mathsf{ARG}}(\pi_{\mathsf{SUBJ}}(\mathtt{play}))$. This is a signal to increase the dot product of $\mathbf{u}_{play}$ and the query vector of $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$, as well as the dot product of $\mathbf{u}_{kid}$ and the query vector of $\pi^{-1}_{\mathsf{ARG}}(\pi_{\mathsf{SUBJ}}(\mathtt{play}))$. When optimized on a large corpus, the "typical" elements of play and kid should be learned by $\mathbf{u}_{play}$ and $\mathbf{u}_{kid}$, respectively. In general, one has

Theorem 1 *Assume the denotation of a DCS tree is not empty. Given any path from node* $\mathtt{x}$ *to* $\mathtt{y}$*, assume edges along the path are labeled by* $(\mathsf{P}, \mathsf{L}), \ldots, (\mathsf{K}, \mathsf{N})$*. Then, an element in the denotation* $\mathtt{y}$ *belongs to* $\pi^{-1}_{\mathsf{N}}(\pi_{\mathsf{K}}(\ldots(\pi^{-1}_{\mathsf{L}}(\pi_{\mathsf{P}}(\mathtt{x}) \ldots).$

Therefore, for any two nodes in a DCS tree, the path from one to another forms a training example, which signals increasing the dot product of the corresponding query and answer vectors.

It is noteworthy that the above formalization happens to be closely related to the skip-gram model (Mikolov et al., 2013b). The skip-gram learns a target vector $\mathbf{v}_w$ and a context vector $\mathbf{u}_w$ for each word w. It assumes the probability of a word y co-occurring with a word x in a context window is proportional to $\exp(\mathbf{v}_x \cdot \mathbf{u}_y)$. Hence, if x and y co-occur within a context window, then one gets a signal to increase $\mathbf{v}_x \cdot \mathbf{u}_y$. If the context window is taken as the same DCS tree, then

the learning of skip-gram and vector-based DCS will be almost the same, except that the target vector $\mathbf{v}_x$ becomes the query vector $\mathbf{v}$, which is no longer assigned to the word x but the path from x to y in the DCS tree (e.g. the query vector for $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$ instead of $\mathbf{v}_{kid}$). Therefore, our model can also be regarded as extending skip-gram to take account of the changes of meanings caused by different syntactic-semantic roles.

Additive Composition Word vectors trained by skip-gram are known to be semantically additive, such as exhibited in word analogy tasks. An effect of adding up two skip-gram vectors is further analyzed in Tian et al. (2015). Namely, the target vector $\mathbf{v}_w$ can be regarded as encoding the distribution of context words surrounding w. If another word x is given, $\mathbf{v}_w$ can be decomposed into two parts, one encodes context words shared with x, and another encodes context words not shared. When $\mathbf{v}_w$ and $\mathbf{v}_x$ are added up, the non-shared part of each of them tend to cancel out, because non-shared parts have nearly independent distributions. As a result, the shared part gets reinforced. An error bound is derived to estimate how close $\frac{1}{2}(\mathbf{v}_w + \mathbf{v}_x)$ gets to the distribution of the shared part. We can see the same mechanism exists in vector-based DCS. In a DCS tree, two paths share a context word if they lead to a same node y; semantically, this means some element in the denotation $\mathtt{y}$ belongs to both denotations of the two paths (e.g. given the sentence "*kids play balls*", $\pi^{-1}_{\mathsf{SUBJ}}(\pi_{\mathsf{ARG}}(\mathtt{kid}))$ and $\pi^{-1}_{\mathsf{COMP}}(\pi_{\mathsf{ARG}}(\mathtt{ball}))$ both contain a playing event whose SUBJ is a kid and COMP is a ball). Therefore, addition of query vectors of two paths approximates their intersection because the shared context y gets reinforced.

Projection Generally, for any two denotations $\mathtt{X}_1, \mathtt{X}_2$ and any projection π_{N}, we have

$$\pi_{\mathsf{N}}(\mathtt{X}_1 \cap \mathtt{X}_2) \subseteq \pi_{\mathsf{N}}(\mathtt{X}_1) \cap \pi_{\mathsf{N}}(\mathtt{X}_2). \quad (2)$$

And the "$\subseteq$" can often become "$=$", for example when π_{N} is a one-to-one map or $\mathtt{X}_1 = \pi^{-1}_{\mathsf{N}}(V)$ for some value set V. Therefore, if intersection is realized by addition, it will be natural to realize projection by linear mapping because

$$(\mathbf{v}_1 + \mathbf{v}_2)M_{\mathsf{N}} = \mathbf{v}_1 M_{\mathsf{N}} + \mathbf{v}_2 M_{\mathsf{N}} \quad (3)$$

holds for any vectors $\mathbf{v}_1, \mathbf{v}_2$ and any matrix M_{N}, which is parallel to (2). If π_{N} is realized by a matrix M_{N}, then π^{-1}_{N} should correspond to the inverse matrix M^{-1}_{N}, because $\pi_{\mathsf{N}}(\pi^{-1}_{\mathsf{N}}(V)) = V$ for any value set V. So we have realized all composition operations in DCS.

Query vector of a DCS tree Now, we can define the query vector of a DCS tree as parallel to (1):

$$\mathbf{v}_{[\![\mathtt{x}]\!]} := \mathbf{v}_x + \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_{[\![\mathtt{y}_i]\!]} M_{\mathsf{L}_i} M^{-1}_{\mathsf{P}_i}. \quad (4)$$

4 Training

As described in Section 3, vector-based DCS assigns a query vector $\mathbf{v}_w$ and an answer vector $\mathbf{u}_w$ to each content word w. And for each field N, it assigns two matrices M_{N} and M^{-1}_{N}. For any path from node x to y sampled from a DCS tree, assume the edges along are labeled by $(\mathsf{P}, \mathsf{L}), \ldots, (\mathsf{K}, \mathsf{N})$. Then, the dot product $\mathbf{v}_x M_{\mathsf{P}} M^{-1}_{\mathsf{L}} \ldots M_{\mathsf{K}} M^{-1}_{\mathsf{N}} \cdot \mathbf{u}_y$ gets a signal to increase.

Formally, we adopt the noise-contrastive estimation (Gutmann and Hyvärinen, 2012) as used in the skip-gram model, and mix the paths sampled from DCS trees with artificially generated noise. Then, $\sigma(\mathbf{v}_x M_{\mathsf{P}} M^{-1}_{\mathsf{L}} \ldots M_{\mathsf{K}} M^{-1}_{\mathsf{N}} \cdot \mathbf{u}_y)$ models the probability of a training example coming from DCS trees, where $\sigma(\theta) = 1/\{1 + \exp(-\theta)\}$ is the sigmoid function. The vectors and matrices are trained by maximizing the log-likelihood of the mixed data. We use stochastic gradient descent (Bottou, 2012) for training. Some important settings are discussed below.

Noise For any $\mathbf{v}_x M_1 M_2^{-1} \ldots M_{2l-1} M_{2l}^{-1} \cdot \mathbf{u}_y$ obtained from a path of a DCS tree, we generate noise by randomly choosing an index $i \in [2, 2l]$, and then replacing M_j or M_j^{-1} ($\forall j \geq i$) and $\mathbf{u}_y$ by $M_{\mathsf{N}(j)}$ or $M^{-1}_{\mathsf{N}(j)}$ and $\mathbf{u}_z$, respectively, where $\mathsf{N}(j)$ and z are independently drawn from the marginal (i.e. unigram) distributions of fields and words.

Update For each data point, when i is the chosen index above for generating noise, we view indices $j < i$ as the "target" part, and $j >= i$ as the "context", which is completely replaced by the noise, as an analogous to the skip-gram model. Then, at each step we only update one vector and one matrix from each of the target, context, and noise part; more specifically, we only update $\mathbf{v}_x$, M_{i-1} or M^{-1}_{i-1}, M_i or M_i^{-1}, $M_{\mathsf{N}(i)}$ or $M^{-1}_{\mathsf{N}(i)}$, $\mathbf{u}_y$ and $\mathbf{u}_z$, at the step. This is much faster than always updating all matrices.

Initialization Matrices are initialized as $\frac{1}{2}(I + G)$, where I is the identity matrix; and G and all

GloVe	no matrix	vecDCS	vecUD
books	*essay/N*	*novel/N*	*essay/N*
author	*novel/N*	*essay/N*	*novel/N*
published	*memoir/N*	*anthology/N*	*article/N*
novel	*books/N*	*publication/N*	*anthology/N*
memoir	*autobiography/N*	*memoir/N*	*poem/N*
wrote	***non-fiction/J***	*poem/N*	*autobiography/N*
biography	***reprint/V***	*autobiography/N*	*publication/N*
autobiography	***publish/V***	*story/N*	*journal/N*
essay	***republish/V***	*pamphlet/N*	*memoir/N*
illustrated	*chapbook/N*	*tale/N*	*pamphlet/N*

Table 1: Top 10 similar words to "*book/N*"

vectors are initialized with i.i.d. Gaussians of variance $1/d$, where d is the vector dimension. We find that the diagonal component I is necessary to bring information from $\mathbf{v}_x$ to $\mathbf{u}_y$, whereas the randomness of G makes convergence faster. M_{N}^{-1} is initialized as the transpose of M_{N}.

Learning Rate We find that the initial learning rate for vectors can be set to 0.1. But for matrices, it should be less than 0.0005 otherwise the model diverges. For stable training, we rescale gradients when their norms exceed a threshold.

Regularizer During training, M_{N} and M_{N}^{-1} are treated as independent matrices. However, we use the regularizer $\gamma\|M_{\mathsf{N}}^{-1}M_{\mathsf{N}} - \frac{1}{d}\operatorname{tr}(M_{\mathsf{N}}^{-1}M_{\mathsf{N}})I\|^2$ to drive M_{N}^{-1} close to the inverse of M_{N}.[1] We also use $\kappa\|M_{\mathsf{N}}^{\perp}M_{\mathsf{N}} - \frac{1}{d}\operatorname{tr}(M_{\mathsf{N}}^{\perp}M_{\mathsf{N}})I\|^2$ to prevent M_{N} from having too different scales at different directions (i.e., to drive M_{N} close to orthogonal). We set $\gamma = 0.001$ and $\kappa = 0.0001$. Despite the rather weak regularizer, we find that M_{N}^{-1} can be learned to be exactly the inverse of M_{N}, and M_{N} can actually be an orthogonal matrix, showing some semantic regularity (Section 5.1).

5 Experiments

For training vector-based DCS, we use Wikipedia Extractor[2] to extract texts from the 2015-12-01 dump of English Wikipedia[3]. Then, we use Stanford Parser[4] (Klein and Manning, 2003) to parse all sentences and convert the UD trees into DCS trees by handwritten rules. We assign a weight to each path of the DCS trees as follows.

[1]Problem with the naive regularizer $\|M^{-1}M - I\|^2$ is that, when the scale of M goes larger, it will drive M^{-1} smaller, which may lead to degeneration. So we scale I according to the trace of $M^{-1}M$.

[2]`http://medialab.di.unipi.it/wiki/Wikipedia_Extractor`

[3]`https://dumps.wikimedia.org/enwiki/`

[4]`http://nlp.stanford.edu/software/lex-parser.shtml`

$\pi_{\mathsf{SUBJ}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{house}))$	$\pi_{\mathsf{COMP}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{house}))$	$\pi_{\mathsf{ARG}}^{-1}(\pi_{\mathsf{in}}(\texttt{house}))$
victorian/J	*build/V*	*sit/V*
stand/V	*rent/V*	*house/N*
vacant/J	*leave/V*	*stand/V*
18th-century/J	*burn down/V*	*live/V*
historic/J	*remodel/V*	*hang/V*
old/J	*demolish/V*	*seat/N*
georgian/J	*restore/V*	*stay/V*
local/J	*renovate/V*	*serve/V*
19th-century/J	*rebuild/V*	*reside/V*
tenement/J	*construct/V*	*hold/V*
$\pi_{\mathsf{ARG}}^{-1}(\pi_{\mathsf{SUBJ}}(\texttt{learn}))$	$\pi_{\mathsf{ARG}}^{-1}(\pi_{\mathsf{COMP}}(\texttt{learn}))$	$\pi_{\mathsf{about}}^{-1}(\pi_{\mathsf{ARG}}(\texttt{learn}))$
teacher/N	*skill/N*	*otherness/N*
skill/N	*lesson/N*	*intimacy/N*
he/P	*technique/N*	*femininity/N*
she/P	*experience/N*	*self-awareness/N*
therapist/N	*ability/N*	*life/N*
student/N	*something/N*	*self-expression/N*
they/P	*knowledge/N*	*sadomasochism/N*
mother/N	*language/N*	*emptiness/N*
lesson/N	*opportunity/N*	*criminality/N*
father/N	*instruction/N*	*masculinity/N*

Table 2: Top 10 answers of high dot products

For any path P passing through k intermediate nodes of degrees $n_1, \ldots, n_k$, respectively, we set

$$\mathrm{Weight}(P) := \prod_{i=1}^{k} \frac{1}{n_i - 1}. \qquad (5)$$

Note that $n_i \geq 2$ because there is a path P passing through the node; and $\mathrm{Weight}(P) = 1$ if P consists of a single edge. The equation (5) is intended to degrade long paths which pass through several high-valency nodes. We use a random walk algorithm to sample paths such that the expected times a path is sampled equals its weight. As a result, the sampled path lengths range from 1 to 19, average 2.1, with an exponential tail. We convert all words which are sampled less than 1000 times to `*UNKNOWN*/POS`, and all prepositions occurring less than 10000 times to an *UNKNOWN* field. As a result, we obtain a vocabulary of 109k words and 211 field names.

Using the sampled paths, vectors and matrices are trained as in Section 4 (vecDCS). The vector dimension is set to $d = 250$. We compare with three baselines: (i) all matrices are fixed to identity ("no matrix"), in order to investigate the effects of meaning changes caused by syntactic-semantic roles and prepositions; (ii) the regularizer enforcing M_{N}^{-1} to be actually the inverse matrix of M_{N} is set to $\gamma = 0$ ("no inverse"), in order to investigate the effects of a semantically motivated constraint; and (iii) applying the same training scheme to UD trees directly, by modeling UD relations as matrices ("vecUD"). In this case, one edge is assigned one UD relation rel, so we implement the transfor-

	AN	NN	VO	SVO	GS11	GS12
vecDCS	**0.51**	**0.49**	**0.41**	**0.62**	0.29	0.33
-no matrix	**0.52**	0.46	**0.42**	**0.62**	0.29	0.33
-no inverse	0.47	0.43	0.38	0.58	0.28	0.33
vecUD	0.44	0.46	**0.41**	0.58	0.25	0.25
GloVe	0.41	0.47	**0.41**	0.60	0.23	0.17
Grefenstette and Sadrzadeh (2011)	-	-	-	-	0.21	-
Blacoe and Lapata (2012):RAE	0.31	0.30	0.28	-	-	-
Grefenstette (2013a)	-	-	-	-	-	0.27
Paperno et al. (2014)	-	-	-	-	-	**0.36**
Hashimoto et al. (2014):Wadd_{nl}	0.48	0.40	0.39		0.34	-
Kartsaklis and Sadrzadeh (2014)	-	-	-	0.43	**0.41**	-

Table 3: Spearman's ρ on phrase similarity

mation from child to parent by M_{rel}, and from parent to child by M_{rel}^{-1}. The same hyper-parameters are used to train vecUD. By comparing vecDCS with vecUD we investigate if applying the semantics framework of DCS makes any difference. Additionally, we compare with the GloVe (6B, 300d) vector[5] (Pennington et al., 2014). Norms of all word vectors are normalized to 1 and Frobenius norms of all matrices are normalized to $\sqrt{d}$.

5.1 Qualitative Analysis

We observe several special properties of the vectors and matrices trained by our model.

Words are clustered by POS In terms of cosine similarity, word vectors trained by vecDCS and vecUD are clustered by POS tags, probably due to their interactions with matrices during training. This is in contrast to the vectors trained by GloVe or "no matrix" (Table 1).

Matrices show semantic regularity Matrices learned for ARG, SUBJ and COMP are exactly orthogonal, and some most frequent prepositions[6] are remarkably close. For these matrices, the corresponding M^{-1} also exactly converge to their inverse. It suggests regularities in the semantic space, especially because orthogonal matrices preserve cosine similarity – if M_{N} is orthogonal, two words x, y and their projections $\pi_{\mathsf{N}}(\mathtt{x})$, $\pi_{\mathsf{N}}(\mathtt{y})$ will have the same similarity measure, which is semantically reasonable. In contrast, matrices trained by vecUD are only orthogonal for three UD relations, namely conj, dep and appos.

Words transformed by matrices To illustrate the matrices trained by vecDCS, we start from the query vectors of two words, house and learn, applying different matrices to them, and show the 10 answer vectors of the highest dot products (Tabel 2). These are the lists of likely words which: take *house* as a subject, take *house* as a complement, fills into "__ *in house*", serve as a subject of *learn*, serve as a complement of *learn*, and fills into "*learn about* __", respectively. As the table shows, matrices in vecDCS are appropriately learned to map word vectors to their syntactic-semantic roles.

5.2 Phrase Similarity

To test if vecDCS has the composition ability to calculate similar things as similar vectors, we conduct evaluation on a wide range of phrase similarity tasks. In these tasks, a system calculates similarity scores for pairs of phrases, and the performance is evaluated as its correlation with human annotators, measured by Spearman's ρ.

Datasets Mitchell and Lapata (2010) create datasets[7] for pairs of three types of two-word phrases: adjective-nouns (AN) (e.g. "*black hair*" and "*dark eye*"), compound nouns (NN) (e.g. "*tax charge*" and "*interest rate*") and verb-objects (VO) (e.g. "*fight war*" and "*win battle*"). Each dataset consists of 108 pairs and each pair is annotated by 18 humans (i.e., 1,944 scores in total). Similarity scores are integers ranging from 1 to 7. Another dataset[8] is created by extending VO to Subject-Verb-Object (SVO), and then assessing similarities by crowd sourcing (Kartsaklis and Sadrzadeh, 2014). The dataset GS11 created by Grefenstette and Sadrzadeh (2011) (100 pairs, 25 annotators) is also of the form SVO, but in each pair only the verbs are different (e.g. "*man pro-*

[5] http://nlp.stanford.edu/projects/glove/

[6] of, in, to, for, with, on, as, at, from

[7] http://homepages.inf.ed.ac.uk/s0453356/

[8] http://www.cs.ox.ac.uk/activities/compdistmeaning/

Message-Topic(e_1, e_2)	*It is a monthly* [*report*]$_1$ *providing* [*opinion*]$_2$ *and advice on current United States government contract issues.*
Message-Topic(e_1, e_2)	*The* [*report*]$_1$ *gives an account of the silvicultural* [*work*]$_2$ *done in Africa, Asia, Australia, South American and the Caribbean.*
Message-Topic(e_1, e_2)	*NUS today responded to the Government's* [*announcement*]$_1$ *of the long-awaited* [*review*]$_2$ *of university funding.*
Component-Whole(e_2, e_1)	*The* [*review*]$_1$ *published political* [*commentary*]$_2$ *and opinion, but even more than that.*
Message-Topic(e_1, e_2)	*It is a 2004* [*book*]$_1$ *criticizing the political and linguistic* [*writings*]$_2$ *of Noam Chomsky.*

Table 4: Similar training instances clustered by cosine similarities between features

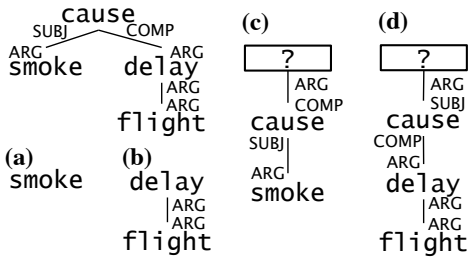


Figure 4: For "[*smoke*]$_1$ *cause flight* [*delay*]$_2$", we construct (a)(b) from subtrees, and (c)(d) from re-rooted trees, to form 4 query vectors as feature.

vide/supply money"). The dataset GS12 described in Grefenstette (2013a) (194 pairs, 50 annotators) is of the form Adjective-Noun-Verb-Adjective-Noun (e.g. "*local family run/move small hotel*"), where only verbs are different in each pair.

Our method We calculate the cosine similarity of query vectors corresponding to phrases. For example, the query vector for "*fight war*" is calculated as $\mathbf{v}_{war} M_{\mathsf{ARG}} M_{\mathsf{COMP}}^{-1} + \mathbf{v}_{fight}$. For vecUD we use M_{nsubj} and M_{dobj} instead of M_{SUBJ} and M_{COMP}, respectively. For GloVe we use additive compositions.

Results As shown in Table 3, vecDCS is competitive on AN, NN, VO, SVO and GS12, consistently outperforming "no inverse", vecUD and GloVe, showing strong compositionality. The weakness of "no inverse" suggests that relaxing the constraint of inverse matrices may hurt compositionaly, though our preliminary examination on word similarities did not find any difference. The GS11 dataset appears to favor models that can learn from interactions between the subject and object arguments, such as the non-linear model Wadd_{nl} in Hashimoto et al. (2014) and the entanglement model in Kartsaklis and Sadrzadeh (2014). However, these models do not show particular advantages on other datasets. The recursive autoencoder (RAE) proposed in Socher et al. (2011) shares an aspect with vecDCS as to construct meanings from parse trees. It is tested by Blacoe and Lapata (2012) for compositionality, where vecDCS appears to be better. Nevertheless, we note that "no matrix" performs as good as vecDCS, suggesting that meaning changes caused by syntactic-semantic roles might not be major factors in these datasets, because the syntactic-semantic relations are all fixed in each dataset.

vecDCS	81.2
-no matrix	69.2
-no inverse	79.7
vecUD	69.2
GloVe	74.1
Socher et al. (2012)	79.1
+3 features	82.4
dos Santos et al. (2015)	84.1
Xu et al. (2015)	**85.6**

Table 5: F1 on relation classification

5.3 Relation Classification

In a relation classification task, the relation between two words in a sentence needs to be classified; we expect vecDCS to perform better than "no matrix" on this task because vecDCS can distinguish the different syntactic-semantic roles of the two slots the two words fit in. We confirm this conjecture in this section.

Dataset We use the dataset of SemEval-2010 Task 8 (Hendrickx et al., 2009), in which 9 directed relations (e.g. *Cause-Effect*) and 1 undirected relation *Other* are annotated, 8,000 instances for training and 2,717 for test. Performance is measured by the 9-class direction-aware Macro-F1 score excluding *Other* class.

Our method For any sentence with two words marked as e_1 and e_2, we construct the DCS tree of the sentence, and take the subtree T rooted at the common ancestor of e_1 and e_2. We construct four vectors from T, namely: the query vector for the subtree rooted at e_1 (resp. e_2), and the query vector of the DCS tree obtained from T by re-rooting it at e_1 (resp. e_2) (Figure 4). The four vectors are normalized and concatenated to form the only feature used to train a classifier. For vecUD, we use the corresponding vectors calculated from UD trees. For GloVe, we use the word vector of e_1 (resp. e_2), and the sum of vectors of all words within the span $[e_1, e_2)$ (resp. $(e_1, e_2]$) as

"banned drugs"	*"banned movies"*	*"banned books"*
drug/N	*bratz/N*	*publish/N*
marijuana/N	*porn/N*	*unfair/N*
cannabis/N	*indecent/N*	*obscene/N*
trafficking/N	*blockbuster/N*	*samizdat/N*
thalidomide/N	*movie/N*	*book/N*
smoking/N	*idiots/N*	*responsum/N*
narcotic/N	*blacklist/N*	*illegal/N*
botox/N	*grindhouse/N*	*reclaiming/N*
doping/N	*doraemon/N*	*redbook/N*

Table 6: Answers for composed query vectors

vecDCS	50
-no matrix	**60**
-no inverse	46
vecUD	31
N-gram (Various)	39-41
Zweig et al. (2012)	52
Mnih and Teh (2012)	55
Gubbins and Vlachos (2013)	50
Mikolov et al. (2013a)	55

Table 7: Accuracy (%) on sentence completion

the four vectors. Classifier is SVM[9] with RBF kernel, $C = 2$ and $\Gamma = 0.25$. The hyper-parameters are selected by 5-fold cross validation.

Results VecDCS outperforms baselines on relation classification (Table 5). It makes 16 errors in misclassifying the direction of a relation, as compared to 144 such errors made by "no matrix", 23 by "no inverse", 30 by vecUD, and 161 by GloVe. This suggests that models with syntactic-semantic transformations (i.e. vecDCS, "no inverse", and vecUD) are indeed good at distinguishing the different roles played by e_1 and e_2. VecDCS scores moderately lower than the state-of-the-art (Xu et al., 2015), however we note that these results are achieved by adding additional features and training task-specific neural networks (dos Santos et al., 2015; Xu et al., 2015). Our method only uses features constructed from unlabeled corpora. From this point of view, it is comparable to the MV-RNN model (without features) in Socher et al. (2012), and vecDCS actually does better. Table 4 shows an example of clustered training instances as assessed by cosine similarities between their features. It suggests that the features used in our method can actually cluster similar relations.

5.4 Sentence Completion

If vecDCS can compose query vectors of DCS trees, one should be able to "execute" the vectors to get a set of answers, as the original DCS trees can do. This is done by taking dot products with answer vectors and then ranking the answers. Examples are shown in Table 6. Since query vectors and answer vectors are trained from unlabeled corpora, we can only obtain a coarse-grained candidate list. However, it is noteworthy that despite a common word "*banned*" shared by the phrases, their answer lists are largely different, suggesting that composition actually can be done. Moreover, some words indeed answer the queries (e.g. *Thalidomide* for "banned drugs" and *Samizdat* for "*banned books*").

Quantitatively, we evaluate this utility of executing queries on the sentence completion task. In this task, a sentence is presented with a blank that need to be filled in. Five possible words are given as options for each blank, and a system needs to choose the correct one. The task can be viewed as a coarse-grained question answering or an evaluation for language models (Zweig et al., 2012). We use the MSR sentence completion dataset[10] which consists of 1,040 test questions and a corpus for training language models. We train vecDCS on this corpus and use it for evaluation.

Results As shown in Table 7, vecDCS scores better than the N-gram model and demonstrates promising performance. However, to our surprise, "no matrix" shows an even better result which is the new state-of-the-art. Here we might be facing the same problem as in the phrase similarity task (Section 5.2); namely, all choices in a question fill into the same blank and the same syntactic-semantic role, so the transforming matrices in vecDCS might not be able to distinguish different choices; on the other hand, vecDCS would suffer more from parsing and POS-tagging errors. Nonetheless, we believe the result by "no matrix" reveals a new horizon of sentence completion, and suggests that composing semantic vectors according to DCS trees could be a promising direction.

6 Discussion

We have demonstrated a way to link a vector composition model to a formal semantics, combining the strength of vector representations to calculate phrase similarities, and the strength of formal semantics to build up structured queries. In this section, we discuss several lines of previous research related to this work.

[9] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

[10] http://research.microsoft.com/en-us/projects/scc/

Logic and Distributional Semantics Logic is necessary for implementing the functional aspects of meaning and organizing knowledge in a structured and unambiguous way. In contrast, distributional semantics provides an elegant methodology for assessing semantic similarity and is well suited for learning from data. There have been repeated calls for combining the strength of these two approaches (Coecke et al., 2010; Baroni et al., 2014; Liang and Potts, 2015), and several systems (Lewis and Steedman, 2013; Beltagy et al., 2014; Tian et al., 2014) have contributed to this direction. In the remarkable work by Beltagy et al. (to appear), word and phrase similarities are explicitly transformed to weighted logical rules that are used in a probabilistic inference framework. However, this approach requires considerable amount of engineering, including the generation of rule candidates (e.g. by aligning sentence fragments), converting distributional similarities to weights, and efficiently handling the rules and inference. What if the distributional representations are equipped with a logical interface, such that the inference can be realized by simple vector calculations? We have shown it possible to realize semantic composition; we believe this may lead to significant simplification of the system design for combining logic and distributional semantics.

Compositional Distributional Models There has been active exploration on how to combine word vectors such that adequate phrase/sentence similarities can be assessed (Mitchell and Lapata, 2010, *inter alia*), and there is nothing new in using matrices to model changes of meanings. However, previous model designs mostly rely on linguistic intuitions (Paperno et al., 2014, *inter alia*), whereas our model has an exact logic interpretation. Furthermore, by using additive composition we enjoy a learning guarantee (Tian et al., 2015).

Vector-based Logic Models This work also shares the spirit with Grefenstette (2013b) and Rocktaeschel et al. (2014), in exploring vector calculations that realize logic operations. However, the previous works did not specify how to integrate contextual distributional information, which is necessary for calculating semantic similarity.

Formal Semantics Our model implements a fragment of logic capable of semantic composition, largely due to the simple framework of Dependency-based Compositional Semantics (Liang et al., 2013). It fits in a long tradition of logic-based semantics (Montague, 1970; Dowty et al., 1981; Kamp and Reyle, 1993), with extensive studies on extracting semantics from syntactic representations such as HPSG (Copestake et al., 2001; Copestake et al., 2005) and CCG (Baldridge and Kruijff, 2002; Bos et al., 2004; Steedman, 2012; Artzi et al., 2015; Mineshima et al., 2015).

Logic for Natural Language Inference The pursue of a logic more suitable for natural language inference is also not new. For example, MacCartney and Manning (2008) has implemented a model of natural logic (Lakoff, 1970). We would not reach the current formalization of logic of DCS without reading the work by Calvanese et al. (1998), which is an elegant formalization of database semantics in description logic.

Semantic Parsing DCS-related representations have been actively used in semantic parsing and we see potential in applying our model. For example, Berant and Liang (2014) convert λ-DCS queries to canonical utterances and assess paraphrases at the surface level; an alternative could be using vector-based DCS to bring distributional similarity directly into calculation of denotations. We also borrow ideas from previous work, for example our training scheme is similar to Guu et al. (2015) in using paths and composition of matrices, and our method is similar to Poon and Domingos (2009) in building structured knowledge from clustering syntactic parse of unlabeled data.

Further Applications Regarding the usability of distributional representations learned by our model, a strong point is that the representation takes into account syntactic/structural information of context. Unlike several previous models (Padó and Lapata, 2007; Levy and Goldberg, 2014; Pham et al., 2015), our approach learns matrices at the same time that can extract the information according to different syntactic-semantic roles. A related application is selectional preference (Baroni and Lenci, 2010; Lenci, 2011; Van de Cruys, 2014), wherein our model might has potential for smoothly handling composition.

Reproducibility Find our code at `https://github.com/tianran/vecdcs`

Acknowledgments This work was supported by CREST, JST. We thank the anonymous reviewers for their valuable comments.

References

Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage ccg semantic parsing with amr. In *Proceedings of EMNLP*.

Jason Baldridge and Geert-Jan Kruijff. 2002. Coupling ccg and hybrid logic dependency semantics. In *Proceedings of ACL*.

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

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*.

Marco Baroni, Raffaella Bernardi, and Roberto Zamparelli. 2014. Frege in space: A program for compositional distributional semantics. *Linguistic Issues in Language Technology*, 9(6).

Islam Beltagy, Katrin Erk, and Raymond Mooney. 2014. Probabilistic soft logic for semantic textual similarity. In *Proceedings of ACL*.

Islam Beltagy, Stephen Roller, Pengxiang Cheng, Katrin Erk, and Raymond J. Mooney. to appear. Representing meaning with a combination of logical form and vectors. *Computational Linguistics*, special issue on formal distributional semantics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of ACL*.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP-CoNLL*.

Johan Bos, Stephen Clark, Mark Steedman, James R. Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a ccg parser. In *Proceedings of ICCL*.

Léon Bottou. 2012. Stochastic gradient descent tricks. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade*. Springer, Berlin.

Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. 1998. On the decidability of query containment under constraints. In *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS98)*.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*.

Ann Copestake, Alex Lascarides, and Dan Flickinger. 2001. An algebra for semantic construction in constraint-based grammars. In *Proceedings of ACL*.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A. Sag. 2005. Minimal recursion semantics: An introduction. *Research on Language and Computation*, 3(2-3).

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL-IJCNLP*.

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Springer Netherlands.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*.

Edward Grefenstette. 2013a. *Category-Theoretic Quantitative Compositional Distributional Models of Natural Language Semantics*. PhD thesis.

Edward Grefenstette. 2013b. Towards a formal distributional semantics: Simulating logical calculi with tensors. In *Proceedings of *SEM*.

Joseph Gubbins and Andreas Vlachos. 2013. Dependency language models for sentence completion. In *Proceedings of EMNLP*.

Michael U. Gutmann and Aapo Hyvärinen. 2012. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13(1).

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of EMNLP*.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of EMNLP*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic*. Springer Netherlands.

Dimitri Kartsaklis and Mehrnoosh Sadrzadeh. 2014. A study of entanglement in a categorical framework of natural language. In *Proceedings of the 11th Workshop on Quantum Physics and Logic (QPL)*.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in NIPS*.

George Lakoff. 1970. Linguistics and natural logic. *Synthese*, 22(1-2).

Alessandro Lenci. 2011. Composing and updating verb argument expectations: A distributional semantic model. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*.

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of ACL*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of ACL*, 3.

Mike Lewis and Mark Steedman. 2013. Combined distributional and logical semantics. *Transactions of ACL*, 1.

Percy Liang and Christopher Potts. 2015. Bringing machine learning and compositional semantics together. *Annual Review of Linguistics*, 1.

Percy Liang, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2).

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of Coling*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proceedings ACL*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*.

Koji Mineshima, Pascual Martínez-Gómez, Yusuke Miyao, and Daisuke Bekki. 2015. Higher-order logical inference with compositional semantics. In *Proceedings of EMNLP*.

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

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *In Proceedings of ICML*.

Richard Montague. 1970. Universal grammar. *Theoria*, 36.

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

Denis Paperno, Nghia The Pham, and Marco Baroni. 2014. A practical and linguistically-motivated approach to compositional distributional semantics. In *Proceedings of ACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Nghia The Pham, Germán Kruszewski, Angeliki Lazaridou, and Marco Baroni. 2015. Jointly optimizing word representations for lexical and sentential tasks with the c-phrase model. In *Proceedings of ACL*.

Hoifung Poon and Pedro Domingos. 2009. Unsupervised semantic parsing. In *Proceedings of EMNLP*.

Tim Rocktaeschel, Matko Bosnjak, Sameer Singh, and Sebastian Riedel. 2014. Low-dimensional embeddings of logic. In *ACL Workshop on Semantic Parsing (SP'14)*.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in NIPS*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*.

Mark Steedman. 2012. *Taking Scope - The Natural Semantics of Quantifiers*. MIT Press.

Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. 2014. Logical inference on dependency-based compositional semantics. In *Proceedings of ACL*.

Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2015. The mechanism of additive composition. arXiv:1511.08407.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37(1).

Tim Van de Cruys. 2014. A neural network approach to selectional preference acquisition. In *Proceedings of EMNLP*.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of EMNLP*.

Geoffrey Zweig, John C. Platt, Christopher Meek, Christopher J.C. Burges, Ainur Yessenalina, and Qiang Liu. 2012. Computational approaches to sentence completion. In *Proceedings of ACL*.

Inner Attention based Recurrent Neural Networks for Answer Selection

Bingning Wang, Kang Liu, Jun Zhao
National Laboratory of Pattern Recognition, Institute of Automation
Chinese Academy of Sciences, Beijing, China
{bingning.wang, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Attention based recurrent neural networks have shown advantages in representing natural language sentences (Hermann et al., 2015; Rocktäschel et al., 2015; Tan et al., 2015). Based on recurrent neural networks (RNN), external attention information was added to hidden representations to get an attentive sentence representation. Despite the improvement over non-attentive models, the attention mechanism under RNN is not well studied. In this work, we analyze the deficiency of traditional attention based RNN models quantitatively and qualitatively. Then we present three new RNN models that add attention information before RNN hidden representation, which shows advantage in representing sentence and achieves new state-of-art results in answer selection task.

1 Introduction

Answer selection (AS) is a crucial subtask of the open domain question answering (QA) problem. Given a question, the goal is to choose the answer from a set of pre-selected sentences (Heilman and Smith, 2010; Yao et al., 2013). Traditional AS models are based on lexical features such as parsing tree edit distance. Neural networks based models are proposed to represent the meaning of a sentence in a vector space and then compare the question and answer candidates in this hidden space (Wang and Nyberg, 2015; Feng et al., 2015), which have shown great success in AS. However, these models represent the question and sentence separately, which may ignore the information subject to the question when representing the answer. For example, given a candidate answer:

Michael Jordan abruptly retired from Chicago Bulls before the beginning of the 1993-94 NBA season to pursue a career in baseball.

For a question: **When did Michael Jordan retired from NBA?** we should focus on *the beginning of the 1993-94* in the sentence; however, when we were asked: **Which sports does Michael Jordan participates after his retirement from NBA?** we should pay more attention to *pursue a career in baseball*.

Recent years, attention based models are proposed in light of this purpose and have shown great success in many NLP tasks such as machine translation (Bahdanau et al., 2014; Sutskever et al., 2014), question answering (Sukhbaatar et al., 2015) and recognizing textual entailments (Rocktäschel et al., 2015). When building the representation of a sentence, some attention information is added to the hidden state. For example, in attention based recurrent neural networks models (Bahdanau et al., 2014) each time-step hidden representation is weighted by attention. Inspired by the attention mechanism, some attention-based RNN answer selection models have been proposed (Tan et al., 2015) in which the attention when computing answer representation is from question representation.

However, in the RNN architecture, at each time step a word is added and the hidden state is updated recurrently, so those hidden states near the end of the sentence are expected to capture more information[1]. Consequently, after adding the attention information to the time sequence hidden representations, the near-the-end hidden variables will be more attended due to their comparatively abundant semantic accumulation, which may result in a biased attentive weight towards the later coming words in RNN.

In this work, we analyze this attention bias

[1] so in many previous RNN-based model use the last hidden variable as the whole sentence representation

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1288–1297,
Berlin, Germany, August 7-12, 2016.

problem qualitatively and quantitatively, and then propose three new models to solve this problem. Different from previous attention based RNN models in which attention information is added after RNN computation, we add the attention before computing the sentence representation. Concretely, the first one uses the question attention to adjust word representation (i.e. word embedding) in the answer directly, and then we use RNN to model the attentive word sequence. However, this model attends a sentence word by word which may ignore the relation between words. For example, if we were asked: *what is his favorite food?* one answer candidate is: *He likes hot dog best*. *hot* or *dog* may be not relate to the question by itself, but they are informative as a whole in the context. So we propose the second model in which every word representation in answer is impacted by not only question attention but also the context representation of the word (i.e. the last hidden state). In our last model, inspired by previous work on adding gate into inner activation of RNN to control the long and short term information flow, we embed the attention to the inner activation gate of RNN to influence the computation of RNN hidden representation. In addition, inspired by recent work called Occam's Gate in which the activation of input units are penalized to be as less as possible, we add regulation to the summation of the attention weights to impose sparsity.

Overall, in this work we make three contributions: (1) We analyze the attention bias problem in traditional attention based RNN models. (2) We propose three inner attention based RNN models and achieve new state-of-the-art results in answer selection. (3) We use Occam's Razor to regulate the attention weights which shows advantage in long sentence representation.

2 Related Work

Recent years, many deep learning framework has been developed to model the text in a vector space, and then use the embedded representations in this space for machine learning tasks. There are many neural networks architectures for this representation such as convolutional neural networks(Yin et al., 2015), recursive neural networks(Socher et al., 2013) and recurrent neural networks(Mikolov et al., 2011). In this work we propose **I**nner **A**ttention based **RNN** (**IARNN**) for answer selection, and there are two main works which we are related to.

2.1 Attention based Models

Many recent works show that attention techniques can improve the performance of machine learning models (Mnih et al., 2014; Zheng et al., 2015). In attention based models, one representation is built with attention (or supervision) from other representation. Weston et al (2014) propose a neural networks based model called Memory Networks which uses an external memory to store the knowledge and the memory are read and written on the fly with respect to the attention, and these attentive memory are combined for inference. Since then, many variants have been proposed to solve question answering problems (Sukhbaatar et al., 2015; Kumar et al., 2015). Hermann (2015) and many other researchers (Tan et al., 2015; Rocktäschel et al., 2015) try to introduce the attention mechanism into the LSTM-RNN architecture. RNN models the input sequence word-by-word and updates its hidden variable recurrently. Compared with CNN, RNN is more capable of exploiting long-distance sequential information. In attention based RNN models, after computing each time step hidden representation, attention information is added to weight each hidden representation, then the hidden states are combined with respect to that weight to obtain the sentence (or document) representation. Commonly there are two ways to get attention from source sentence, either by the whole sentence representation (which they call *attentive*) or word by word attention (called *impatient*).

2.2 Answer Selection

Answer selection is a sub-task of QA and many other tasks such as machine comprehension. Given a question and a set of candidate sentences, one should choose the best sentence from a candidate sentence set that can answer the question. Previous works usually stuck in employing feature engineering, linguistic tools, or external resources. For example, Yih et al. (2013) use semantic features from WordNet to enhance lexical features. Wang and Manning (2007) try to compare the question and answer sentence by their syntactical matching in parse trees. Heilman and Smith (Heilman and Smith, 2010) try to fulfill the matching using minimal edit sequences between their dependency parse trees. Severyn and Moschitti (2013) automate the extraction of discriminative tree-edit features over parsing trees.

While these methods show effectiveness, they might suffer from the availability of additional resources and errors of many NLP tools such as dependency parsing. Recently there are many works use deep learning architecture to represent the question and answer in a same hidden space, and then the task can be converted into a classification or learning-to-rank problem (Feng et al., 2015; Wang and Nyberg, 2015). With the development of attention mechanism, Tan et.al(2015) propose an attention-based RNN models which introduce question attention to answer representation.

3 Traditional Attention based RNN Models and Their Deficiency

The attention-based models introduce the attention information into the representation process. In answer selection, given a question $Q = \{q_1, q_2, q_3, ..., q_n\}$ where q_i is i-th word, n is the question length, we can compute its representation in RNN architecture as follows:

$$\begin{aligned} \mathbf{X} &= \mathbf{D}[q_1, q_2, ..., q_n] \\ \mathbf{h}_t &= \sigma(\mathbf{W}_{ih}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \\ \mathbf{y}_t &= \sigma(\mathbf{W}_{ho}\mathbf{h}_t + \mathbf{b}_o) \end{aligned} \tag{1}$$

where $\mathbf{D}$ is an embedding matrix that projects word to its embedding space in R^d; $\mathbf{W}_{ih}$, $\mathbf{W}_{hh}$, $\mathbf{W}_{ho}$ are weight matrices and $\mathbf{b}_h$, $\mathbf{b}_o$ are bias vectors; σ is active function such as $tanh$. Usually we can ignore the output variables and use the hidden variables. After recurrent process, the last hidden variable $\mathbf{h}_n$ or all hidden states average $\frac{1}{n}\sum_{t=1}^{n}\mathbf{h}_t$ is adopted as the question representation $\mathbf{r}_q$.

When modeling the candidate answer sentence with length m:$S = \{s_1, s_2, s_3, ..., s_m\}$ in attention based RNN model,

instead of using the last hidden state or average hidden states, we use attentive hidden states that are weighted by $\mathbf{r}_q$:

$$\begin{aligned} \mathbf{H}_a &= [\mathbf{h}_a(1), \mathbf{h}_a(2), ..., \mathbf{h}_a(m)] \\ s_t &\propto f_{attention}(\mathbf{r}_q, \mathbf{h}_a(t)) \\ \tilde{\mathbf{h}}_a(t) &= \mathbf{h}_a(t)s_t \\ \mathbf{r}_a &= \sum_{t=1}^{m}\tilde{\mathbf{h}}_a(t) \end{aligned} \tag{2}$$

where $\mathbf{h}_a(t)$ is hidden state of the answer at time t. In many previous work (Hermann et al., 2015; Rocktäschel et al., 2015; Tan et al., 2015), the attention function $f_{attention}$ was computed as:

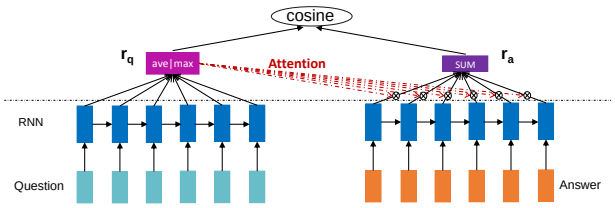


Figure 1: Traditional attention based RNN answer selection model. Dark blue rectangles represent hidden virable, $\otimes$ means gate opperation.

$$\begin{aligned} \mathbf{m}(t) &= tanh(\mathbf{W}_{hm}\mathbf{h}_a(t) + \mathbf{W}_{qm}\mathbf{r}_q) \\ f_{attention}(\mathbf{r}_q, \mathbf{h}_a(t)) &= exp(\mathbf{w}_{ms}^T\mathbf{m}(t)) \end{aligned} \tag{3}$$

$\mathbf{W}_{hm}$ and $\mathbf{W}_{qm}$ are attentive weight matrices and $\mathbf{w}_{ms}$ is attentive weight vector. So we can expect that the candidate answer sentence representation $\mathbf{r}_a$ may be represented in a question-guided way: when its hidden state $\mathbf{h}_a(t)$ is irrelevant to the question (determined by attention weight s_t), it will take less part in the final representation; but when this hidden state is relavent to the question, it will contribute more in representing $\mathbf{r}_a$. We call this type of attention based RNN model OARNN which stands for *Outer Attention based RNN* models because this kind of model adds attention information outside the RNN hidden representation computing process. An illustration of traditional attention-based RNN model is in Figure 1.

However, we know in the RNN architecture, the input words are processed in time sequence and the hidden states are updated recurrently, so the current hidden state $\mathbf{h}_t$ is supposed to contain all the information up to time t, when we add question attention information, aiming at finding the useful part of the sentence, these near-the-end hidden states are prone to be selected because they contains much more information about the whole sentence. In other word, if the question pays attention to the hidden states at time t , then it should also pay attention to those hidden states after t (i.e $\{\mathbf{h}_{t'}|t' > t\}$) as they contain the information at least as much as $\mathbf{h}_t$, but in answer selection for a specific candidate answer, the useful parts to answer the question may be located anywhere in a sentence, so the attention should also distribute uniformly around the sentence. Traditional attention-based RNN models under *attention after representation* mechanism may cause the attention to bias towards the later coming hidden states. We will analyze this attention bias problem quantita-

tively in the experiments.

4 Inner Attention based Recurrent Neural Networks

In order to solve the attention bias problem, we propose an intuition:

Attention before representation

Instead of adding attention information after encoding the answer by RNN, we add attention before computing the RNN hidden representations. Based on this intuition, we propose three inner attention based RNN models detailed below.

4.1 IARNN-WORD

As attention mechanism aims at finding useful part of a sentence, the first model applies the above intuition directly. Instead of using the original answer words to the RNN model, we weight the words representation according to question attention as follows:

$$\begin{aligned} \alpha_t &= \sigma(\mathbf{r}_q^T \mathbf{M}_{qi} \mathbf{x}_t) \\ \tilde{\mathbf{x}}_t &= \alpha_t * \mathbf{x}_t \end{aligned} \tag{4}$$

where $\mathbf{M}_{qi}$ is an attention matrix to transform a question representaion into the word embedding space. Then we use the dot value to determine the question attention strength, σ is sigmoid function to normalize the weight α_t between 0 and 1.

The above attention process can be understood as sentence distillation where the input words are distilled (or filtered) by question attention. Then, we can represent the whole sentence based on this distilled input using traditional RNN model. In this work, we use GRU instead of LSTM as building block for RNN because it has shown advantages in many tasks and has comparatively less parameter(Jozefowicz et al., 2015) which is formulated as follows:

$$\begin{aligned} \mathbf{z}_t &= \sigma(\mathbf{W}_{xz}\tilde{\mathbf{x}}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1}) \\ \mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\tilde{\mathbf{x}}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1}) \\ \tilde{\mathbf{h}}_t &= tanh(\mathbf{W}_{xh}\tilde{\mathbf{x}}_t + \mathbf{W}_{hh}(\mathbf{f}_t \odot \mathbf{h}_{t-1})) \\ \mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t \end{aligned} \tag{5}$$

where $\mathbf{W}_{xz}, \mathbf{W}_{hz}, \mathbf{W}_{xf}, \mathbf{W}_{hh}, \mathbf{W}_{xh}$ are weight matrices and $\odot$ stands for element-wise multiplication. Finally, we get candidate answer representation by average pooling all the hidden state $\mathbf{h}_t$. we call this model IARNN-WORD as the attention is paid to the original input words. This model is shown in Figure 2.

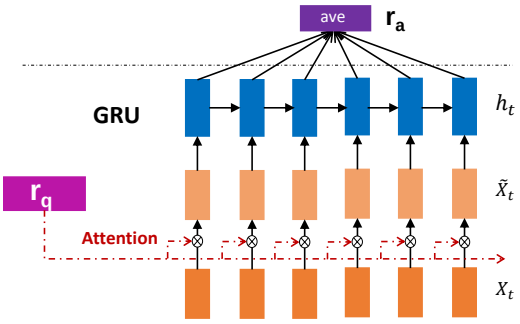


Figure 2: IARNN-WORD architecture. $\mathbf{r}_q$ is question representation.

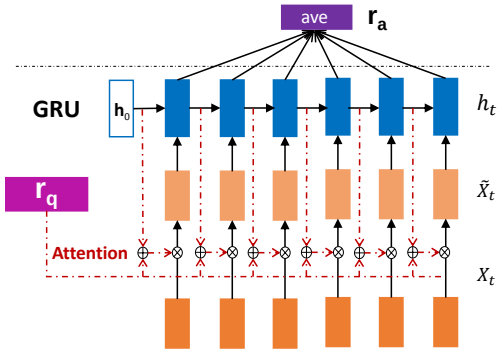


Figure 3: IARNN-CONTEXT architecture for building candidate answer sentence representation. $\mathbf{h}_0$ is added for completeness.

4.2 IARNN-CONTEXT

IABRNN-WORD attend input word embedding directly. However, the answer sentence may consist of consecutive words that are related to the question, and a word may be irrelevant to question by itself but relevant in the context of answer sentence.

So the above word by word attention mechanism may not capture the relationship between multiple words. In order to import contextual information into attention process, we modify the attention weights in Equation 4 with additional context information:

$$\begin{aligned} \mathbf{w}_C(t) &= \mathbf{M}_{hc}\mathbf{h}_{t-1} + \mathbf{M}_{qc}\mathbf{r}_q \\ \alpha_C^t &= \sigma(\mathbf{w}_C^T(t)\mathbf{x}_t) \\ \tilde{\mathbf{x}}_t &= \alpha_C^t * \mathbf{x}_t \end{aligned} \tag{6}$$

where we use $\mathbf{h}_{t-1}$ as context, $\mathbf{M}_{hc}$ and $\mathbf{M}_{qc}$ are attention weight matrices, $\mathbf{w}_C(t)$ is the attention representation which consists of both question and word context information. This additional context attention endows our model to capture relevant part in longer text span. We show this model in Figure 3.

4.3 IARNN-GATE

Inspired by the previous work of LSTM (Hochreiter and Schmidhuber, 1997) on solving the gradient exploding problem in RNN and recent work on building distributed word representation with topic information(Ghosh et al., 2016), instead of adding attention information to the original input, we can apply attention deeper to the GRU inner activation (i.e $\mathbf{z}_t$ and $\mathbf{f}_t$). Because these inner activation units control the flow of the information within the hidden stage and enables information to pass long distance in a sentence, we add attention information to these active gates to influence the hidden representation as follows:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_{xz}\mathbf{x}_t + \mathbf{W}_{hz}\mathbf{h}_{t-1} + \mathbf{M}_{qz}\mathbf{r}_q) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{M}_{qf}\mathbf{r}_q) \\
\tilde{\mathbf{h}}_t &= tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}(\mathbf{f}_t \odot \mathbf{h}_{t-1})) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t
\end{aligned} \tag{7}$$

where $\mathbf{M}_{qz}$ and $\mathbf{M}_{hz}$ are attention weight matrices. In this way, the update and forget units in GRU can focus on not only long and short term memory but also the attention information from the question. The architecture is shown in Figure 4.

4.4 IARNN-OCCAM

In answer selection, the answer sentence may only contain small number of words that are related to the question. In IARNN-WORD and IARNN-CONTEXT, we calculate each word attention weight without considering total weights. Similar with Raiman(2015) who adds regulation to the input gate, we punish the summation of the attention weights to enforce sparsity. This is an application of Occam's Razor: *Among the whole words set, we choose those with fewest number that can represent the sentence*. However, assigning a pre-defined hyper-parameter for this regulation[2] is not an ideal way because it punishes all question attention weights with same strength. For different questions there may be different number of snippets in candidate answer that are required. For example, when the question type is *When* or *Who*, answer sentence may only contains a little relavant words so we should impose more sparsity on the summation of the attention. But when the question type is *Why* or *How*, there may be much more words on the sentence that are relevant to the question so we should set the regulation value small accordingly. In this work, this attention regulation is added as follows: for the specific question Q_i and its representation $\mathbf{r}_q^i$, we use a vector $\mathbf{w}_{qp}$ to project it into scalar value n_p^i, and then we add it into the original objective J_i as follows:

$$\begin{aligned}
n_p^i &= \max\{\mathbf{w}_{qp}^T \mathbf{r}_q^i, \lambda_q\} \\
J_i^* &= J_i + n_p^i \sum_{t=1}^{mc} \alpha_t^i
\end{aligned} \tag{8}$$

where α_t^i is attention weights in Equation 4 and Equation 6. λ_q is a small positive hyper-parameter. It needs to mention that we do not regulate IARNN-GATE because the attention has been embedded to gate activation.

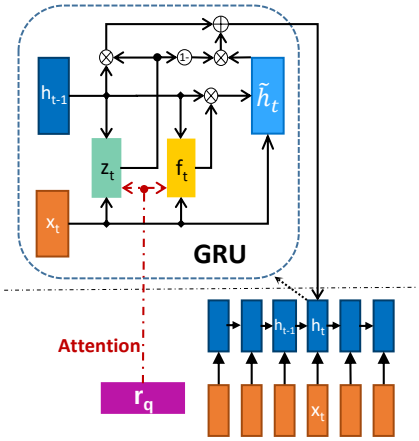


Figure 4: IABRNN-GATE architecture. We show one time step GRU inner state process within the blue dotted line.

5 Experiments

5.1 Quantify Traditional Attention based Model Bias Problem

In order to quantify the outer attention based RNN model's attention bias problem in Section 3, we build an outer attention based model similar with Tan (2015). First of all, for the question we build its representation by averaging its hidden states in LSTM, then we build the candidate answer sentence representation in an attentive way introduced in Section 3. Next we use the cosine similarity to compare question and answer representation similarity. Finally, we adopt max-margin hinge loss as objective:

$$\begin{aligned}
L = max\{0, M - cosine(\mathbf{r}_q, \mathbf{r}_{a_+}) \\
+ cosine(\mathbf{r}_q, \mathbf{r}_{a_-})\}
\end{aligned} \tag{9}$$

[2]For example, in many machine learning problem the original objective sometimes followed with a L_1 or L_2 regulation with hyper-parameter $\lambda 1$ or $\lambda 2$ to control the tradeoff between the original objective J and the sparsity criterion:$J^* = J + (\lambda 1 | \lambda 2) \sum(L_1 | L_2 norm)$

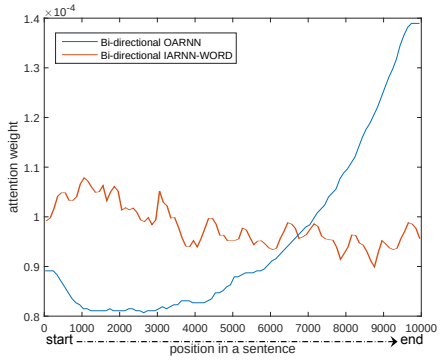


Figure 5: One directional OARNN attention distribution, the horizontal axis is position of word in a sentence that has been normalized from 1 to 10000.

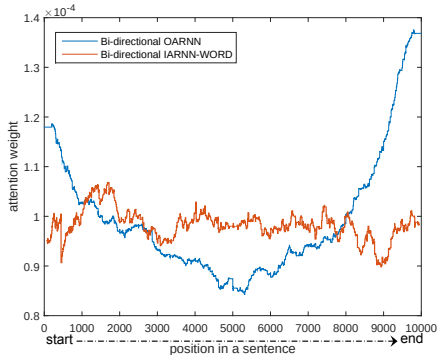


Figure 6: Bi-directional OARNN attention distribution, the horizontal axis is the postion of the word in a sentence that has been normalized from 1 to 10000.

where a_+ is ground truth answer candidate and a_- stands for negative one, the scalar M is a predefined margin. When training result saturates after 50 epoches, we get the attention weight distribution (i.e. s_q in Equation 2). The experiment is conducted on two answer selection datasets: WikiQA (Yang et al., 2015) and TrecQA (Wang et al., 2007). The normalized attention weights is reported in Figure 5.

However, the above model use only forward LSTM to build hidden state representation, the attention bias problem may attribute to the biased answer distribution: the useful part of the answer to the question sometimes may located at the end of the sentence. So we try OARNN in bidirectional architecture, where the forward LSTM and backward LSTM are concatenated for hidden representation, The bidirectional attention based LSTM attention distribution is shown in Figure 6.

Analysis: As is shown in Figure 5 and 6, for one-directional OARNN, as we move from beginning to the end in a sentence, the question attention gains continuously; when we use bidirectional OARNN, the hidden representations near two ends of a sentence get more attention. This is consistent with our assumption that for a hidden representation in RNN, the closer to the end of a sentence, the more attention it should drawn from question. But the relevant part may be located anywhere in a answer. As a result, when the sample size is large enough[3], the attention weight should be unformly distributed. The traditional *attention after representation* style RNN may suffer from the biased attention problem. Our IARNN models are free from this problem and distribute nearly uniform (orange line) in a sentence.

5.2 IARNN evaluation

Common Setup: We use the off-the-shelf 100-dimension word embeddings from word2vec[4], and initiate all weights and attention matrices by fixing their largest singular values to 1 (Pascanu et al., 2013). IARNN-OCCAM base regulation hyper-parameter λ_q is set to 0.05, we add L_2 penalty with a coefficient of 10^{-5} . Dropout (Srivastava et al., 2014) is further applied to every parameters with probability 30%. We use Adadelta(Zeiler, 2012) with $\rho = 0.90$ to update parameters.

We choose three datasets for evaluation: InsuranceQA, WikiQA and TREC-QA. These datasets contain questions from different domains. Table 1 presents some statistics about these datasets. We adopt a max-margin hinge loss as training objective. The results are reported in terms of MAP and MRR in WikiQA and TREC-QA and accuracy in InsuranceQA.

We use bidirectional GRU for all models. We share the GRU parameter between question and answer which has shown significant improvement on performance and convergency rate (Tan et al., 2015; Feng et al., 2015).

There are two common baseline systems for above three datasets:

- **GRU**: A non-attentive GRU-RNN that models the question and answer separately.
- **OARNN**: Outer attention-based RNN models (OARNN) with GRU which is detailed in Section 5.1.

WikiQA (Yang et al., 2015) is a recently released open-domain question answering

[3] 10000 for WikiQA and 5000 for TrecQA in experiment.
[4] https://code.google.com/archive/p/word2vec/

Dataset(train / test / dev)	InsuranceQA	WikiQA	TREC-QA
# of questions	12887 / 1800x2 /1000	873 / 243 / 126	78 / 68 / 65
# of sentences	24981(ALL)	20360 / 6165 / 2733	5919 / 1442 / 1117
Ave length of question	7.16	7.16 / 7.26 / 7.23	11.39 / 8.63 / 8.00
Ave length of sentence	49.5	25.29 / 24.59 / 24.59	30.39 / 25.61 / 24.9

Table 1: The statistics of three answer selection datasets. For the TREC-QA, we use the cleaned dataset that has been edit by human. For WikiQA and TREC-QA we remove all the questions that has no right or wrong answers.

System	**MAP**	**MRR**
(Yang et al., 2015)	0.652	0.6652
(Yin et al., 2015)	0.6921	0.7108
(Santos et al., 2016)	0.6886	0.6957
GRU	0.6581	0.6691
OARNN	0.6881	0.7013
IARNN-word	0.7098	0.7234
IARNN-Occam(word)	0.7121	0.7318
IARNN-context	0.7182	0.7339
IARNN-Occam(context)	**0.7341**	**0.7418**
IARNN-Gate	0.7258	0.7394

Table 2: Performances on WikiQA

dataset in which all answers are collected from Wikipedia. In addition to the original (question,positive,negative) triplets, we randomly select a bunch of negative answer candidates from answer sentence pool and finally we get a relatively abundant 50,298 triplets. We use cosine similarity to compare the question and candidate answer sentence. The hidden variable's length is set to 165 and batch size is set to 1. We use *sigmoid* as GRU inner active function, we keep word embedding fixed during training. Margin M was set to 0.15 which is tuned in the development set. We adopt three additional baseline systems applied to WikiQA: (1) A bigram CNN models with average pooling(Yang et al., 2015). (2) An attention-based CNN model which uses an interactive attention matrix for both question and answer(Yin et al., 2015) [5] (3) An attention based CNN models which builds the attention matrix after sentence representation(Santos et al., 2016). The result is shown in Table 2.

InsuranceQA (Feng et al., 2015) is a domain specific answer selection dataset in which all questions is related to insurance. Its vocabulary size is comparatively small (22,353), we set the batch size to 16 and the hidden variable size to 145, hinge loss margin M is adjusted to 0.12 by evaluation behavior. Word embeddings are also learned during training. We adopt the Geometric mean of Euclidean and Sigmoid Dot (GESD) proposed in (Feng et al., 2015) to measure the similarity between two representations:

$$GESD(x,y) = \frac{1}{1+||x-y||} \times \frac{1}{1+exp(-\gamma(xy^T+c))} \quad (10)$$

which shows advantage over cosine similarity in experiments.

We report accuracy instead of MAP/MRR because one question only has one right answers in InsuranceQA. The result is shown in Table 3.

System	Dev	Test1	Test2
(Feng et al., 2015)	65.4	65.3	61.0
(Santos et al., 2016)	66.8	67.8	60.3
GRU	59.4	53.2	58.1
OARNN	65.4	66.1	60.2
IARNN-word	67.2125	67.0651	61.5896
IARNN-Occam(word)	69.9130	69.5923	63.7317
IARNN-context	67.1025	66.7211	63.0656
IARNN-Occam(context)	69.1125	68.8651	**65.1396**
IARNN-Gate	**69.9812**	**70.1128**	62.7965

Table 3: Experiment result in InsuranceQA, (Feng et al., 2015) is a CNN architecture without attention mechanism.

System	**MAP**	**MRR**
(Wang and Nyberg, 2015) †	0.7134	0.7913
(Wang and Ittycheriah, 2015) †	0.7460	0.8200
(Santos et al., 2016) †	**0.7530**	**0.8511**
GRU	0.6487	0.6991
OARNN	0.6887	0.7491
IARNN-word	0.7098	0.7757
IARNN-Occam(word)	0.7162	0.7916
IARNN-context	0.7232	0.8069
IARNN-Occam(context)	0.7272	0.8191
IARNN-Gate	0.7369	0.8208

Table 4: Result of different systems in Trec-QA.(Wang and Ittycheriah, 2015) propose a question similarity model to extract features from word alignment between two questions which is suitable to FAQ based QA. It needs to mention that the system marked with † are learned on TREC-QA original full training data.

TREC-QA was created by Wang et al.(2007) based on Text REtrieval Conference (TREC) QA track (8-13) data. The size of hidden variable was set to 80, M was set to 0.1. This dataset is comparatively small so we set word embedding vector

[5] In their experiment some extra linguistic features was also added for better performance.

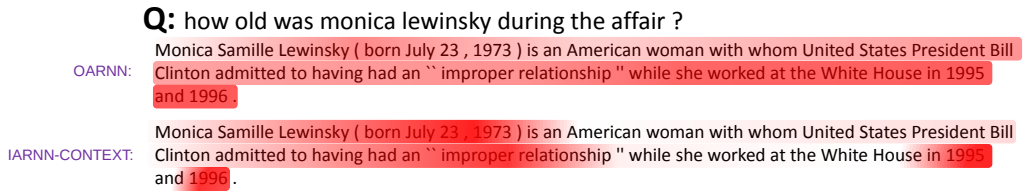


Figure 7: An example demonstrates the advantage of IARNN in capturing the informed part of a sentence compared with OARNN.

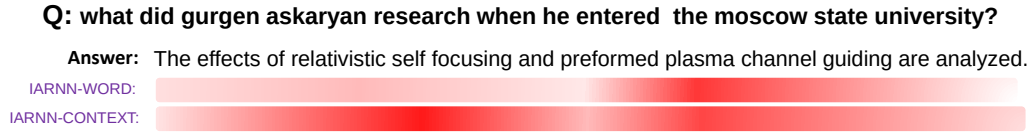


Figure 8: An example illustrates the IARNN-CONTEXT could attend the consecutive words in a sentence.

size to 50 and update it during training. It needs to mention that we do not use the original TREC-QA training data but the smaller one which has been edited by human. The result is shown in Table 4.

6 Result and Analysis

We can see from the result tables that the attention based RNN models achieve better results than the non-attention RNN models (GRU). OARNN and IARNN beat the non-attentive GRU in every datasets by a large margin, which proves the importance of attention mechanism in representing answer sentence in AS. For the non-attentive models, the fixed width of the hidden vectors is a bottleneck for interactive information flow, so the informative part of the question could only propagate through the similarity score which is blurred for the answer representation to be properly learned. But in attention based models, the question attention information is introduced to influence the answer sentence representation explicitly, in this way we can improve sentence representation for the specific target (or topic (Ghosh et al., 2016)).

The inner attention RNN models outperform outer attention model in three datasets, this is corresponds to our intuition that the bias attention problem in OARNN may cause a biasd sentence representation. An example of the attention heatmap is shown in Figure7. To answer the question, we should focus on *"born July 23 , 1973"* which is located at the beginning of the sentence. But in OARNN, the attention is biases towards the last few last words in the answer. In IARNN-CONTEXT, the attention is paid to the relevant part and thus results in a more relevant representation.

The attention with context information could also improves the result, we can see that IARNN-CONTEXT and IARNN-GATE outperform IARNN-WORD in three experiments. IARNN-WORD may ignore the importance of some words because it attends answer word by word, for example in Figure8, the specific word *self* or *focusing* may not be related to the question by itself, but their combination and the previous word *relativistic* is very informative for answering the question. In IARNN-CONTEXT we add attention information dynamically in RNN process, thus it could capture the relationship between word and its context.

In general, we can see from table3-5 that the IARNN-GATE outperforms IARNN-CONTEXT and IARNN-WORD. In IARNN-WORD and IARNN-CONTEXT, the attention is added to impact each word representation, but the recurrent process of updating RNN hidden state representations are not influenced. IARNN-GATE embeds the attention into RNN inner activation, the *attentive activation gate* are more capable of controlling the attention information in RNN. This enlights an important future work: we could add attention information as an individual activation gate, and use this additional gate to control attention information flow in RNN. The regulation of the attention weights (Occam's attention) could also improve the representation. We also conduct an experiment on WikiQA (training process) to measure the Occam's attention regulation on different type of questions. We use rules to classify question into 6 types

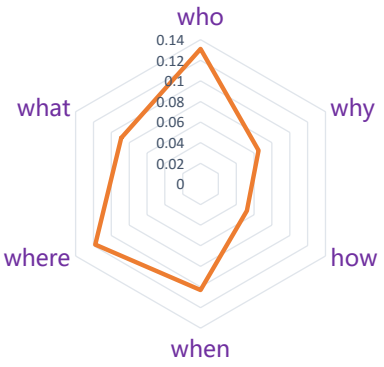


Figure 9: The Occam's attention regulation on different types of question.

(i.e. who,why,how,when,where,what), and each of them has the same number of samples to avoid data imbalance. We report the Occam'm regulation (n_p^i in Equation.8) in Figure 9. As we can see from the radar graph, *who* and *where* are regulized severely compared with other types of question, this is correspond to their comparetively less information in the answer candidate to answer the question. This emphasize that different types question should impose different amount of regulation on its candidate answers. The experiment result on three AS datasets shows that the improvement of Occam's attention is significant in WikiQA and insuranceQA. Because most of the sentence are relatively long in these two datasets, and the longer the sentence, the more noise it may contain, so we should punish the summation of the attention weights to remove some irrelevant parts. Our question-specific Occam's attention punishes the summation of attention and thus achieves a better result for both IARNN-WORD and IARNN-CONTEXT.

7 Conclusion and Future Work

In this work we present some variants of traditional attention-based RNN models with GRU. The key idea is *attention before representation*. We analyze the deficiency of traditional outer attention-based RNN models qualitatively and quantitatively. We propose three models where attention is embedded into representation process. Occam's Razor is further implemented to this attention for better representation. Our results on answer selection demonstrate that the inner attention outperforms the outer attention in RNN. Our models can be further extended to other NLP tasks such as recognizing textual entailments where attention mechanism is important for sentence representation. In the future we plan to apply our *inner-attention* intuition to other neural networks such as CNN or multi-layer perceptron.

Acknowledgments

The work was supported by the Natural Science Foundation of China (No.61533018), the National High Technology Development 863 Program of China (No.2015AA015405) and the National Natural Science Foundation of China (No.61272332). And this research work was also supported by Google through focused research awards program.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Minwei Feng, Bing Xiang, Michael R Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: A study and an open task. *IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Shalini Ghosh, Oriol Vinyals, Brian Strope, Scott Roy, Tom Dean, and Larry Heck. 2016. Contextual lstm (clstm) models for large scale nlp tasks. *arXiv preprint arXiv:1602.06291*.

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

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350.

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5528–5531. IEEE.

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1310–1318.

Jonathan Raiman and Szymon Sidor. 2015. Occam's gates. *arXiv preprint arXiv:1506.08251*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*, pages 458–467.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642. Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

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

Ming Tan, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108*.

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*.

Di Wang and Eric Nyberg. 2015. A long short-term memory model for answer sentence selection in question answering. *ACL, July*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*, volume 7, pages 22–32.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *arXiv preprint arXiv:1410.3916*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Citeseer.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *HLT-NAACL*, pages 858–867. Citeseer.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of ACL*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Yin Zheng, Richard S Zemel, Yu-Jin Zhang, and Hugo Larochelle. 2015. A neural autoregressive approach to attention-based recognition. *International Journal of Computer Vision*, 113(1):67–79.

Relation Classification via Multi-Level Attention CNNs

Linlin Wang[1*], **Zhu Cao**[1*], **Gerard de Melo**[2], **Zhiyuan Liu**[3†]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
[2] Department of Computer Science, Rutgers University, Piscataway, NJ, USA
[3]State Key Laboratory of Intelligent Technology and Systems,
Tsinghua National Laboratory for Information Science and Technology,
Department of Computer Science and Technology, Tsinghua University, Beijing, China
{ll-wang13,cao-z13}@mails.tsinghua.edu.cn, gdm@demelo.org

Abstract

Relation classification is a crucial ingredient in numerous information extraction systems seeking to mine structured facts from text. We propose a novel convolutional neural network architecture for this task, relying on two levels of attention in order to better discern patterns in heterogeneous contexts. This architecture enables end-to-end learning from task-specific labeled data, forgoing the need for external knowledge such as explicit dependency structures. Experiments show that our model outperforms previous state-of-the-art methods, including those relying on much richer forms of prior knowledge.

1 Introduction

Relation classification is the task of identifying the semantic relation holding between two nominal entities in text. It is a crucial component in natural language processing systems that need to mine explicit facts from text, e.g. for various information extraction applications as well as for question answering and knowledge base completion (Tandon et al., 2011; Chen et al., 2015). For instance, given the example input

"Fizzy [drinks] and meat cause heart disease and [diabetes]."

with annotated target entity mentions e_1 = "drinks" and e_2 = "diabetes", the goal would be to automatically recognize that this sentence expresses a cause-effect relationship between e_1 and e_2, for which we use the notation Cause-Effect(e_1,e_2). Accurate relation classification facilitates precise sentence interpretations, discourse processing, and higher-level NLP tasks (Hendrickx et al., 2010). Thus, relation classification has attracted considerable attention from researchers over the course of the past decades (Zhang, 2004; Qian et al., 2009; Rink and Harabagiu, 2010).

In the example given above, the verb corresponds quite closely to the desired target relation. However, in the wild, we encounter a multitude of different ways of expressing the same kind of relationship. This challenging variability can be lexical, syntactic, or even pragmatic in nature. An effective solution needs to be able to account for useful semantic and syntactic features not only for the meanings of the target entities at the lexical level, but also for their immediate context and for the overall sentence structure.

Thus, it is not surprising that numerous feature- and kernel-based approaches have been proposed, many of which rely on a full-fledged NLP stack, including POS tagging, morphological analysis, dependency parsing, and occasionally semantic analysis, as well as on knowledge resources to capture lexical and semantic features (Kambhatla, 2004; Zhou et al., 2005; Suchanek et al., 2006; Qian et al., 2008; Mooney and Bunescu, 2005; Bunescu and Mooney, 2005). In recent years, we have seen a move towards deep architectures that are capable of learning relevant representations and features without extensive manual feature engineering or use of external resources. A number of convolutional neural network (CNN), recurrent neural network (RNN), and other neural architectures have been proposed for relation classification (Zeng et al., 2014; dos Santos et al., 2015; Xu et al., 2015b). Still, these models often fail to identify critical cues, and many of them still require an external dependency parser.

We propose a novel CNN architecture that addresses some of the shortcomings of previous approaches. Our key contributions are as follows:

1. Our CNN architecture relies on a novel multi-

[*] Equal contribution.
[†] Corresponding author. Email: liuzy@tsinghua.edu.cn

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1298–1307,
Berlin, Germany, August 7-12, 2016.

level attention mechanism to capture both entity-specific attention (primary attention at the input level, with respect to the target entities) and relation-specific pooling attention (secondary attention with respect to the target relations). This allows it to detect more subtle cues despite the heterogeneous structure of the input sentences, enabling it to automatically learn which parts are relevant for a given classification.

2. We introduce a novel pair-wise margin-based objective function that proves superior to standard loss functions.
3. We obtain the new state-of-the-art results for relation classification with an F1 score of 88.0% on the SemEval 2010 Task 8 dataset, outperforming methods relying on significantly richer prior knowledge.

2 Related Work

Apart from a few unsupervised clustering methods (Hasegawa et al., 2004; Chen et al., 2005), the majority of work on relation classification has been supervised, typically cast as a standard multi-class or multi-label classification task. Traditional feature-based methods rely on a set of features computed from the output of an explicit linguistic preprocessing step (Kambhatla, 2004; Zhou et al., 2005; Boschee et al., 2005; Suchanek et al., 2006; Chan and Roth, 2010; Nguyen and Grishman, 2014), while kernel-based methods make use of convolution tree kernels (Qian et al., 2008), subsequence kernels (Mooney and Bunescu, 2005), or dependency tree kernels (Bunescu and Mooney, 2005). These methods thus all depend either on carefully handcrafted features, often chosen on a trial-and-error basis, or on elaborately designed kernels, which in turn are often derived from other pre-trained NLP tools or lexical and semantic resources. Although such approaches can benefit from the external NLP tools to discover the discrete structure of a sentence, syntactic parsing is error-prone and relying on its success may also impede performance (Bach and Badaskar, 2007). Further downsides include their limited lexical generalization abilities for unseen words and their lack of robustness when applied to new domains, genres, or languages.

In recent years, deep neural networks have shown promising results. The Recursive Matrix-Vector Model (MV-RNN) by Socher et al. (2012) sought to capture the compositional aspects of the sentence semantics by exploiting syntactic trees. Zeng et al. (2014) proposed a deep convolutional neural network with softmax classification, extracting lexical and sentence level features. However, these approaches still depend on additional features from lexical resources and NLP toolkits. Yu et al. (2014) proposed the Factor-based Compositional Embedding Model, which uses syntactic dependency trees together with sentence-level embeddings. In addition to dos Santos et al. (2015), who proposed the Ranking CNN (CR-CNN) model with a class embedding matrix, Miwa and Bansal (2016) similarly observed that LSTM-based RNNs are outperformed by models using CNNs, due to limited linguistic structure captured in the network architecture. Some more elaborate variants have been proposed to address this, including bidirectional LSTMs (Zhang et al., 2015), deep recurrent neural networks (Xu et al., 2016), and bidirectional tree-structured LSTM-RNNs (Miwa and Bansal, 2016). Several recent works also reintroduce a dependency tree-based design, e.g., RNNs operating on syntactic trees (Hashimoto et al., 2013), shortest dependency path-based CNNs (Xu et al., 2015a), and the SDP-LSTM model (Xu et al., 2015b). Finally, Nguyen and Grishman (2015) train both CNNs and RNNs and variously aggregate their outputs using voting, stacking, or log-linear modeling (Nguyen and Grishman, 2015). Although these recent models achieve solid results, ideally, we would want a simple yet effective architecture that does not require dependency parsing or training multiple models. Our experiments in Section 4 demonstrate that we can indeed achieve this, while also obtaining substantial improvements in terms of the obtained F1 scores.

3 The Proposed Model

Given a sentence S with a labeled pair of entity mentions e_1 and e_2 (as in our example from Section 1), relation classification is the task of identifying the semantic relation holding between e_1 and e_2 among a set of candidate relation types (Hendrickx et al., 2010). Since the only input is a raw sentence with two marked mentions, it is non-trivial to obtain all the lexical, semantic and syntactic cues necessary to make an accurate prediction.

To this end, we propose a novel multi-level attention-based convolution neural network model. A schematic overview of our architecture is given

Notation	Definition	Notation	Definition
$\mathbf{w}_i^{\mathrm{M}}$	Final word emb.	$\mathbf{z}_i$	Context emb.
W_f	Conv. weight	B_f	Conv. bias
$\mathbf{w}^{\mathrm{O}}$	Network output	W^{L}	Relation emb.
A^j	Input att.	A^{p}	Pooling att.
G	Correlation matrix		

Table 1: Overview of main notation.

in Figure 1. The input sentence is first encoded using word vector representations, exploiting the context and a positional encoding to better capture the word order. A primary attention mechanism, based on diagonal matrices is used to capture the relevance of words with respect to the target entities. To the resulting output matrix, one then applies a convolution operation in order to capture contextual information such as relevant n-grams, followed by max-pooling. A secondary attention pooling layer is used to determine the most useful convolved features for relation classification from the output based on an attention pooling matrix. The remainder of this section will provide further details about this architecture. Table 1 provides an overview of the notation we will use for this. The final output is given by a new objective function, described below.

3.1 Classification Objective

We begin with top-down design considerations for the relation classification architecture. For a given sentence S, our network will ultimately output some $\mathbf{w}^{\mathrm{O}}$. For every output relation $y \in \mathcal{Y}$, we assume there is a corresponding output embedding W_y^{L}, which will automatically be learnt by the network (dos Santos et al., 2015).

We propose a novel distance function $\delta_\theta(S)$ that measures the proximity of the predicted network output $\mathbf{w}^{\mathrm{O}}$ to a candidate relation y as follows.

$$\delta_\theta(S, y) = \left\| \frac{\mathbf{w}^{\mathrm{O}}}{|\mathbf{w}^{\mathrm{O}}|} - W_y^{\mathrm{L}} \right\| \quad (1)$$

using the L_2 norm (note that W_y^{L} are already normalized). Based on this distance function, we design a margin-based pairwise loss function $\mathcal{L}$ as

$$\begin{aligned}\mathcal{L} &= \left[\delta_\theta(S, y) + (1 - \delta_\theta(S, \hat{y}^-))\right] + \beta\|\theta\|^2 \\ &= \left[1 + \left\| \frac{\mathbf{w}^{\mathrm{O}}}{|\mathbf{w}^{\mathrm{O}}|} - W_y^{L} \right\| - \left\| \frac{\mathbf{w}^{\mathrm{O}}}{|\mathbf{w}^{\mathrm{O}}|} - W_{\hat{y}^-}^{\mathrm{L}} \right\|\right] \\ &\quad + \beta\|\theta\|^2, \end{aligned} \quad (2)$$

where 1 is the margin, β is a parameter, $\delta_\theta(S, y)$ is the distance between the predicted label embedding W^{L} and the ground truth label y and $\delta_\theta(S, \hat{y}^-)$ refers to the distance between $\mathbf{w}^{\mathrm{O}}$ and a selected incorrect relation label $\hat{y}^-$. The latter is chosen as the one with the highest score among all incorrect classes (Weston et al., 2011; dos Santos et al., 2015), i.e.

$$\hat{y}^- = \underset{y' \in \mathcal{Y}, y' \neq y}{\operatorname{argmax}} \ \delta(S, y'). \quad (3)$$

This margin-based objective has the advantage of a strong interpretability and effectiveness compared with empirical loss functions such as the ranking loss function in the CR-CNN approach by dos Santos et al. (2015). Based on a distance function motived by word analogies (Mikolov et al., 2013b), we minimize the gap between predicted outputs and ground-truth labels, while maximizing the distance with the selected incorrect class. By minimizing this pairwise loss function iteratively (see Section 3.5), $\delta_\theta(S, y)$ are encouraged to decrease, while $\delta_\theta(S, \hat{y}^-)$ increase.

3.2 Input Representation

Given a sentence $S = (w_1, w_2, ..., w_n)$ with marked entity mentions e_1(=w_p) and e_2(=w_t), $(p, t \in [1, n],\ p \neq t)$, we first transform every word into a real-valued vector to provide lexical-semantic features. Given a word embedding matrix W_V of dimensionality $d_{\mathrm{w}} \times |V|$, where V is the input vocabulary and d_{w} is the word vector dimensionality (a hyper-parameter), we map every w_i to a column vector $\mathbf{w}_i^{\mathrm{d}} \in \mathbb{R}^{d_{\mathrm{w}}}$.

To additionally capture information about the relationship to the target entities, we incorporate word position embeddings (WPE) to reflect the relative distances between the i-th word to the two marked entity mentions (Zeng et al., 2014; dos Santos et al., 2015). For the given sentence in Fig. 1, the relative distances of word "and" to entity e_1 "drinks" and e_2 "diabetes" are -1 and 6, respectively. Every relative distance is mapped to a randomly initialized position vector in $\mathbb{R}^{d_{\mathrm{p}}}$, where d_{p} is a hyper-parameter. For a given word i, we obtain two position vectors $\mathbf{w}_{i,1}^{\mathrm{p}}$ and $\mathbf{w}_{i,2}^{\mathrm{p}}$, with regard to entities e_1 and e_2, respectively. The overall word embedding for the i-th word is $\mathbf{w}_i^{\mathrm{M}} = [(\mathbf{w}_i^{\mathrm{d}})^\intercal, (\mathbf{w}_{i,1}^{\mathrm{p}})^\intercal, (\mathbf{w}_{i,2}^{\mathrm{p}})^\intercal]^\intercal$.

Using a sliding window of size k centered around the i-th word, we encode k successive

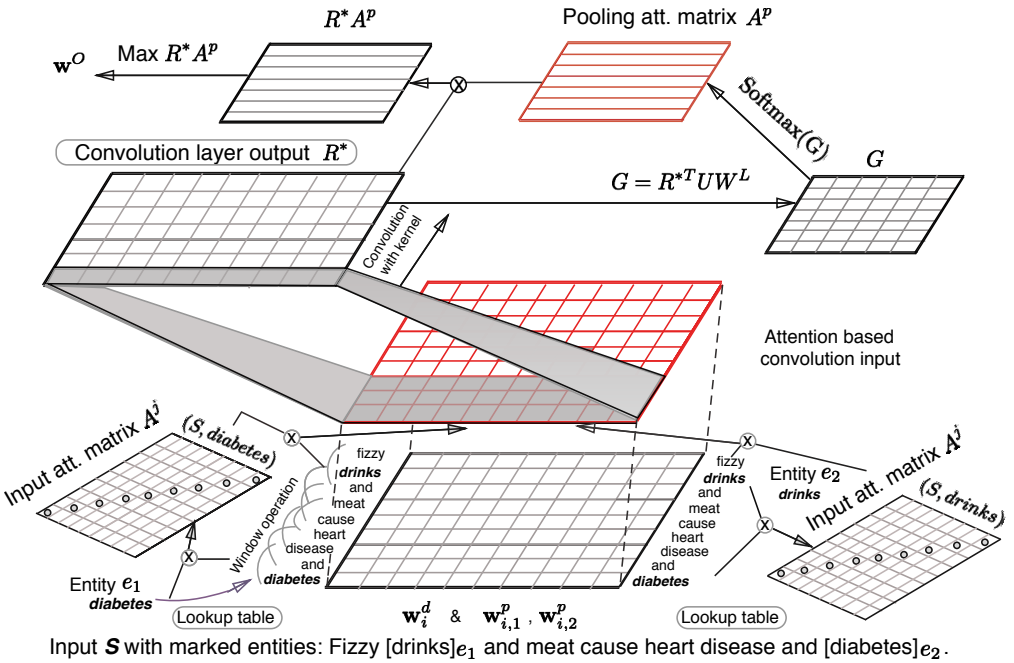


Figure 1: Schematic overview of our Multi-Level Attention Convolutional Neural Networks

words into a vector $\mathbf{z}_i \in \mathbb{R}^{(d_\mathrm{w}+2d_\mathrm{p})k}$ to incorporate contextual information as

$$\mathbf{z}_i = [(\mathbf{w}^{\mathrm{M}}_{i-(k-1)/2})^\intercal, ..., (\mathbf{w}^{\mathrm{M}}_{i+(k-1)/2})^\intercal]^\intercal \quad (4)$$

An extra padding token is repeated multiple times for well-definedness at the beginning and end of the input.

3.3 Input Attention Mechanism

While position-based encodings are useful, we conjecture that they do not suffice to fully capture the relationships of specific words with the target entities and the influence that they may bear on the target relations of interest. We design our model so as to automatically identify the parts of the input sentence that are relevant for relation classification.

Attention mechanisms have successfully been applied to sequence-to-sequence learning tasks such as machine translation (Bahdanau et al., 2015; Meng et al., 2015) and abstractive sentence summarization (Rush et al., 2015), as well as to tasks such as modeling sentence pairs (Yin et al., 2015) and question answering (Santos et al., 2016). To date, these mechanisms have generally been used to allow for an alignment of the input and output sequence, e.g. the source and target sentence in machine translation, or for an alignment between two input sentences as in sentence similarity scoring and question answering.

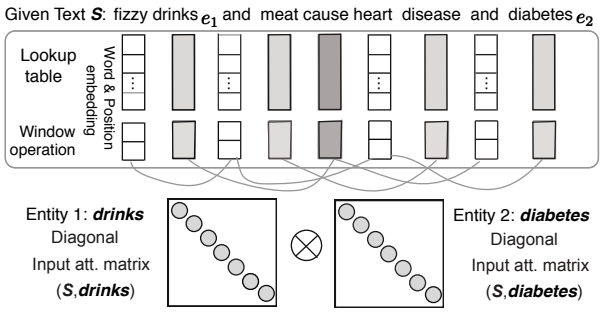


Figure 2: Input and Primary Attention

In our work, we apply the idea of modeling attention to a rather different kind of scenario involving heterogeneous objects, namely a sentence and two entities. With this, we seek to give our model the capability to determine which parts of the sentence are most influential with respect to the two entities of interest. Consider that in a long sentence with multiple clauses, perhaps only a single verb or noun might stand in a relevant relationship with a given target entity.

As depicted in Fig. 2, the input representation layer is used in conjunction with diagonal attention matrices and convolutional input composition.

Contextual Relevance Matrices. Consider the example in Fig. 1, where the non-entity word "cause" is of particular significance in determining the relation. Fortunately, we can exploit the fact

that there is a salient connection between the words "cause" and "diabetes" also in terms of corpus cooccurrences. We introduce two diagonal attention matrices A^j with values $A^j_{i,i} = f(e_j, w_i)$ to characterize the strength of contextual correlations and connections between entity mention e_j and word w_i. The scoring function f is computed as the inner product between the respective embeddings of word w_i and entity e_j, and is parametrized into the network and updated during the training process. Given the A^j matrices, we define

$$\alpha^j_i = \frac{\exp(A^j_{i,i})}{\sum_{i'=1}^{n} \exp(A^j_{i',i'})}, \tag{5}$$

to quantify the relative degree of relevance of the i-th word with respect to the j-th entity ($j \in \{1, 2\}$).

Input Attention Composition. Next, we take the two relevance factors α^1_i and α^2_i and model their joint impact for recognizing the relation via simple averaging as

$$\mathbf{r}_i = \mathbf{z}_i \, \frac{\alpha^1_i + \alpha^2_i}{2}. \tag{6}$$

Apart from this default choice, we also evaluate two additional variants. The first (Variant-1) concatenates the word vectors as

$$\mathbf{r}_i = [(\mathbf{z}_i \, \alpha^1_i)^\intercal, (\mathbf{z}_i \, \alpha^2_i)^\intercal]^\intercal, \tag{7}$$

to obtain an information-enriched input attention component for this specific word, which contains the relation relevance to both entity 1 and entity 2.

The second variant (Variant-2) interprets relations as mappings between two entities, and combines the two entity-specific weights as

$$\mathbf{r}_i = \mathbf{z}_i \, \frac{\alpha^1_i - \alpha^2_i}{2}, \tag{8}$$

to capture the relation between them.

Based on these $\mathbf{r}_i$, the final output of the input attention component is the matrix $R = [\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n]$, where n is the sentence length.

3.4 Convolutional Max-Pooling with Secondary Attention

After this operation, we apply convolutional max-pooling with another secondary attention model to extract more abstract higher-level features from the previous layer's output matrix R.

Convolution Layer. A convolutional layer may, for instance, learn to recognize short phrases such as trigrams. Given our newly generated input attention-based representation R, we accordingly apply a filter of size d^{c} as a weight matrix W_{f} of size $d^{\mathrm{c}} \times k(d^{\mathrm{w}} + 2d^{\mathrm{p}})$. Then we add a linear bias B_{f}, followed by a non-linear hyperbolic tangent transformation to represent features as follows:

$$R^* = \tanh(W_{\mathrm{f}} R + B_{\mathrm{f}}). \tag{9}$$

Attention-Based Pooling. Instead of regular pooling, we rely on an attention-based pooling strategy to determine the importance of individual windows in R^*, as encoded by the convolutional kernel. Some of these windows could represent meaningful n-grams in the input. The goal here is to select those parts of R^* that are relevant with respect to our objective from Section 3.1, which essentially calls for a relation encoding process, while neglecting sentence parts that are irrelevant for this process.

We proceed by first creating a correlation modeling matrix G that captures pertinent connections between the convolved context windows from the sentence and the relation class embedding W^{L} introduced earlier in Section 3.1:

$$G = R^{*\intercal} \, U \, W^{\mathrm{L}}, \tag{10}$$

where U is a weighting matrix learnt by the network.

Then we adopt a softmax function to deal with this correlation modeling matrix G to obtain an attention pooling matrix A^{p} as

$$A^{\mathrm{p}}_{i,j} = \frac{\exp(G_{i,j})}{\sum_{i'=1}^{n} \exp(G_{i',j})}, \tag{11}$$

where $G_{i,j}$ is the (i,j)-th entry of G and $A^{\mathrm{p}}_{i,j}$ is the (i,j)-th entry of A^{p}.

Finally, we multiply this attention pooling matrix with the convolved output R^* to highlight important individual phrase-level components, and apply a max operation to select the most salient one (Yin et al., 2015; Santos et al., 2016) for a given dimension of the output. More precisely, we obtain the output representation $\mathbf{w}^{\mathrm{O}}$ as follows in Eq. (12):

$$\mathbf{w}^{\mathrm{O}}_i = \max_j \, (R^* A^{\mathrm{p}})_{i,j}, \tag{12}$$

where $\mathbf{w}^{\mathrm{O}}_i$ is the i-th entry of $\mathbf{w}^{\mathrm{O}}$ and $(R^* A^{\mathrm{p}})_{i,j}$ is the (i,j)-th entry of $R^* A^{\mathrm{p}}$.

3.5 Training Procedure

We rely on stochastic gradient descent (SGD) to update the parameters with respect to the loss function in Eq. (2) as follows:

$$\theta' = \theta + \lambda \frac{d(\sum_{i=1}^{|\mathcal{S}|} [\delta_\theta(S_i, y) + (1 - \delta_\theta(S_i, \hat{y}_i^-))])}{d\theta} + \lambda_1 \frac{d(\beta||\theta||^2)}{d\theta} \quad (13)$$

where λ and λ_1 are learning rates, and incorporating the β parameter from Eq. (2).

4 Experiments

4.1 Experimental Setup

Dataset and Metric. We conduct our experiments on the commonly used SemEval-2010 Task 8 dataset (Hendrickx et al., 2010), which contains 10,717 sentences for nine types of annotated relations, together with an additional "Other" type. The nine types are: Cause-Effect, Component-Whole, Content-Container, Entity-Destination, Entity-Origin, Instrument-Agency, Member-Collection, Message-Topic, and Product-Producer, while the relation type "Other" indicates that the relation expressed in the sentence is not among the nine types. However, for each of the aforementioned relation types, the two entities can also appear in inverse order, which implies that the sentence needs to be regarded as expressing a different relation, namely the respective inverse one. For example, Cause-Effect(e_1,e_2) and Cause-Effect(e_2,e_1) can be considered two distinct relations, so the total number $|\mathcal{Y}|$ of relation types is 19. The SemEval-2010 Task 8 dataset consists of a training set of 8,000 examples, and a test set with the remaining examples. We evaluate the models using the official scorer in terms of the Macro-F1 score over the nine relation pairs (excluding Other).

Settings. We use the word2vec skip-gram model (Mikolov et al., 2013a) to learn initial word representations on Wikipedia. Other matrices are initialized with random values following a Gaussian distribution. We apply a cross-validation procedure on the training data to select suitable hyperparameters. The choices generated by this process are given in Table 2.

Parameter	Parameter Name	Value
d^{p}	Word Pos. Emb. Size	25
d^{c}	Conv. Size	1000
k	Word Window Size	3
λ	Learning rate	0.03
λ_1	Learning rate	0.0001

Table 2: Hyperparameters.

4.2 Experimental Results

Table 3 provides a detailed comparison of our Multi-Level Attention CNN model with previous approaches. We observe that our novel attention-based architecture achieves new state-of-the-art results on this relation classification dataset. Att-Input-CNN relies only on the primal attention at the input level, performing standard max-pooling after the convolution layer to generate the network output $\mathbf{w}^{\mathrm{O}}$, in which the new objective function is utilized. With Att-Input-CNN, we achieve an F1-score of 87.5%, thus already outperforming not only the original winner of the SemEval task, an SVM-based approach (82.2%), but also the well-known CR-CNN model (84.1%) with a relative improvement of 4.04%, and the newly released DRNNs (85.8%) with a relative improvement of 2.0%, although the latter approach depends on the Stanford parser to obtain dependency parse information. Our full dual attention model Att-Pooling-CNN achieves an even more favorable F1-score of 88%.

Table 4 provides the experimental results for the two variants of the model given by Eqs. (7) and (8) in Section 3.3. Our main model outperforms the other variants on this dataset, although the variants may still prove useful when applied to other tasks. To better quantify the contribution of the different components of our model, we also conduct an ablation study evaluating several simplified models. The first simplification is to use our model without the input attention mechanism but with the pooling attention layer. The second removes both attention mechanisms. The third removes both forms of attention and additionally uses a regular objective function based on the inner product $s = r \cdot w$ for a sentence representation r and relation class embedding w. We observe that all three of our components lead to noticeable improvements over these baselines.

4.3 Detailed Analysis

Primary Attention. To inspect the inner workings of our model, we considered the primary attention matrices of our multi-level attention model

Classifier	F1
Manually Engineered Methods	
SVM (Rink and Harabagiu, 2010)	82.2
Dependency Methods	
RNN (Socher et al., 2012)	77.6
MVRNN (Socher et al., 2012)	82.4
FCM (Yu et al., 2014)	83.0
Hybrid FCM (Yu et al., 2014)	83.4
SDP-LSTM (Xu et al., 2015b)	83.7
DRNNs (Xu et al., 2016)	85.8
SPTree (Miwa and Bansal, 2016)	84.5
End-To-End Methods	
CNN+ Softmax (Zeng et al., 2014)	82.7
CR-CNN (dos Santos et al., 2015)	84.1
DepNN (Liu et al., 2015)	83.6
depLCNN+NS (Xu et al., 2015a)	85.6
STACK-FORWARD$*$	83.4
VOTE-BIDIRECT$*$	84.1
VOTE-BACKWARD$*$	84.1
Our Architectures	
Att-Input-CNN	87.5
Att-Pooling-CNN	**88.0**

Table 3: Comparison with results published in the literature, where '$*$' refers to models from Nguyen and Grishman (2015).

for the following randomly selected sentence from the test set:

> The disgusting scene was retaliation against her brother Philip who rents the [room]$_{e_1}$ inside this apartment [house]$_{e_2}$ on Lombard street.

Fig. 3 plots the word-level attention values for the input attention layer to act as an example, using the calculated attention values for every individual word in the sentence. We find the word "inside" was assigned the highest attention value, while words such as "room" and "house" also are deemed important. This appears sensible in light of the ground-truth labeling as a Component-Whole(e_1,e_2) relationship. Additionally, we observe that words such as "this", which are rather irrelevant with respect to the target relationship, indeed have significantly lower attention scores.

Most Significant Features for Relations. Table 5 lists the top-ranked trigrams for each relation class y in terms of their contribution to the score for determining the relation classification. Recall the definition of $\delta_\theta(x, y)$ in Eq. (1). In the network, we trace back the trigram that contributed most to the correct classification in terms of $\delta_\theta(S_i, y)$ for each sentence S_i. We then rank all such trigrams in the sentences in the test set according to their total contribution and list the top-ranked trigrams.[*] In Table 5, we see that these are indeed very informative for deducing the relation. For example, the top trigram for Cause-Effect(e_2,e_1) is "are caused by", which strongly implies that the first entity is an effect caused by the latter. Similarly, the top trigram for Entity-Origin(e_1,e_2) is "from the e_2", which suggests that e_2 could be an original location, at which entity e_1 may have been located.

Classifier	F1
Att-Input-CNN (Main)	87.5
Att-Input-CNN (Variant-1)	87.2
Att-Input-CNN (Variant-2)	87.3
Att-Pooling-CNN (regular)	88.0
– w/o input attention	86.6
– w/o any attention	86.1
– w/o any attention, w/o δ-objective	84.1

Table 4: Comparison between the main model and variants as well as simplified models.

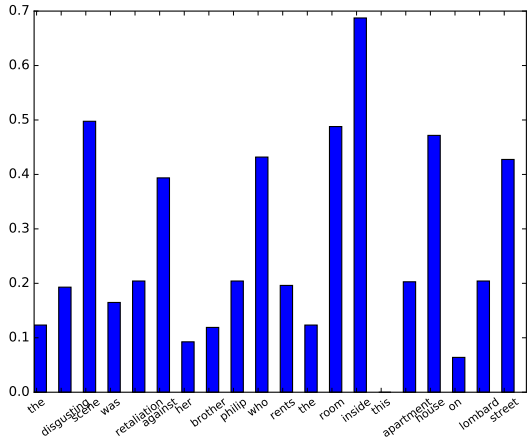


Figure 3: Input Attention Visualization. The value of the y-coordinate is computed as $100 * (A_i^t - \min_{i \in \{1,\ldots,n\}} A_i^t)$, where A_i^t stands for the overall attention weight assigned to the word i.

Error Analysis. Further, we examined some of the misclassifications produced by our model. The following is a typical example of a wrongly classified sentence:

[*]For Entity-Destination(e_2,e_1), there was only one occurrence in the test set.

Relation	(e_1, e_2)	(e_2, e_1)
Cause-Effect	e_1 caused a, caused a e_2, e_1 resulted in, the cause of, had caused the, poverty cause e_2	e_2 caused by, e_2 from e_1, is caused by, are caused by, was caused by, been caused by
Component-Whole	e_1 of the, of a e_2, of the e_2, in the e_2, part of the	with its e_2, e_1 consists of, e_1 has a, e_1 comprises e_2
Content-Container	in a e_2, was hidden in, inside a e_2, was contained in	e_1 with e_2, filled with e_2, e_1 contained a, full of e_2,
Entity-Destination	e_1 into the, e_1 into a, was put inside, in a e_2	had thrown into
Entity-Origin	from this e_2, is derived from, from the e_2, away from the	e_1 e_2 is, the e_1 e_2, for e_1 e_2, the source of
Instrument-Agency	for the e_2, is used by, by a e_2, with the e_2, a e_1 e_2	e_1 use e_2, with a e_2, by using e_2
Member-Collection	of the e_2, in the e_2, a member of, from the e_2	a e_1 of, e_1 of various, e_1 of e_2, the e_1 of
Message-Topic	on the e_2, e_1 asserts the, e_1 points out, e_1 is the	the e_1 of, described in the, the topic for, in the e_2
Product-Producer	e_1 made by, made by e_2, from the e_2, by the e_2	has constructed a, came up with, has drawn up, e_1 who created

Table 5: Most representative trigrams for different relations.

> A [film]$_{e_1}$ revolves around a [cadaver]$_{e_2}$ who seems to bring misfortune on those who come in contact with it.

This sentence is wrongly classified as belonging to the "Other" category, while the ground-truth label is Message-Topic(e_1,e_2). The phrase "revolves around" does not appear in the training data, and moreover is used metaphorically, rather than in its original sense of turning around, making it difficult for the model to recognize the semantic connection.

Another common issue stems from sentences of the form "... e_1 e_2 ...", such as the following ones:

> The size of a [tree]$_{e_1}$ [crown]$_{e_2}$ is strongly ...
> Organic [sesame]$_{e_1}$ [oil]$_{e_2}$ has an ...
> Before heading down the [phone]$_{e_1}$ [operator]$_{e_2}$ career ...

These belong to three different relation classes, Component-Whole(e_2,e_1), Entity-Origin(e_2,e_1), and Instrument-Agency(e_1,e_2), respectively, which are only implicit in the text, and the context is not particularly helpful. More informative word embeddings could conceivably help in such cases.

Convergence. Finally, we examine the convergence behavior of our two main methods. We plot the performance of each iteration in the Att-Input-CNN and Att-Pooling-CNN models in Fig. 4. It can be seen that Att-Input-CNN quite smoothly converges to its final F1 score, while for the Att-Pooling-CNN model, which includes an additional attention layer, the joint effect of these two attention layer induces stronger back-propagation effects. On the one hand, this leads to a seesaw phenomenon in the result curve, but on the other hand it enables us to obtain better-suited models with slightly higher F1 scores.

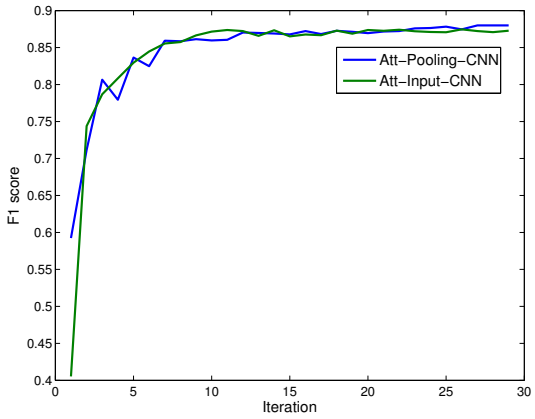


Figure 4: Training Progress of Att-Input-CNN and Att-Pooling-CNN across iterations.

5 Conclusion

We have presented a CNN architecture with a novel objective and a new form of attention mechanism that is applied at two different levels. Our results show that this simple but effective model is able to outperform previous work relying on substantially richer prior knowledge in the form of structured models and NLP resources. We expect this sort of architecture to be of interest also beyond the specific task of relation classification, which we intend to explore in future work.

Acknowledgments

The research at IIIS is supported by China 973 Program Grants 2011CBA00300, 2011CBA00301, and NSFC Grants 61033001, 61361136003, 61550110504. Prof. Liu is supported by the China 973 Program Grant 2014CB340501 and NSFC Grants 61572273 and 61532010.

References

Nguyen Bach and Sameer Badaskar. 2007. A review of relation extraction. *Online at* `http://www.cs.cmu.edu/%7Enbach/papers/A-survey-on-Relation-Extraction.pdf`.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Elizabeth Boschee, Ralph Weischedel, and Alex Zamanian. 2005. Automatic information extraction. In *Proceedings of the 2005 International Conference on Intelligence Analysis, McLean, VA*, pages 2–4.

Razvan C Bunescu and Raymond J Mooney. 2005. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 724–731. Association for Computational Linguistics.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 152–160. Association for Computational Linguistics.

Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. 2005. Unsupervised feature selection for relation extraction. In *Proceedings of IJCNLP*.

Jiaqiang Chen, Niket Tandon, and Gerard de Melo. 2015. Neural word representations from large-scale commonsense knowledge. In *Proceedings of WI 2015*.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 626–634.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, page 415. Association for Computational Linguistics.

Kazuma Hashimoto, Makoto Miwa, Yoshimasa Tsuruoka, and Takashi Chikayama. 2013. Simple customization of recursive neural networks for semantic relation classification. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1372–1376, Seattle, Washington, USA, October. Association for Computational Linguistics.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38. Association for Computational Linguistics.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of the 42nd Annual Meeting of the Association for Computation Linguistics*, page 22. Association for Computational Linguistics.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 285–290.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. 2015. Encoding source language with convolutional neural network for machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7the International Joint Conference on Natural Language Processing*, pages 20–30.

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

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

Makoto Miwa and Mohit Bansal. 2016. End-to-end relation extraction using LSTMs on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Raymond J Mooney and Razvan C Bunescu. 2005. Subsequence kernels for relation extraction. In *Advances in Neural Information Processing Systems*, pages 171–178.

Thien Huu Nguyen and Ralph Grishman. 2014. Employing word representations and regularization for domain adaptation of relation extraction. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Short Papers)*, pages 68–74.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics*, volume 1, pages 697–704. Association for Computational Linguistics.

Longhua Qian, Guodong Zhou, Fang Kong, and Qiaoming Zhu. 2009. Semi-supervised learning for semantic relation classification using stratified sampling strategy. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259. Association for Computational Linguistics.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of EMNLP 2015*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv preprint arXiv:1602.03609*.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211.

Fabian M Suchanek, Georgiana Ifrim, and Gerhard Weikum. 2006. Combining linguistic and statistical analysis to extract relations from web documents. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 712–717. ACM.

Niket Tandon, Gerard de Melo, and Gerhard Weikum. 2011. Deriving a Web-scale common sense fact database. In *Proceedings of the Twenty-fifth AAAI Conference on Artificial Intelligence (AAAI 2011)*, pages 152–157, Palo Alto, CA, USA. AAAI Press.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of IJCAI*, volume 11, pages 2764–2770.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015a. Semantic relation classification via convolutional neural networks with simple negative sampling. *Proceedings of EMNLP 2015*.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015b. Classifying relations via long short term memory networks along shortest dependency paths. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (to appear)*.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. ABCNN: attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193*.

Mo Yu, Matthew Gormley, and Mark Dredze. 2014. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, Jun Zhao, et al. 2014. Relation classification via convolutional deep neural network. In *COLING*, pages 2335–2344.

Shu Zhang, Dequan Zheng, Xinchen Hu, and Ming Yang. 2015. Bidirectional long short-term memory networks for relation classification. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation pages*, pages 73–78.

Zhu Zhang. 2004. Weakly-supervised relation classification for information extraction. In *In Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management*.

Guodong Zhou, Su Jian, Zhang Jie, and Zhang Min. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 427–434. Association for Computational Linguistics.

Knowledge Base Completion via Coupled Path Ranking

Quan Wang[†], **Jing Liu**[‡], **Yuanfei Luo**[†], **Bin Wang**[†], **Chin-Yew Lin**[‡]
[†]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
{wangquan,luoyuanfei,wangbin}@iie.ac.cn
[‡]Microsoft Research, Beijing 100080, China
{liudani,cyl}@microsoft.com

Abstract

Knowledge bases (KBs) are often greatly incomplete, necessitating a demand for KB completion. The path ranking algorithm (PRA) is one of the most promising approaches to this task. Previous work on PRA usually follows a single-task learning paradigm, building a prediction model for each relation independently with its own training data. It ignores meaningful associations among certain relations, and might not get enough training data for less frequent relations. This paper proposes a novel multi-task learning framework for PRA, referred to as coupled PRA (CPRA). It first devises an agglomerative clustering strategy to automatically discover relations that are highly correlated to each other, and then employs a multi-task learning strategy to effectively couple the prediction of such relations. As such, CPRA takes into account relation association and enables implicit data sharing among them. We empirically evaluate CPRA on benchmark data created from Freebase. Experimental results show that CPRA can effectively identify coherent clusters in which relations are highly correlated. By further coupling such relations, CPRA significantly outperforms PRA, in terms of both predictive accuracy and model interpretability.

1 Introduction

Knowledge bases (KBs) like Freebase (Bollacker et al., 2008), DBpedia (Lehmann et al., 2014), and NELL (Carlson et al., 2010) are extremely useful resources for many NLP tasks (Cucerzan, 2007; Schuhmacher and Ponzetto, 2014). They provide large collections of facts about entities and their relations, typically stored as (*head entity*, *relation*, *tail entity*) triples, *e.g.*, (`Paris`, `capitalOf`, `France`). Although such KBs can be impressively large, they are still quite incomplete and missing crucial facts, which may reduce their usefulness in downstream tasks (West et al., 2014; Choi et al., 2015). KB completion, *i.e.*, automatically inferring missing facts by examining existing ones, has thus attracted increasing attention. Approaches to this task roughly fall into three categories: (i) path ranking algorithms (PRA) (Lao et al., 2011); (ii) embedding techniques (Bordes et al., 2013; Guo et al., 2015); and (iii) graphical models such as Markov logic networks (MLN) (Richardson and Domingos, 2006). This paper focuses on PRA, which is easily interpretable (as opposed to embedding techniques) and requires no external logic rules (as opposed to MLN).

The key idea of PRA is to explicitly use paths connecting two entities to predict potential relations between them. In PRA, a KB is encoded as a graph which consists of a set of heterogeneous edges. Each edge is labeled with a relation type that exists between two entities. Given a specific relation, random walks are first employed to find paths between two entities that have the given relation. Here a path is a sequence of relations linking two entities, *e.g.*, $h \xrightarrow{\texttt{bornIn}} e \xrightarrow{\texttt{capitalOf}} t$. These paths are then used as features in a binary classifier to predict if new instances (*i.e.*, entity pairs) have the given relation.

While KBs are naturally composed of multiple relations, PRA models these relations separately during the inference phase, by learning an individual classifier for each relation. We argue, however, that it will be beneficial for PRA to model certain relations in a collective way, particularly when the relations are closely related to each other. For example, given two relations `bornIn` and `livedIn`,

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1308–1318,
Berlin, Germany, August 7-12, 2016.

there must be a lot of paths (features) that are predictive for both relations, *e.g.*, $h \xrightarrow{\texttt{nationality}} e \xrightarrow{\texttt{hasCapital}} t$. These features make the corresponding relation classification tasks highly related. Numerous studies have shown that learning multiple related tasks simultaneously (a.k.a. multi-task learning) usually leads to better predictive performance, profiting from the relevant information available in different tasks (Carlson et al., 2010; Chapelle et al., 2010).

This paper proposes a novel multi-task learning framework that couples the path ranking of multiple relations, referred to as coupled PRA (CPRA). The new model needs to answer two critical questions: (i) *which relations should be coupled*, and (ii) *in what manner they should be coupled*.

As to the first question, it is obvious that not all relations are suitable to be learned together. For instance, modeling `bornIn` together with `hasWife` might not bring any real benefits, since there are few common paths between these two relations. CPRA introduces a common-path based similarity measure, and accordingly devises an agglomerative clustering strategy to group relations. Only relations that are grouped into the same cluster will be coupled afterwards.

As to the second question, CPRA follows the common practice of multi-task learning (Evgeniou and Pontil, 2004), and couples relations by using classifiers with partially shared parameters. Given a cluster of relations, CPRA builds the classifiers upon (i) relation-specific parameters to address the specifics of individual relations, and (ii) shared parameters to model the commonalities among different relations. These two types of parameters are balanced by a coupling coefficient, and learned jointly for all relations. In this way CPRA couples the classification tasks of multiple relations, and enables implicit data sharing and regularization.

The major contributions of this paper are as follows. (i) We design a novel framework for multi-task learning with PRA, *i.e.*, CPRA. To the best of our knowledge, this is the first study on multi-task PRA. (ii) We empirically verify the effectiveness of CPRA on a real-world, large-scale KB. Specifically, we evaluate CPRA on benchmark data created from Freebase. Experimental results show that CPRA can effectively identify coherent clusters in which relations are highly correlated. By further coupling such relations, CPRA substantially outperforms PRA, in terms of not only predictive accuracy but also model interpretability. (iii) We compare CPRA and PRA to the embedding-based TransE model (Bordes et al., 2013), and demonstrate their superiority over TransE. As far as we know, this is the first work that formally compares PRA-style approaches to embedding-based ones, on publicly available Freebase data.

In the remainder of this paper, we first review related work in Section 2, and formally introduce PRA in Section 3. We then detail the proposed CPRA framework in Section 4. Experiments and results are reported in Section 5, followed by the conclusion and future work in Section 6.

2 Related Work

We first review three lines of related work: (i) KB completion, (ii) PRA and its extensions, and (iii) multi-task learning, and then discuss the connection between CPRA and previous approaches.

KB completion. This task is to automatically infer missing facts from existing ones. Prior work roughly falls into three categories: (i) path ranking algorithms (PRA) which use paths that connect two entities to predict potential relations between them (Lao et al., 2011; Lao and Cohen, 2010); (ii) embedding-based models which embed entities and relations into a latent vector space and make inferences in that space (Nickel et al., 2011; Bordes et al., 2013); (iii) probabilistic graphical models such as the Markov logic network (MLN) and its variants (Pujara et al., 2013; Jiang et al., 2012). This paper focuses on PRA, since it is easily interpretable (as opposed to embedding-based models) and requires no external logic rules (as opposed to MLN and its variants).

PRA and its extensions. PRA is a random walk inference technique designed for predicting new relation instances in KBs, first proposed by Lao and Cohen (2010). Recently various extensions have been explored, ranging from incorporating a text corpus as additional evidence during inference (Gardner et al., 2013; Gardner et al., 2014), to introducing better schemes to generate more predictive paths (Gardner and Mitchell, 2015; Shi and Weninger, 2015), or using PRA in a broader context such as Google's Knowledge Vault (Dong et al., 2014). All these approaches are based on some single-task version of PRA, while our work explores multi-task learning for it.

Multi-task learning. Numerous studies have shown that learning multiple related tasks simulta-

neously can provide significant benefits relative to learning them independently (Caruana, 1997). A key ingredient of multi-task learning is to model the notion of task relatedness, through either parameter sharing (Evgeniou and Pontil, 2004; Ando and Zhang, 2005) or feature sharing (Argyriou et al., 2007; He et al., 2014). In recent years, there has been increasing work showing the benefits of multi-task learning in NLP-related tasks, such as relation extraction (Jiang, 2009; Carlson et al., 2010) and machine translation (Sennrich et al., 2013; Cui et al., 2013; Dong et al., 2015). This paper investigates the possibility of multi-task learning with PRA, in a parameter sharing manner.

Connection with previous methods. Actually, modeling multiple relations collectively is a common practice in embedding-based approaches. In such a method, embeddings are learned jointly for all relations, over a set of shared latent features (entity embeddings), and hence can capture meaningful associations among different relations. As shown by (Toutanova and Chen, 2015), observed features such as PRA paths usually perform better than latent features for KB completion. In this context, CPRA is designed in a way that gets the multi-relational benefit of embedding techniques while keeping PRA-style path features. Nickel et al. (2014) and Neelakantan et al. (2015) have tried similar ideas. However, their work focuses on improving embedding techniques with observed features, while our approach aims at improving PRA with multi-task learning.

3 Path Ranking Algorithm

PRA was first proposed by Lao and Cohen (2010), and later slightly modified in various ways (Gardner et al., 2014; Gardner and Mitchell, 2015). The key idea of PRA is to explicitly use paths that connect two entities as features to predict potential relations between them. Here a path is a sequence of relations $\langle r_1, r_2, \cdots, r_\ell \rangle$ that link two entities. For example, $\langle \texttt{bornIn}, \texttt{capitalOf} \rangle$ is a path linking `SophieMarceau` to `France`, through an intermediate node `Paris`. Such paths are then used as features to predict the presence of specific relations, *e.g.*, `nationality`. A typical PRA model consists of three steps: feature extraction, feature computation, and relation-specific classification.

Feature extraction. The first step is to generate and select path features that are potentially useful for predicting new relation instances. To this end, PRA first encodes a KB as a multi-relation graph. Given a pair of entities (h, t), PRA then finds the paths by performing random walks over the graph, recording those starting from h and ending at t with bounded lengths. More exhaustive strategies like breadth-first (Gardner and Mitchell, 2015) or depth-first (Shi and Weninger, 2015) search could also be used to enumerate the paths. After that a set of paths are selected as features, according to some precision-recall measure (Lao et al., 2011), or simply frequency (Gardner et al., 2014).

Feature computation. Once path features are selected, the next step is to compute their values. Given an entity pair (h, t) and a path π, PRA computes the feature value as a random walk probability $p(t|h, \pi)$, *i.e.*, the probability of arriving at t given a random walk starting from h and following exactly all relations in π. Computing these random walk probabilities could be at great expense. Gardner and Mitchell (2015) recently showed that such probabilities offer no discernible benefits. So they just used a binary value to indicate the presence or absence of each path. Similarly, Shi and Weninger (2015) used the frequency of a path as its feature value. Besides paths, other features such as path bigrams and vector space similarities could also be incorporated (Gardner et al., 2014).

Relation-specific classification. The last step of PRA is to train an individual classifier for each relation, so as to judge whether two entities should be linked by that relation. Given a relation and a set of training instances (*i.e.*, pairs of entities that are linked by the relation or not, with features selected and computed as above), one can use any kind of classifier to train a model. Most previous work simply chooses logistic regression.

4 Coupled Path Ranking Algorithm

As we can see, PRA (as well as its variants) follows a single-task learning paradigm, which builds a classifier for each relation independently with its own training data. We argue that such a single-task strategy might not be optimal for KB completion: (i) by learning the classifiers independently, it fails to discover and leverage meaningful associations among different relations; (ii) it might not perform well on less frequent relations for which only a few training instances are available. This section presents coupled PRA (CPRA), a novel multi-task learning framework that couples the path ranking of multiple relations. Through a multi-task strat-

egy, CPRA takes into account relation association and enables implicit data sharing among them.

4.1 Problem Formulation

Suppose we are given a KB containing a collection of triples $\mathcal{O} = \{(h, r, t)\}$. Each triple is composed of two entities $h, t \in \mathcal{E}$ and their relation $r \in \mathcal{R}$, where $\mathcal{E}$ is the entity set and $\mathcal{R}$ the relation set. The KB is then encoded as a graph $\mathcal{G}$, with entities represented as nodes, and triple (h, r, t) a directed edge from node h to node t. We formally define KB completion as a binary classification problem. That is, given a particular relation r, for any entity pair (h, t) such that $(h, r, t) \notin \mathcal{O}$, we would like to judge whether h and t should be linked by r, by exploiting the graph structure of $\mathcal{G}$. Let $\overline{\mathcal{R}} \subseteq \mathcal{R}$ denote a set of relations to be predicted.

Each relation $r \in \overline{\mathcal{R}}$ is associated with a set of training instances. Here a training instance is an entity pair (h, t), with a positive label if $(h, r, t) \in \mathcal{O}$ or a negative label otherwise.[1] For each of the entity pairs, path features could be extracted and computed using techniques described in Section 3. We denote by $\mathbf{\Pi}_r$ the set of path features extracted for relation r, and define its training set as $\mathcal{T}_r = \{(\mathbf{x}_{ir}, y_{ir})\}$. Here $\mathbf{x}_{ir}$ is the feature vector for an entity pair, with each dimension corresponding to a path $\pi \in \mathbf{\Pi}_r$, and $y_{ir} = \pm 1$ is the label. Note that our primary goal is to verify the possibility of multi-task learning with PRA. It is beyond the scope of this paper to further explore better feature extraction or computation.

Given the relations and their training instances, CPRA performs KB completion using a multi-task learning strategy. It consists of two components: relation clustering and relation coupling. The former automatically discovers highly correlated relations, and the latter further couples the learning of these relations, described in detail as follows.

4.2 Relation Clustering

It is obvious that not all relations are suitable to be coupled. We propose an agglomerative clustering algorithm to automatically discover relations that are highly correlated and should be learned together. Our intuition is that relations sharing more common paths (features) are probably more similar in classification, and hence should be coupled.

Specifically, we start with $|\overline{\mathcal{R}}|$ clusters and each cluster contains a single relation $r \in \overline{\mathcal{R}}$. Here $|\cdot|$ is the cardinality of a set. Then we iteratively merge the most similar clusters, say $\mathcal{C}_m$ and $\mathcal{C}_n$, into a new cluster $\mathcal{C}$. The similarity between two clusters is defined as:

$$\text{Sim}(\mathcal{C}_i, \mathcal{C}_j) = \frac{|\mathbf{\Pi}_{\mathcal{C}_i} \cap \mathbf{\Pi}_{\mathcal{C}_j}|}{\min(|\mathbf{\Pi}_{\mathcal{C}_i}|, |\mathbf{\Pi}_{\mathcal{C}_j}|)}, \qquad (1)$$

where $\mathbf{\Pi}_{\mathcal{C}_i}$ is the feature set associated with cluster $\mathcal{C}_i$ (if $\mathcal{C}_i$ contains a single relation, $\mathbf{\Pi}_{\mathcal{C}_i}$ the feature set associated with that relation). It essentially measures the overlap between two feature sets. The larger the overlap is, the higher the similarity will be. Once two clusters are merged, we update the feature set associated with the new cluster: $\mathbf{\Pi}_{\mathcal{C}} = \mathbf{\Pi}_{\mathcal{C}_m} \cup \mathbf{\Pi}_{\mathcal{C}_n}$. The algorithm stops when the highest cluster similarity is below some predefined threshold δ. This paper empirically sets $\delta = 0.5$. As such, relations sharing a substantial number of common paths are grouped into the same cluster.

4.3 Relation Coupling

After clustering, the next step of CPRA is to couple the path ranking of different relations within each cluster, *i.e.*, to learn the classification tasks for these relations simultaneously. We employ a multi-task classification algorithm similar to (Evgeniou and Pontil, 2004), and learn the classifiers jointly in a parameter sharing manner.

Consider a cluster containing K relations $\mathcal{C} = \{r_1, r_2, \cdots, r_K\}$. Recall that during the clustering phase a shared feature set has been generated for that cluster, *i.e.*, $\mathbf{\Pi}_{\mathcal{C}} = \mathbf{\Pi}_{r_1} \cup \cdots \cup \mathbf{\Pi}_{r_K}$. We first reform the training instances for the K relations using this shared feature set, so that all training data is represented in the same space.[2] We denote by $\mathcal{T}_k = \{(\mathbf{x}_{ik}, y_{ik})\}_{i=1}^{N_k}$ the reformed training data associated with the k-th relation. Then our goal is to jointly learn K classifiers $f_1, f_2, \cdots, f_K$ such that $f_k(\mathbf{x}_{ik}) \approx y_{ik}$.

We first assume that the classifier for each relation has a linear form $f_k(\mathbf{x}) = \mathbf{w}_k \cdot \mathbf{x} + b_k$, where $\mathbf{w}_k \in \mathbb{R}^d$ is the weight vector and b_k the bias. To model associations among different relations, we further assume that all $\mathbf{w}_k$ and b_k can be written, for every $k \in \{1, \cdots, K\}$, as:

$$\mathbf{w}_k = \mathbf{w}_0 + \mathbf{v}_k \ \text{ and } \ b_k = b_0. \qquad (2)$$

Here the shared $\mathbf{w}_0$ is used to model the commonalities among different relations, and the relation-specific $\mathbf{v}_k$ to address the specifics of individual

[1] We will introduce the details of generating negative training instances in Section 5.1.

[2] Note that $\mathbf{\Pi}_{r_k} \subseteq \mathbf{\Pi}_{\mathcal{C}}$. We just assign zero values to features that are contained in $\mathbf{\Pi}_{\mathcal{C}}$ but not in $\mathbf{\Pi}_{r_k}$.

relations. If the relations are closely related ($\mathbf{v}_k \approx \mathbf{0}$), they will have similar weights ($\mathbf{w}_t \approx \mathbf{w}_0$) on the common paths. We use the same bias b_0 for all the relations.[3]

We estimate $\mathbf{v}_k$, $\mathbf{w}_0$, and b_0 simultaneously in a joint optimization problem, defined as follows.

Problem 1 *CPRA amounts to solving the general optimization problem:*

$$\min_{\{\mathbf{v}_k\},\mathbf{w}_0,b_0} \sum_{k=1}^{K}\sum_{i=1}^{N_k} \ell\left(\mathbf{x}_{ik}, y_{ik}\right) + \frac{\lambda_1}{K}\sum_{k=1}^{K}\left\|\mathbf{v}_k\right\|_2^2 + \lambda_2\left\|\mathbf{w}_0\right\|_2^2,$$

where $\ell\left(\mathbf{x}_{ik}, y_{ik}\right)$ is the loss on a training instance. It can be instantiated into a logistic regression (LR) or support vector machine (SVM) version, by respectively defining the loss $\ell\left(\mathbf{x}_{ik}, y_{ik}\right)$ as:

$$\ell\left(\mathbf{x}_{ik}, y_{ik}\right) = \log\left(1+\exp\left(-y_{ik}f_k(\mathbf{x}_{ik})\right)\right),$$
$$\ell\left(\mathbf{x}_{ik}, y_{ik}\right) = \left[1-y_{ik}f_k(\mathbf{x}_{ik})\right]_+,$$

where $f_k(\mathbf{x}_{ik}) = (\mathbf{w}_0+\mathbf{v}_k)\cdot\mathbf{x}_{ik}+b_0$. We call them CPRA-LR and CPRA-SVM respectively.

In this problem, λ_1 and λ_2 are regularization parameters. By adjusting their values, we control the degree of parameter sharing among different relations. The larger the ratio $\frac{\lambda_1}{\lambda_2}$ is, the more we believe that all $\mathbf{w}_t$ should conform to the common model $\mathbf{w}_0$, and the smaller the relation-specific weight $\mathbf{v}_t$ will be.

The multi-task learning problem can be directly linked to a standard single-task learning one, built on all training data from different relations.

Proposition 1 *Suppose the training data associated with the k-th relation, for every $k=1,\cdots,K$, is transformed into:*

$$\widetilde{\mathbf{x}}_{ik} = [\frac{\mathbf{x}_{ik}}{\sqrt{\rho K}}, \underbrace{\mathbf{0},\cdots,\mathbf{0}}_{k-1}, \mathbf{x}_{ik}, \underbrace{\mathbf{0},\cdots,\mathbf{0}}_{K-k}],$$

where $\mathbf{0}\in\mathbb{R}^d$ is a vector whose coordinates are all zero, and $\rho=\frac{\lambda_2}{\lambda_1}$ a coupling coefficient. Consider a linear classifier for the transformed data $\widetilde{f}(\widetilde{\mathbf{x}}) = \widetilde{\mathbf{w}}\cdot\widetilde{\mathbf{x}}+\widetilde{b}$, with $\widetilde{\mathbf{w}}$ and $\widetilde{b}$ constructed as:

$$\widetilde{\mathbf{w}} = [\sqrt{\rho K}\mathbf{w}_0, \mathbf{v}_1, \cdots, \mathbf{v}_K] \;\; \textit{and} \;\; \widetilde{b} = b_0.$$

Then the objective function of Problem 1 is equivalent to:

$$\mathcal{L} = \sum_{k=1}^{K}\sum_{i=1}^{N_k} \widetilde{\ell}\left(\widetilde{\mathbf{x}}_{ik}, y_{ik}\right) + \widetilde{\lambda}\left\|\widetilde{\mathbf{w}}\right\|_2^2,$$

where $\widetilde{\ell}=\log(1+\exp(-y_{ik}\widetilde{f}(\widetilde{\mathbf{x}}_{ik})))$ is a logistic loss for CPRA-LR, and $\widetilde{\ell}=[1-y_{ik}\widetilde{f}(\widetilde{\mathbf{x}}_{ik})]_+$ a hinge loss for CPRA-SVM; and $\widetilde{\lambda}=\frac{\lambda_1}{K}$.

That means, after transforming data from different relations into a unified representation, Problem 1 is equivalent to a standard single-task learning problem, built on the transformed data from all the relations. So it can easily be solved by existing tools such as LR or SVM.

5 Experiments

In this section we present empirical evaluation of CPRA in the KB completion task.

5.1 Experimental Setups

We create our data on the basis of FB15K (Bordes et al., 2011)[4], a relatively dense subgraph of Freebase containing 1,345 relations and the corresponding triples.

KB graph construction. We notice that in most cases FB15K encodes a relation and its reverse relation at the same time. That is, once a new fact is observed, FB15K creates two triples for it, *e.g.*, (`x`, `film/edited-by`, `y`) and (`y`, `editor/film`, `x`). Reverse relations provide no additional knowledge. They may even hurt the performance of PRA-style methods. Actually, to enhance graph connectivity, PRA-style methods usually automatically add an inverse version for each relation in a KB (Lao and Cohen, 2010; Lao et al., 2011). That is, for each observed triple (h,r,t), another triple (t,r^{-1},h) is constructed and added to the KB. Consider the prediction of a relation, say `film/edited-by`. In the training phase, we could probably find that every two entities connected by this relation are also connected by the path `editor/film`$^{-1}$, and hence assign an extremely high weight to it.[5] However, in the testing phase, for any entity pair (`x`, `y`) such that (`y`, `editor/film`, `x`) has not been encoded, we might not even find that path and hence could always make a negative prediction.[6]

For this reason, we remove reverse relations in FB15K. Specifically, we regard r_2 to be a reverse relation of r_1 if the triple (t,r_2,h) holds whenever (h,r_1,t) is observed, and we randomly discard

[3] It implicitly assumes that all the relations have the same proportion of positive instances. This assumption actually holds since given any relation we can always generate the same number of negative instances for each positive one. We set this number to 4 in our experiments.

[4] https://everest.hds.utc.fr/doku.php?id=en:smemlj12

[5] For every observed triple (`x`, `film/edited-by`, `y`), FB15K also encodes (`y`, `editor/film`, `x`), for which (`x`, `editor/film`$^{-1}$, `y`) is further constructed.

[6] Note that such test cases are generally more meaningful: if we already know (`y`, `editor/film`, `x`), predicting (`x`, `film/edited-by`, `y`) could be trivial.

one of the two relations.[7] As such, we keep 774 out of 1,345 relations in FB15K, covering 14,951 entities and 327,783 triples. Then we build a graph based on this data and use it as input to CPRA (and our baseline methods).

Labeled instance generation. We select 171 relations to test our methods. To do so, we pick 10 popular domains, including award, education, film, government, location, music, olympics, organization, people, and tv. Relations in these domains with at least 50 triples observed for them are selected. For each of the 171 relations, we split the associated triples into roughly 80% training, 10% validation, and 10% testing. Since the triple number varies significantly among the relations, we allow at most 200 validation/testing triples for each relation, so as to make the test cases as balanced as possible. Note that validation and testing triples are not used for constructing the graph.

We generate positive instances for each relation directly from these triples. Given a relation r and a triple (h, r, t) observed for it (training, validation, or testing), we take the pair of entities (h, t) as a positive instance for that relation. Then we follow (Shi and Weninger, 2015; Krompaß et al., 2015) to generate negative instances. Given each positive instance (h, t) we generate four negative ones, two by randomly corrupting the head h, and the other two the tail t. To make the negative instances as difficult as possible, we corrupt a position using only entities that have appeared in that position. That means, given the relation `capitalOf` and the positive instance (`Paris`, `France`), we could generate a negative instance (`Paris`, `UK`) but never (`Paris`, `NBA`), since `NBA` never appears as a tail entity of the relation. We further ensure that the negative instances do not overlap with the positive ones.

Feature extraction and computation. Given the labeled instances, we extract path features for them using the code provided by Shi and Weninger (2015)[8]. It is a depth-first search strategy that enumerates all paths between two entities. We set the maximum path length to be $\ell = 3$. There are about 8.2% of the labeled instances for which no path could be extracted. We remove such cases, giving on average about 5,250 training, 323 validation, and 331 testing instances per relation. Then we remove paths that appear only once in each relation, getting 5,515 features on average per relation. We simply compute the value of each feature as its frequency in an instance. Table 1 lists the statistics of the data used in our experiments.

# Relations	774
# Entities	14,951
# Triples	327,783
# Relations tested	171
# Avg. training instances/relation	5,250
# Avg. validation instances/relation	323
# Avg. testing instances/relation	331
# Avg. features/relation	5,515

Table 1: Statistics of the data.

Evaluation metrics. As evaluation metrics, we use mean average precision (MAP) and mean reciprocal rank (MRR), following recent work evaluating KB completion performance (West et al., 2014; Gardner and Mitchell, 2015). Both metrics evaluate some ranking process: if a method ranks the positive instances before the negative ones for each relation, it will get a high MAP or MRR.

Baseline methods. We compare CPRA to traditional single-task PRA. CPRA first groups the 171 relations into clusters, and then learns classifiers jointly for relations within the same cluster. We implement two versions of it: CPRA-LR and CPRA-SVM. As we have shown in Proposition 1, both of them could be solved by standard classification tools. PRA learns an individual classifier for each of the relations, using LR or SVM classification techniques, denoted by PRA-LR or PRA-SVM. We use LIBLINEAR (Fan et al., 2008)[9] to solve the LR and SVM classification problems. For all these methods, we tune the cost c in the range of $\{2^{-5}, 2^{-4}, \cdots, 2^4, 2^5\}$. And we set the coupling coefficient $\rho = \frac{\lambda_2}{\lambda_1}$ in CPRA in the range of $\{0.1, 0.2, 0.5, 1, 2, 5, 10\}$.

We further compare CPRA to TransE, a widely adopted embedding-based method (Bordes et al., 2013). TransE learns vector representations for entities and relations (*i.e.*, embeddings), and uses the learned embeddings to determine the plausibility of missing facts. Such plausibility can then be used to rank the labeled instances. We implement TransE using the code provided by Bordes et al. (2013)[10]. To learn embeddings, we take as input the triples used to construct the graph (from which CPRA and PRA extract their paths). We tune the embedding dimension in $\{20, 50, 100\}$, the margin in $\{0.1, 0.2, 0.5, 1, 2, 5\}$, and the learning rate

[7] We still add an inverse version for the relation kept during path extraction.

[8] https://github.com/nddsg/KGMiner

[9] http://www.csie.ntu.edu.tw/ cjlin/liblinear

[10] https://github.com/glorotxa/SME

film/casting-director	gov-jurisdiction/dist-represent
film/cinematography	location/contain
film/costume-design-by	location/adjoin
film/art-direction-by	us-county/county-seat
film/crewmember	county-place/county
film/set-decoration-by	location/partially-contain
film/production-design-by	region/place-export
film/edited-by	
film/written-by	
film/story-by	
org/place-founded	country/divisions
org/headquarter-city	country/capital
org/headquarter-state	country/fst-level-divisions
org/geographic-scope	country/snd-level-divisions
org/headquarter-country	admin-division/capital
org/service-location	
tv/tv-producer	music-group-member/instrument
tv/recurring-writer	music-artist/recording-role
tv/program-creator	music-artist/track-role
tv/regular-appear-person	music-group-member/role
tv/tv-actor	

Table 2: Six largest clusters of relations (with the stopping criterion $\delta = 0.5$).

in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$. For details please refer to (Bordes et al., 2013). For each of these methods, we select the optimal configuration that leads to the highest MAP on the validation set and report its performance on the test set.

5.2 Relation Clustering Results

We first test the effectiveness of our agglomerative strategy (Section 4.2) in relation clustering. With the stopping criterion $\delta = 0.5$, 96 out of the 171 relations are grouped into clusters which contain at least two relations. Each of these 96 relations will later be learned jointly with some other relations. The other 75 relations cannot be merged, and will still be learned individually. Table 2 shows the six largest clusters discovered by our algorithm. Relations in each cluster are arranged in the order they were merged. The results indicate that our algorithm can effectively identify coherent clusters in which relations are highly correlated to each other. For example, the top left cluster describes relations between a film and its crew members, and the middle left between an organization and a location.

During clustering we might obtain clusters that contain too many relations and hence too many training instances for our CPRA model to learn efficiently. We split such clusters into sub-clusters, either according to the domain (*e.g.*, the film cluster and tv cluster) or randomly (*e.g.*, the two location clusters on the top right).

5.3 KB Completion Results

We further test the effectiveness of our multi-task learning strategy (Section 4.3) in KB completion. Table 3 gives the results on the 96 relations that are actually involved in multi-tasking learning (*i.e.*, grouped into clusters with size larger than one).[11] The 96 relations are grouped into 29 clusters, and relations within the same cluster are learned jointly. Table 3 reports (i) MAP and MRR within each cluster and (ii) overall MAP and MRR on the 96 relations. Numbers marked in bold type indicate that CPRA-LR/SVM outperforms PRA-LR/SVM, within a cluster (with its ID listed in the first column) or on all the 96 relations (ALL). We judge statistical significance of the overall improvements achieved by CPRA-LR/SVM over PRA-LR/SVM and TransE, using a paired t-test. The average precision (or reciprocal rank) on each relation is used as paired data. The symbol "$**$" indicates a significance level of $p < 0.0001$, and "$*$" a significance level of $p < 0.05$.

From the results, we can see that (i) CPRA outperforms PRA (using either LR or SVM) and TransE on the 96 relations (ALL) in both metrics. All the improvements are statistically significant, with a significance level of $p < 0.0001$ for MAP and a significance level of $p < 0.05$ for MRR. (ii) CPRA-LR/SVM outperforms PRA-LR/SVM in 22/24 out of the 29 clusters in terms of MAP. Most of the improvements are quite substantial. (iii) Improving PRA-LR and PRA-SVM in terms of MRR could be hard, since they already get the best performance (MRR = 1) in 19 out of the 29 clusters. But even so, CPRA-LR/SVM still improves 7/8 out of the remaining 10 clusters. (iv) The PRA-style methods perform substantially better than the embedding-based TransE model in most of the 29 clusters and on all the 96 relations. This observation demonstrates the superiority of observed features (*i.e.*, PRA paths) over latent features.

Table 4 further shows the top 5 most discriminative paths (*i.e.*, features with the highest weights) discovered by PRA-SVM (left) and CPRA-SVM (right) for each relation in the 6th cluster.[12] The average precision on each relation is also provid-

[11] The other 75 relations are still learned individually. So CPRA and PRA perform the same on these relations. The MAP values on these 75 relations are 0.6360, 0.6558, 0.6543 for TransE, PRA-LR, and PRA-SVM respectively, and the MRR values are 0.9049, 0.9033, and 0.9013 respectively.

[12] This is one of the largest clusters on which CPRA-SVM improves PRA-SVM substantially.

	MAP					MRR				
	TransE	PRA-LR	CPRA-LR	PRA-SVM	CPRA-SVM	TransE	PRA-LR	CPRA-LR	PRA-SVM	CPRA-SVM
1	0.5419	0.5160	**0.5408**	0.4687	**0.5204**	0.7500	0.8333	**1.0000**	0.7778	**0.8333**
2	0.7480	0.7888	0.7807	0.8010	**0.8092**	1.0000	1.0000	1.0000	1.0000	1.0000
3	0.4624	0.4625	**0.4788**	0.4634	0.4560	0.8333	1.0000	1.0000	1.0000	0.8333
4	0.5495	0.5378	**0.5423**	0.5385	**0.5460**	0.7667	0.6400	**0.7000**	0.7167	0.7000
5	0.5164	0.5789	**0.6030**	0.5891	**0.6072**	0.8333	0.6667	**1.0000**	0.8333	**1.0000**
6	0.6918	0.7733	**0.7950**	0.7369	**0.8084**	1.0000	0.8333	0.8333	0.8056	**0.9167**
7	0.7381	0.7531	**0.7754**	0.7456	0.7414	0.7500	1.0000	1.0000	1.0000	1.0000
8	0.4258	0.5180	**0.5446**	0.3162	**0.4606**	1.0000	1.0000	1.0000	0.3056	**0.7500**
9	0.6353	0.7879	0.7708	0.7680	**0.7685**	0.7500	1.0000	1.0000	1.0000	1.0000
10	0.8615	0.7773	0.7738	0.7618	0.7507	1.0000	1.0000	1.0000	1.0000	1.0000
11	0.4549	0.5814	**0.6014**	0.5717	**0.5896**	0.8333	1.0000	1.0000	0.8750	**1.0000**
12	0.6202	0.7187	**0.7479**	0.7455	**0.7457**	0.7500	0.5833	**1.0000**	1.0000	1.0000
13	0.5530	0.6681	**0.6716**	0.6373	**0.6502**	0.6667	1.0000	1.0000	1.0000	1.0000
14	0.5082	0.4360	**0.5280**	0.4715	**0.5806**	0.3750	0.6667	0.6250	1.0000	1.0000
15	0.9881	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
16	0.5324	0.6818	**0.6863**	0.6522	**0.6705**	0.8750	1.0000	1.0000	1.0000	1.0000
17	0.3759	0.3351	**0.3593**	0.3273	0.3219	0.6111	0.5667	**0.7778**	0.5111	**0.6667**
18	0.9423	0.9968	**1.0000**	0.9947	**0.9975**	1.0000	1.0000	1.0000	1.0000	1.0000
19	0.7903	0.8376	0.8310	0.8296	**0.8328**	0.8714	0.9286	0.8571	0.8571	0.8571
20	0.7920	0.8285	**0.8746**	0.8491	**0.8754**	1.0000	1.0000	1.0000	1.0000	1.0000
21	0.4885	0.5869	0.5799	0.5554	**0.5952**	0.6250	1.0000	1.0000	1.0000	1.0000
22	0.7894	0.8371	**0.8486**	0.8371	**0.8374**	1.0000	1.0000	1.0000	1.0000	1.0000
23	0.7123	0.7848	**0.8191**	0.7811	**0.7957**	0.9500	1.0000	1.0000	1.0000	1.0000
24	0.5982	0.7923	**0.8048**	0.8204	**0.8220**	1.0000	1.0000	1.0000	1.0000	1.0000
25	0.6223	0.8723	0.8723	0.7785	**0.8109**	0.7500	1.0000	1.0000	1.0000	1.0000
26	0.5253	0.5377	**0.5685**	0.5337	**0.5447**	0.8750	0.8125	**0.8750**	0.7083	**0.8333**
27	0.8763	0.6890	**0.8124**	0.7014	**0.8016**	1.0000	0.6667	**1.0000**	0.7500	**1.0000**
28	0.7588	0.8131	**0.8154**	0.8130	**0.8146**	1.0000	1.0000	1.0000	1.0000	1.0000
29	0.4894	0.5921	**0.6543**	0.6093	**0.6566**	0.7500	1.0000	1.0000	1.0000	1.0000
ALL	0.6540	0.7058	**0.7254****	0.6943	**0.7162****	0.8682	0.9061	**0.9436***	0.8982	**0.9358***

Table 3: KB completion results on the 96 relations that have been grouped into clusters with size larger than one (with the stopping criterion $\delta = 0.5$), and hence involved in multi-tasking learning.

ed. We can observe that (i) CPRA generally discovers more predictive paths than PRA. Almost all the top paths discovered by CPRA are easily interpretable and provide sensible reasons for the final prediction, while some of the top paths discovered by PRA are hard to interpret and less predictive. Take `org/place-founded` as an example. All the 5 CPRA paths are useful to predict the place where an organization was founded, *e.g.*, the 3rd one tells that "the organization headquarter in a city which is located in that place". However, the PRA path "common/class $\rightarrow$ common/class^{-1} $\rightarrow$ film/debut-venue" is hard to interpret and less predictive. (ii) For the 1st/4th/6th relation on which PRA gets a low average precision, CPRA learns almost completely different top paths and gets a substantially higher average precision. While for the other relations (2nd/3rd/5th) on which PRA already performs well enough, CPRA learns similar top paths and gets a comparable average precision. We have conducted the same analyses with CPRA-LR and PRA-LR, and observed similar phenomena. All these observations demonstrate the superiority of CPRA, in terms of not only predictive accuracy but also model interpretability.

6 Conclusion

In this paper we have studied the path ranking algorithm (PRA) from the viewpoint of multi-task learning. We have designed a novel multi-task learning framework for PRA, called coupled PRA (CPRA). The key idea of CPRA is to (i) automatically discover relations highly correlated to each other through agglomerative clustering, and (ii) effectively couple the prediction of such relations through multi-task learning. By coupling different relations, CPRA takes into account relation associations and enables implicit data sharing among them. We have tested CPRA on benchmark data created from Freebase. Experimental results show that CPRA can effectively identify coherent clusters in which relations are highly correlated. By further coupling such relations, CPRA significantly outperforms PRA, in terms of both predictive

org/place-founded (0.4920 vs. 0.6750)	
org/headquarter-city	location/contain^{-1}
common/class → common/class^{-1} → film/debut-venue	org/headquarter-city
common/class → common/class^{-1} → sports-team/location	org/headquarter-city → location/contain^{-1}
employer/job-title → employer/job-title^{-1} → location/contain	org/headquarter-state → location/contain
music-artist/label^{-1} → person/place-of-birth	org/headquarter-city → bibs-location/state
org/headquarter-city (0.9014 vs. 0.9141)	
location/contain^{-1}	location/contain^{-1}
org/place-founded	org/headquarter-state → location/contain
org/headquarter-state → location/contain	org/place-founded
org/child^{-1} → org/child → org/place-founded	org/child^{-1} → org/child → org/place-founded
sports-team/location	industry/company^{-1} → industry/company → org/place-founded
org/headquarter-state (0.9522 vs. 0.9558)	
location/contain^{-1}	location/contain^{-1}
org/headquarter-city → location/contain^{-1}	org/headquarter-city → location/contain^{-1}
org/headquarter-city → bibs-location/state	org/headquarter-city → bibs-location/state
org/headquarter-city → county-place/county → location/contain^{-1}	org/headquarter-city
org/headquarter-city → location/contain^{-1} → location/contain^{-1}	org/place-founded
org/geographic-scope (0.5252 vs. 0.6075)	
common/class → common/class^{-1} → location/vacationer^{-1}	location/contain^{-1}
common/class → common/class^{-1} → country/languages^{-1}	org/headquarter-city → location/contain^{-1}
common/class → common/class^{-1} → gov-jurisdiction/gov-body^{-1}	location/contain^{-1} → location/contain^{-1}
common/class → common/class^{-1} → region/currency-of-gdp^{-1}	org/place-founded → location/contain^{-1}
politician/party^{-1} → person/nationality → location/adjoins	org/headquarter-city → location/contain^{-1} → location/contain^{-1}
org/headquarter-country (0.9859 vs. 0.9938)	
org/headquarter-city → airline/city-served^{-1} → org/service-location	location/contain^{-1}
org/headquarter-city → admin-area/child^{-1} → region/place-export	org/headquarter-city → location/contain^{-1}
org/headquarter-city → country/divisions^{-1} → region/place-export	org/headquarter-city → county-place/county → location/contain^{-1}
org/headquarter-city → film/feat-location^{-1} → film/feat-location	location/contain^{-1} → location/contain^{-1}
org/headquarter-city → gov-jurisdiction/title → employer/job-title^{-1}	org/place-founded → location/contain^{-1}
org/service-location (0.5644 vs. 0.7044)	
org/headquarter-city → country/divisions^{-1}	org/headquarter-city → location/contain^{-1}
org-extra/service-location	org/headquarter-city → county-place/county → location/contain^{-1}
film/production-company^{-1} → film/subjects → admin-area/child^{-1}	location/contain^{-1} → location/contain^{-1}
org/legal-structure → entry/taxonomy → entry/taxonomy^{-1}	org/place-founded → location/contain^{-1}
airline/city-served → region/currency → region/currency-of-gdp^{-1}	org-extra/service-location

Table 4: Top paths given by PRA-SVM (left) and CPRA-SVM (right) for each relation in the 6th cluster.

accuracy and model interpretability.

This is the first work that investigates the possibility of multi-task learning with PRA, and we just provide a very simple solution. There are still many interesting topics to study. For instance, the agglomerative clustering strategy can only identify highly correlated relations, *i.e.*, those sharing a lot of common paths. Relations that are only loosely correlated, *e.g.*, those sharing no common paths but a lot of sub-paths, will not be identified. We would like to design new mechanisms to discover loosely correlated relations, and investigate whether coupling such relations still provides benefits. Another example is that the current method is a two-step approach, performing relation clustering first and then relation coupling. It will be interesting to study whether one can merge the clustering step and the coupling step so as to have a richer inter-task dependent structure. We will investigate such topics in our future work.

Acknowledgments

We would like to thank Baoxu Shi for providing the code for path extraction. We would also like to thank the anonymous reviewers for their valuable comments and suggestions. This work is supported by the National Natural Science Foundation of China (grant No. 61402465), the Strategic Priority Research Program of the Chinese Academy of Sciences (grant No. XDA06030200), and the Microsoft Research Asia StarTrack Program. This work was done when Quan Wang was a visiting researcher at Microsoft Research Asia.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Reseach*, 6:1817–1853.

Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. 2007. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, pages 41–48.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 301–306.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr, and Tom M. Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1306–1313.

Rich Caruana. 1997. Multitask learning. *Machine Learning*, 28(1):41–75.

Olivier Chapelle, Pannagadatta Shivaswamy, Srinivas Vadrevu, Kilian Weinberger, Ya Zhang, and Belle Tseng. 2010. Multi-task learning for boosting with application to web search ranking. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1189–1198.

Eunsol Choi, Tom Kwiatkowski, and Luke Zettlemoyer. 2015. Scalable semantic parsing with partial ontologies. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1311–1320.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 708–716.

Lei Cui, Xilun Chen, Dongdong Zhang, Shujie Liu, Mu Li, and Ming Zhou. 2013. Multi-domain adaptation for SMT using multi-task learning. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1055–1065.

Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 601–610.

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. 2015. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1723–1732.

Theodoros Evgeniou and Massimiliano Pontil. 2004. Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Matt Gardner and Tom Mitchell. 2015. Efficient and expressive knowledge base completion using subgraph feature extraction. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1488–1498.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 833–838.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 397–406.

Shu Guo, Quan Wang, Lihong Wang, Bin Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 84–94.

Jingrui He, Yan Liu, and Qiang Yang. 2014. Linking heterogeneous input spaces with pivots for multi-task learning. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 181–189.

Shangpu Jiang, Daniel Lowd, and Dejing Dou. 2012. Learning to refine an automatically extracted knowledge base using markov logic. In *Proceedings of the 2012 IEEE International Conference on Data Mining*, pages 912–917.

Jing Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1012–1020.

Denis Krompaß, Stephan Baier, and Volker Tresp. 2015. Type-constrained representation learning in knowledge graphs. In *Proceedings of the 13th International Semantic Web Conference*, pages 640–655.

Ni Lao and William W. Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, et al. 2014. Dbpedia: A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 156–166.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning*, pages 809–816.

Maximilian Nickel, Xueyan Jiang, and Volker Tresp. 2014. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems*, pages 1179–1187.

Jay Pujara, Hui Miao, Lise Getoor, and William Cohen. 2013. Knowledge graph identification. In *Proceedings of the 11th International Semantic Web Conference*, pages 542–557.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine Learning*, 62(1-2):107–136.

Michael Schuhmacher and Simone Paolo Ponzetto. 2014. Knowledge-based graph document modeling. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, pages 543–552.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 832–840.

Baoxu Shi and Tim Weninger. 2015. Fact checking in large knowledge graphs: A discriminative predict path mining approach. In *arXiv:1510.05911*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and Their Compositionality*.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 515–526.

Larger-Context Language Modelling with Recurrent Neural Network*

Tian Wang
Center for Data Science
New York University
t.wang@nyu.edu

Kyunghyun Cho
Courant Institute of Mathematical Sciences
and Center for Data Science
New York University
kyunghyun.cho@nyu.edu

Abstract

In this work, we propose a novel method to incorporate corpus-level discourse information into language modelling. We call this larger-context language model. We introduce a late fusion approach to a recurrent language model based on long short-term memory units (LSTM), which helps the LSTM unit keep intra-sentence dependencies and inter-sentence dependencies separate from each other. Through the evaluation on four corpora (IMDB, BBC, Penn TreeBank, and Fil9), we demonstrate that the proposed model improves perplexity significantly. In the experiments, we evaluate the proposed approach while varying the number of context sentences and observe that the proposed late fusion is superior to the usual way of incorporating additional inputs to the LSTM. By analyzing the trained larger-context language model, we discover that content words, including nouns, adjectives and verbs, benefit most from an increasing number of context sentences. This analysis suggests that larger-context language model improves the unconditional language model by capturing the theme of a document better and more easily.

1 Introduction

The goal of language modelling is to estimate the probability distribution of various linguistic units, e.g., words, sentences (Rosenfeld, 2000). Among the earliest techniques were count-based n-gram language models which intend to assign the probability distribution of a given word observed after a fixed number of previous words. Later Bengio et al. (2003) proposed feed-forward neural language model, which achieved substantial improvements in perplexity over count-based language models. Bengio et al. showed that this neural language model could simultaneously learn the conditional probability of the latest word in a sequence as well as a vector representation for each word in a predefined vocabulary.

Recently recurrent neural networks have become one of the most widely used models in language modelling (Mikolov et al., 2010). Long short-term memory unit (LSTM, Hochreiter and Schmidhuber, 1997) is one of the most common recurrent activation function. Architecturally, the memory state and output state are explicitly separated by activation gates such that the vanishing gradient and exploding gradient problems described in Bengio et al. (1994) is avoided. Motivated by such gated model, a number of variants of RNNs (e.g. Cho et al. (GRU, 2014b), Chung et al. (GF-RNN, 2015)) have been designed to easily capture long-term dependencies.

When modelling a corpus, these language models assume the mutual independence among sentences, and the task is often reduced to assigning a probability to a single sentence. In this work, we propose a method to incorporate corpus-level discourse dependency into neural language model. We call this larger-context language model. It models the influence of context by defining a conditional probability in the form of $P(w_n|w_{1:n-1}, S)$, where $w_1, ..., w_n$ are words from the same sentence, and S represents the context which consists a number of previous sentences of arbitrary length.

We evaluated our model on four different corpora (IMDB, BBC, Penn TreeBank, and Fil9). Our experiments demonstrate that the proposed larger-context language model improve perplex-

*Recently, (Ji et al., 2015) independently proposed a similar approach.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1319–1329,
Berlin, Germany, August 7-12, 2016.

ity for sentences, significantly reducing per-word perplexity compared to the language models without context information. Further, through Part-Of-Speech tag analysis, we discovered that content words, including nouns, adjectives and verbs, benefit the most from increasing number of context sentences. Such discovery led us to the conclusion that larger-context language model improves the unconditional language model by capturing the theme of a document.

To achieve such improvement, we proposed a late fusion approach, which is a modification to the LSTM such that it better incorporates the discourse context from preceding sentences. In the experiments, we evaluated the proposed approach against early fusion approach with various numbers of context sentences, and demonstrated the late fusion is superior to the early fusion approach.

Our model explores another aspect of context-dependent recurrent language model. It is novel in that it also provides an insightful way to feed information into LSTM unit, which could benefit all encoder-decoder based applications.

2 Statistical Language Modelling with Recurrent Neural Network

Given a document $D = (S_1, S_2, \ldots, S_L)$ which consists of L sentences, statistical language modelling aims at computing its probability $P(D)$. It is often assumed that each sentence in the whole document is mutually independent from each other:

$$P(D) \approx \prod_{l=1}^{L} P(S_l). \tag{1}$$

We call this probability (before approximation) a *corpus-level probability*. Under this assumption of mutual independence among sentences, the task of language modelling is often reduced to assigning a probability to a single sentence $P(S_l)$.

A sentence $S_l = (w_1, w_2, \ldots, w_{T_l})$ is a variable-length sequence of words or tokens. By assuming that a word at any location in a sentence is largely predictable by preceding words, we can rewrite the sentence probability into

$$P(S) = \prod_{t=1}^{T_l} p(w_t | w_{<t}), \tag{2}$$

where $w_{<t}$ denotes all the preceding words. We call this a *sentence-level probability*.

This rewritten probability expression can be either directly modelled by a recurrent neural network (Mikolov et al., 2010) or further approximated as a product of n-gram conditional probabilities such that

$$P(S) \approx \prod_{t=1}^{T_l} p(w_t | w_{t-(n-1)}^{t-1}), \tag{3}$$

where $w_{t-(n-1)}^{t-1} = (w_{t-(n-1)}, \ldots, w_{t-1})$. The latter is called *$n$-gram language modelling*.

A recurrent language model is composed of two functions–transition and output functions. The transition function reads one word w_t and updates its hidden state such that

$$\mathbf{h}_t = \phi\left(w_t, \mathbf{h}_{t-1}\right), \tag{4}$$

where $\mathbf{h}_0$ is an all-zero vector. ϕ is a recurrent activation function. For more details on widely-used recurrent activation units, we refer the reader to (Jozefowicz et al., 2015; Greff et al., 2015).

At each timestep, the output function computes the probability over all possible *next* words in the vocabulary V. This is done by

$$p(w_{t+1} = w' | w_1^t) \propto \exp\left(g_{w'}(\mathbf{h}_t)\right). \tag{5}$$

g is commonly an affine transformation:

$$g(\mathbf{h}_t) = \mathbf{W}_o \mathbf{h}_t + \mathbf{b}_o,$$

where $\mathbf{W}_o \in \mathbb{R}^{|V| \times d}$ and $\mathbf{b}_o \in \mathbb{R}^{|V|}$.

The whole model is trained by maximizing the log-likelihood of a training corpus often using stochastic gradient descent with backpropagation through time (see, e.g., Rumelhart et al., 1988).

This conventional approach to statistical language modelling often treats every sentence in a document to be independent from each other This is often due to the fact that downstream tasks, such as speech recognition and machine translation, are done sentence-wise. In this paper, we ask how strong an assumption this is, how much impact this assumption has on the final language model quality and how much gain language modelling can get by making this assumption less strong.

Long Short-Term Memory Here let us briefly describe a long short-term memory unit which is widely used as a recurrent activation function ϕ (see Eq. (4)) for language modelling (see, e.g., Graves, 2013).

A layer of long short-term memory (LSTM) unit consists of three gates and a single memory cell. They are computed by

$$\begin{aligned}\mathbf{i}_t =& \sigma\left(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i\right)\\ \mathbf{o}_t =& \sigma\left(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o\right)\\ \mathbf{f}_t =& \sigma\left(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f\right),\end{aligned}$$

where σ is a sigmoid function. $\mathbf{x}_t$ is the input at time t. The memory cell is computed by

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_c\mathbf{x} + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c\right),$$

where $\odot$ is an element-wise multiplication. This adaptive leaky integration of the memory cell allows the LSTM to easily capture long-term dependencies in the input sequence.

The output, or the activation of this LSTM layer, is then computed by $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$.

3 Larger-Context Language Modelling

In this paper, we aim not at improving the sentence-level probability estimation $P(S)$ (see Eq. (2)) but at improving the corpus-level probability $P(D)$ from Eq. (1) directly. One thing we noticed at the beginning of this work is that it is not necessary for us to make the assumption of mutual independence of sentences in a corpus. Rather, similarly to how we model a sentence probability, we can loosen this assumption by

$$P(D) \approx \prod_{l=1}^{L} P(S_l|S_{l-n}^{l-1}), \quad (6)$$

where $S_{l-n}^{l-1} = (S_{l-n}, S_{l-n+1}, \ldots, S_{l-1})$. n decides on how many preceding sentences each conditional sentence probability conditions on, similarly to what happens with a usual n-gram language modelling.

From the statistical modelling's perspective, estimating the corpus-level language probability in Eq. (6) is equivalent to build a statistical model that approximates

$$P(S_l|S_{l-n}^{l-1}) = \prod_{t=1}^{T_l} p(w_t|w_{<t}, S_{l-n}^{l-1}), \quad (7)$$

similarly to Eq. (2). One major difference from the existing approaches to statistical language modelling is that now each conditional probability of a next word is conditioned not only on the preceding words in the same sentence, but also on the $n-1$ preceding *sentences*.

A conventional, count-based n-gram language model is not well-suited due to the issue of data sparsity. In other words, the number of rows in the table storing n-gram statistics will explode as the number of possible sentence combinations grows exponentially with respect to both the vocabulary size, each sentence's length and the number of context sentences.

Either neural or recurrent language modelling however does not suffer from this issue of data sparsity. This makes these models ideal for modelling the *larger-context sentence probability* in Eq. (7). More specifically, we are interested in adapting the recurrent language model for this.

In doing so, we answer two questions in the following subsections. First, there is a question of how we should represent the context sentences S_{l-n}^{l-1}. We consider two possibilities in this work. Second, there is a large freedom in how we build a recurrent activation function to be conditioned on the context sentences. We also consider two alternatives in this case.

3.1 Context Representation

A sequence of preceding sentences can be represented in many different ways. Here, let us describe two alternatives we test in the experiments.

The first representation is to simply bag all the words in the preceding sentences into a single vector $\mathbf{s} \in [0, 1]^{|V|}$. Any element of $\mathbf{s}$ corresponding to the word that exists in one of the preceding sentences will be assigned the frequency of that word, and otherwise 0. This vector is multiplied from left by a matrix $\mathbf{P}$ which is tuned together with all the other parameters: $\mathbf{p} = \mathbf{P}\mathbf{s}$. We call this representation $\mathbf{p}$ a *bag-of-words (BoW) context*.

Second, we try to represent the preceding context sentences as a sequence of bag-of-words. Each bag-of-word $\mathbf{s}_j$ is the bag-of-word representation of the j-th context sentence, and they are put into a sequence $(\mathbf{s}_{l-n}, \ldots, \mathbf{s}_{l-1})$. Unlike the first BoW context, this allows us to incorporate the order of the preceding context sentences.

This sequence of BoW vectors are read by a recurrent neural network which is separately from the one used for modelling a sentence (see Eq. (4).) We use LSTM units as recurrent activations, and for each context sentence in the sequence, we get $\mathbf{z}_t = \phi\left(\mathbf{x}_t, \mathbf{z}_{t-1}\right)$, for $t = l-n, \ldots, l-1$. We set the last hidden state $\mathbf{z}_{l-1}$ of this *context recurrent neural network* as the con-

text vector $\mathbf{p}$.

Attention-based Context Representation The sequence of BoW vectors can be used in a bit different way from the above. Instead of a unidirectional recurrent neural network, we first use a bidirectional recurrent neural network to read the sequence. The forward recurrent neural network reads the sequence as usual in a forward direction, and the reverse recurrent neural network in the opposite direction. The hidden states from these two networks are then concatenated for each context sentence in order to form a sequence of annotation vectors $(\mathbf{z}_{l-n}, \ldots, \mathbf{z}_{l-1})$.

Unlike the other approaches, in this case, the context vector $\mathbf{p}$ differs for each word w_t in the current sentence, and we denote it by $\mathbf{p}_t$. The context vector $\mathbf{p}_t$ for the t-th word is computed as the weighted sum of the annotation vectors:

$$\mathbf{p}_t = \sum_{l'=l-n}^{l-1} \alpha_{t,l'} \mathbf{z}_{l'},$$

where the attention weight $\alpha_{t,l'}$ is computed by

$$\alpha_{t,l'} = \frac{\exp \text{score}\left(\mathbf{z}_{l'}, \mathbf{h}_t\right)}{\sum_{k=l-n}^{l-1} \exp \text{score}\left(\mathbf{z}_k, \mathbf{h}_t\right)}.$$

$\mathbf{h}_t$ is the hidden state of the recurrent language model of the current sentence from Eq. (5). The scoring function $\text{score}(\mathbf{z}_{l'}, \mathbf{h}_t)$ returns a relevance score of the l'-th context sentence w.r.t. $\mathbf{h}_t$.

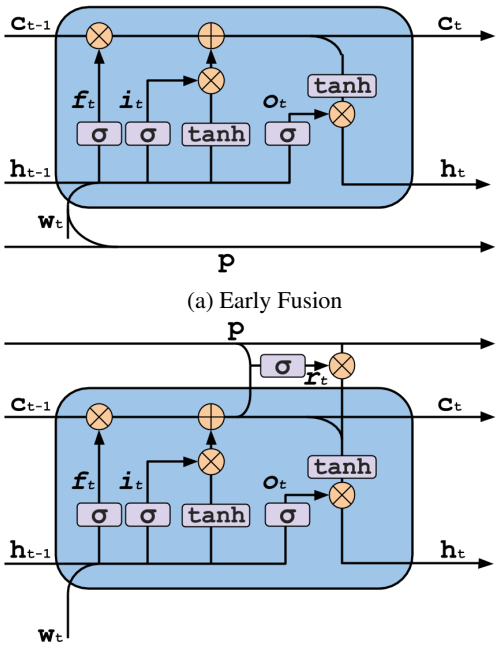
3.2 Conditional LSTM

Early Fusion Once the context vector $\mathbf{p}$ is computed from the n preceding sentences, we need to feed this into the sentence-level recurrent language model. One most straightforward way is to simply consider it as an input at every time step such that

$$\mathbf{x} = \mathbf{E}^\top \mathbf{w}_t + \mathbf{W}_p \mathbf{p},$$

where $\mathbf{E}$ is the word embedding matrix that transforms the one-hot vector of the t-th word into a continuous word vector. We call this approach an *early fusion* of the context.

Late Fusion In addition to this approach, we propose here a modification to the LSTM such that it better incorporates the context from the preceding sentences (summarized by $\mathbf{p}_t$.) The basic idea is to keep dependencies within the sentence being modelled (*intra-sentence dependencies*) and those between the preceding sentences and the current sent (*inter-sentence dependencies*) separately from each other.



(a) Early Fusion

(b) Late Fusion

Figure 1: Proposed fusion methods

We let the memory cell $\mathbf{c}_t$ of the LSTM to model intra-sentence dependencies. This simply means that there is no change to the existing formulation of the LSTM.

The inter-sentence dependencies are reflected on the interaction between the memory cell $\mathbf{c}_t$, which models intra-sentence dependencies, and the context vector $\mathbf{p}$, which summarizes the n preceding sentences. We model this by first computing the amount of influence of the preceding context sentences as

$$\mathbf{r}_t = \sigma\left(\mathbf{W}_r \left(\mathbf{W}_p \mathbf{p}\right) + \mathbf{W}_r \mathbf{c}_t + \mathbf{b}_r\right).$$

This vector $\mathbf{r}_t$ controls the strength of each of the elements in the context vector $\mathbf{p}$. This amount of influence from the n preceding sentences is decided based on the currently captured intra-sentence dependency structures and the preceding sentences.

This controlled context vector $\mathbf{r}_t \odot (\mathbf{W}_p \mathbf{p})$ is used to compute the output of the LSTM layer:

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{c}_t + \mathbf{r}_t \odot \left(\mathbf{W}_p \mathbf{p}\right)\right).$$

This is illustrated in Fig. 1 (b).

We call this approach a *late fusion*, as the effect of the preceding context is fused together with

the intra-sentence dependency structure in the later stage of the recurrent activation.

Late fusion is a simple, but effective way to mitigate the issue of vanishing gradient in corpus-level language modelling. By letting the context representation flow without having to pass through saturating nonlinear activation functions, it provides a linear path through which the gradient for the context flows easily.

4 Related Work

Context-dependent Language Model This possibility of extending a neural or recurrent language modeling to incorporate larger context was explored earlier. Especially, (Mikolov and Zweig, 2012) proposed an approach, called context-dependent recurrent neural network language model, very similar to the proposed approach here. The basic idea of their approach is to use a topic distribution, represented as a vector of probabilities, of previous n *words* when computing the hidden state of the recurrent neural network each time.

There are three major differences in the proposed approach from the work by Mikolov and Zweig (2012). First, the goal in this work is to explicitly model *preceding sentences* to better approximate the corpus-level probability (see Eq. (6)) rather than to get a better context of the current sentence. Second, Mikolov and Zweig (2012) use an external method, such as latent Dirichlet allocation (Blei et al., 2003) or latent semantics analysis (Dumais, 2004) to extract a feature vector, whereas we learn the whole model, including the context vector extraction, end-to-end. Third, we propose a late fusion approach which is well suited for the LSTM units which have recently been widely adopted many works involving language models (see, e.g., Sundermeyer et al., 2015). This late fusion is later shown to be superior to the early fusion approach.

Dialogue Modelling with Recurrent Neural Networks A more similar model to the proposed larger-context recurrent language model is a hierarchical recurrent encoder decoder (HRED) proposed recently by Serban et al. (2015). The HRED consists of three recurrent neural networks to model a dialogue between two people from the perspective of one of them, to which we refer as a speaker. If we consider the last utterance of the speaker, the HRED is a larger-context recurrent language model with early fusion.

Aside the fact that the ultimate goals differ (in their case, dialogue modelling and in our case, document modelling), there are two technical differences. First, they only test with the early fusion approach. We show later in the experiments that the proposed late fusion gives a better language modelling quality than the early fusion. Second, we use a sequence of bag-of-words to represent the preceding sentences, while the HRED a sequence of sequences of words. This allows the HRED to potentially better model the order of the words in each preceding sentence, but it increases computational complexity (one more recurrent neural network) and decreases statistical efficient (more parameters with the same amount of data.)

Skip-Thought Vectors Perhaps the most similar work is the skip-thought vector by Kiros et al. (2015). In their work, a recurrent neural network is trained to read a current sentence, as a sequence of words, and extract a so-called skip-thought vector of the sentence. There are two other recurrent neural networks which respectively model preceding and following sentences. If we only consider the prediction of the following sentence, then this model becomes a larger-context recurrent language model which considers a single preceding sentence as a context.

As with the other previous works we have discussed so far, the major difference is in the ultimate goal of the model. Kiros et al. (2015) fully focused on using their model to extract a good, generic sentence vector, while in this paper we are focused on obtaining a good language model. There are less major technical differences. First, the skip-thought vector model conditions only on the immediate preceding sentence, while we extend this to multiple preceding sentences. Second, similarly to the previous works by Mikolov and Zweig (2012), the skip-thought vector model only implements early fusion.

Neural Machine Translation Neural machine translation is another related approach (Forcada and Ñeco, 1997; Kalchbrenner and Blunsom, 2013; Cho et al., 2014b; Sutskever et al., 2014; Bahdanau et al., 2014). In neural machine translation, often two recurrent neural networks are used. The first recurrent neural network, called an encoder, reads a source sentence, represented as a sequence of words in a source language, to form a context vector, or a set of context vectors. The

other recurrent neural network, called a decoder, then, models the target translation *conditioned on* this source context.

This is similar to the proposed larger-context recurrent language model, if we consider the source sentence as a preceding sentence in a corpus. The major difference is in the ultimate application, machine translation vs. language modelling, and technically, the differences between neural machine translation and the proposed larger-context language model are similar to those between the HRED and the larger-context language model.

Context-Dependent Question-Answering Models Context-dependent question-answering is a task in which a model is asked to answer a question based on the facts from a natural language paragraph. The question and answer are often formulated as filling in a missing word in a query sentence (Hermann et al., 2015; Hill et al., 2015). This task is closely related to the larger-context language model we proposed in this paper in the sense that its goal is to build a model to learn

$$p(q_k|q_{<k}, q_{>k}, D), \tag{8}$$

where q_k is the missing k-th word in a query Q, and $q_{<k}$ and $q_{>k}$ are the context words from the query. D is the paragraph containing facts about this query. It is explicitly constructed so that the query q does not appear in the paragraph D.

It is easy to see the similarity between Eq. (8) and one of the conditional probabilities in the r.h.s. of Eq. (7). By replacing the context sentences S_{l-n}^{l-1} in Eq. (7) with D in Eq. (8) and conditioning w_t on both the preceding and following words, we get a context-dependent question-answering model. In other words, the proposed larger-context language model can be used for context-dependent question-answering, however, with computational overhead. The overhead comes from the fact that for every possible answer the conditional probability completed query sentence must be evaluated.

5 Experimental Settings

5.1 Models

There are six possible combinations of the proposed methods. First, there are two ways of representing the context sentences; (1) bag-of-words (BoW) and (2) a sequence of bag-of-words (SeqBoW), from Sec. 3.1. There are two separate ways to incorporate the SeqBoW; (1) with attention mechanism (ATT) and (2) without it. Then, there are two ways of feeding the context vector into the main recurrent language model (RLM); (1) early fusion (EF) and (2) late fusion (LF), from Sec. 3.2. We will denote them by

1. RLM-BoW-EF-n
2. RLM-SeqBoW-EF-n
3. RLM-SeqBoW-ATT-EF-n
4. RLM-BoW-LF-n
5. RLM-SeqBoW-LF-n
6. RLM-SeqBoW-ATT-LF-n

n denotes the number of preceding sentences to have as a set of context sentences. We test four different values of n; 1, 2, 4 and 8.

As a baseline, we also train a recurrent language model without any context information. We refer to this model by RLM. Furthermore, we also report the result with the conventional, count-based n-gram language model with the modified Kneser-Ney smoothing with KenLM (Heafield et al., 2013).

Each recurrent language model uses 1000 LSTM units and is trained with Adadelta (Zeiler, 2012) to maximize the log-likelihood; $\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{K}\sum_{k=1}^{K} \log p(S_k|S_{k-n}^{k-1})$. We early-stop training based on the validation log-likelihood and report the perplexity on the test set using the best model according to the validation log-likelihood.

We use only those sentences of length up to 50 words when training a recurrent language model for the computational reason. For KenLM, we used all available sentences in a training corpus.

5.2 Datasets

We evaluate the proposed larger-context language model on three different corpora. For detailed statistics, see Table 1.

IMDB Movie Reviews A set of movie reviews is an ideal dataset to evaluate many different settings of the proposed larger-context language models, because each review is highly likely of a single theme (the movie under review.) A set of words or the style of writing will be well determined based on the preceding sentences.

We use the IMDB Movie Review Corpus (IMDB) prepared by Maas et al. (2011).[1] This corpus has 75k training reviews and 25k test reviews.

[1] http://ai.stanford.edu/~amaas/data/sentiment/

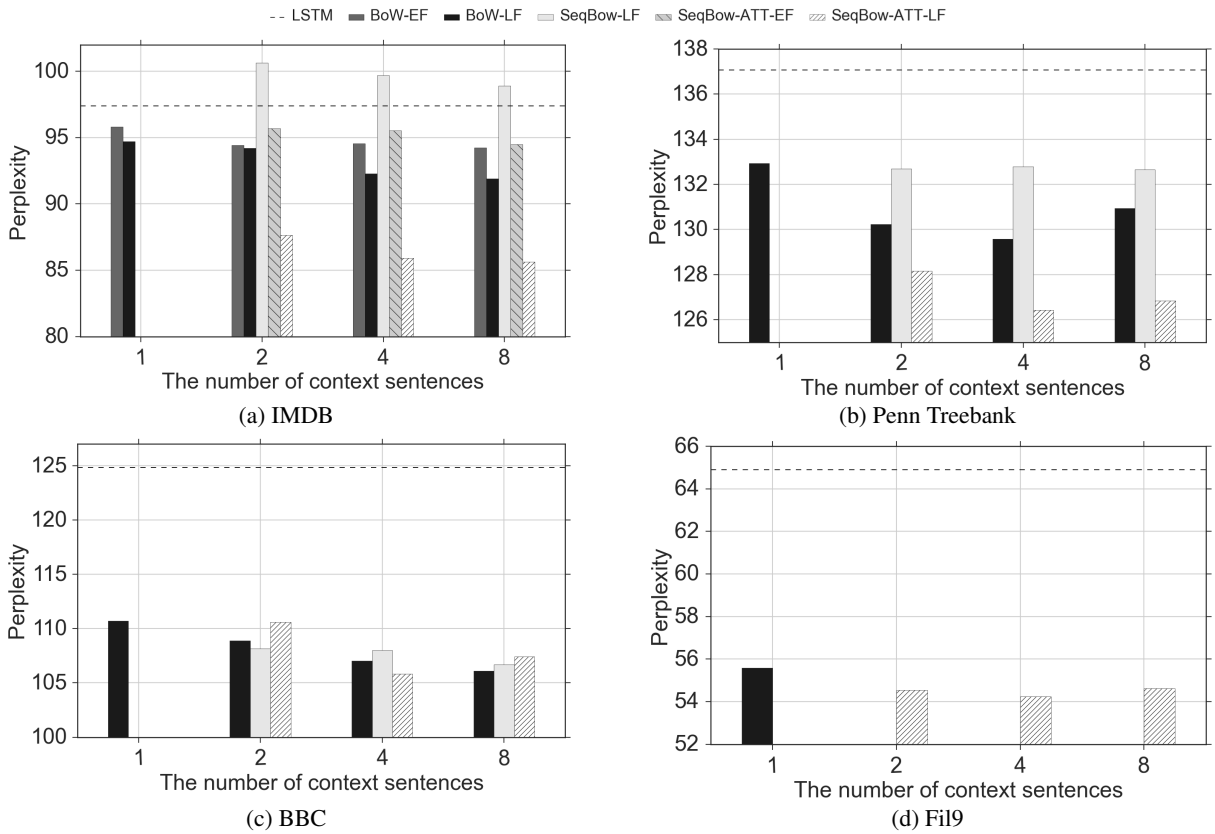


Figure 2: Corpus-level perplexity on (a) IMDB, (b) Penn Treebank, (c) BBC and (d) Fil9. The count-based 5-gram language models with Kneser-Ney smoothing respectively resulted in the perplexities of 110.20, 148, 127.32 and 65.21, and are not shown here.

We use the 30k most frequent words in the training corpus for recurrent language models.

BBC Similarly to movie reviews, each new article tends to convey a single theme. We use the BBC corpus prepared by Greene and Cunningham (2006).[2] Unlike the IMDB corpus, this corpus contains news articles which are almost always written in a formal style. By evaluating the proposed approaches on both the IMDB and BBC corpora, we can tell whether the benefits from larger context exist in both informal and formal languages. We use the 10k most frequent words in the training corpus for recurrent language models.

Both with the IMDB and BBC corpora, we did not do any preprocessing other than tokenization.[3]

Penn Treebank We evaluate a normal recurrent language model, count-based n-gram language model as well as the proposed RLM-BoW-EF-n and RLM-BoW-LF-n with varying $n = 1, 2, 4, 8$ on the Penn Treebank Corpus. We preprocess the corpus according to (Mikolov et al., 2011) and use a vocabulary of 10k words from the training corpus.

Fil9 Fil9 is a cleaned Wikipedia corpus, consisting of approximately 140M tokens, and is provided on Matthew Mahoney's website.[4] We tokenized the corpus and used the 44k most frequent words in the training corpus for recurrent language models.

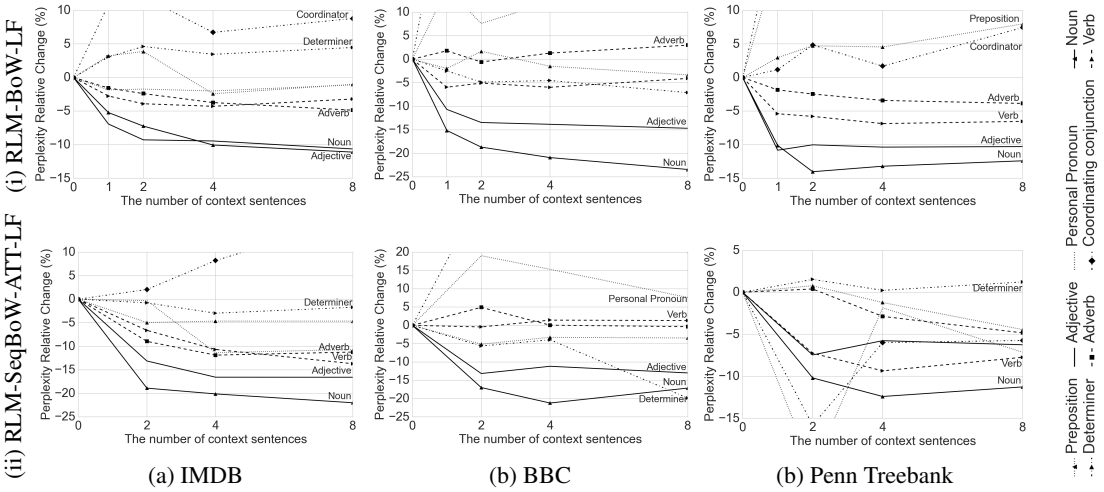
6 Results and Analysis

Corpus-level Perplexity We evaluated the models, including all the proposed approaches (RLM-{BoW,SeqBoW}-{ATT,∅}-{EF,LF}-n), on the IMDB corpus. In Fig. 2 (a), we see three major trends. First, RLM-BoW, either with the early fusion or late fusion, outperforms both the count-based n-gram and recurrent language model (LSTM) regardless of the number of context sentences. Second, the improvement grows as the number n of context sentences increases, and this is most visible with the novel *late fusion*. Lastly,

[2] http://mlg.ucd.ie/datasets/bbc.html

[3] https://github.com/moses-smt/mosesdecoder/blob/master/scripts/tokenizer/tokenizer.perl

[4] http://mattmahoney.net/dc/textdata



(a) IMDB (b) BBC (b) Penn Treebank

Figure 3: Perplexity per POS tag on the (a) IMDB, (b) BBC and (c) Penn Treebank corpora.

we see that the RLM-SeqBoW does not work well regardless of the fusion type (RLM-SeqBow-EF not shown), while the attention-based model (RLM-SeqBow-ATT) outperforms all the others.

After observing that the late fusion clearly outperforms the early fusion, we evaluated only RLM-{BoW,SeqBoW}-{ATT}-LF-n's on the other two corpora.

On the other two corpora, PTB and BBC, we observed a similar trend of RLM-SeqBoW-ATT-LF-n and RLM-BoW-LF-n outperforming the two conventional language models, and that this trend strengthened as the number n of the context sentences grew. We also observed again that the RLM-SeqBoW-ATT-LF outperforms RLM-SeqBoW-LF and RLM-BoW in almost all the cases.

Observing the benefit of RLM-SeqBoW-ATT-LF, we evaluated only such model on Fil9 to validate its performance on large corpus. Similar to the results on all three previous corpora, we continue to observe the advantage of RLM-SeqBoW-ATT-LF-n on Fil9 corpus.

From these experiments, the benefit of allowing larger context to a recurrent language model is clear, however, with the right choice of the context representation (see Sec. 3.1) and the right mechanism for feeding the context information to the recurrent language model (see Sec. 3.2.) In these experiments, the sequence of bag-of-words representation with attention mechanism, together with the late fusion was found to be the best choice in all four corpora.

One possible explanation on the failure of the SeqBoW representation with a context recurrent neural network is that it is simply difficult for the context recurrent neural network to compress multiple sentences into a single vector. This difficulty in training a recurrent neural network to compress a long sequence into a single vector has been observed earlier, for instance, in neural machine translation (Cho et al., 2014a). Attention mechanism, which was found to avoid this problem in machine translation (Bahdanau et al., 2014), is found to solve this problem in our task as well.

Perplexity per Part-of-Speech Tag Next, we attempted at discovering why the larger-context recurrent language model outperforms the unconditional one. In order to do so, we computed the *perplexity per part-of-speech (POS) tag*.

We used the Stanford log-linear part-of-speech tagger (Stanford POS Tagger, Toutanova et al., 2003) to tag each word of each sentence in the corpora.[5] We then computed the perplexity of each word and averaged them for each tag type separately. Among the 36 POS tags used by the Stanford POS Tagger, we looked at the perplexities of the ten most frequent tags (NN, IN, DT, JJ, RB, NNS, VBZ, VB, PRP, CC), of which we combined NN and NNS into a new tag Noun and VB and VBZ into a new tag Verb.

We show the results using the RLM-BoW-LF and RLM-SeqBoW-ATT-LF on three corpora– IMDB, BBC and Penn Treebank– in Fig. 3. We observe that the predictability, measured by the perplexity (negatively correlated), grows most for nouns (Noun) and adjectives (JJ) as the number of context sentences increases. They are followed by verbs (Verb). In other words, nouns, adjec-

[5] http://nlp.stanford.edu/software/tagger.shtml

	IMDB		BBC		Penn TreeBank		Fil9	
	# Sentences	# Words	# Sentences	# Words	# Sentences	# Words	# Sentences	# Words
Training	930,139	21M	37,207	890K	42,068	888K	6,619,098	115M
Validation	152,987	3M	1,998	49K	3,370	70K	825,919	14M
Test	151,987	3M	2,199	53K	3,761	79K	827,416	14M

Table 1: Statistics of IMDB, BBC, Penn TreeBank and Fil9.

tives and verbs are the ones which become more predictable by a language model given more context. We however noticed the relative degradation of quality in coordinating conjunctions (CC), determiners (DT) and personal pronouns (PRP).

It is worthwhile to note that nouns, adjectives and verbs are open-class, content, words, and conjunctions, determiners and pronouns are closed-class, function, words (see, e.g., Miller, 1999). The functions words often play grammatical roles, while the content words convey the content of a sentence or discourse, as the name indicates. From this, we may carefully conclude that the larger-context language model improves upon the conventional, unconditional language model by capturing the theme of a document, which is reflected by the improved perplexity on "content-heavy" open-class words (Chung and Pennebaker, 2007). In our experiments, this came however at the expense of slight degradation in the perplexity of function words, as the model's capacity stayed same (though, it is not necessary.)

This observation is in line with a recent finding by Hill et al. (2015). They also observed significant gain in predicting open-class, or content, words when a question-answering model, including humans, was allowed larger context.

7 Conclusion

In this paper, we proposed a method to improve language model on corpus-level by incorporating larger context. Using this model results in the improvement in perplexity on the IMDB, BBC, Penn Treebank and Fil9 corpora, validating the advantage of providing larger context to a recurrent language model.

From our experiments, we found that the sequence of bag-of-words with attention is better than bag-of-words for representing the context sentences (see Sec. 3.1), and the late fusion is better than the early fusion for feeding the context vector into the main recurrent language model (see Sec. 3.2). Our part-of-speech analysis revealed that content words, including nouns, adjectives and verbs, benefit most from an increasing number of context sentences. This analysis suggests that larger-context language model improves perplexity because it captures the theme of a document better and more easily.

To explore the potential of such a model, there are several aspects in which more research needs to be done. First, the four datasets we used in this paper are relatively small in the context of language modelling, therefore the proposed larger-context language model should be evaluated on larger corpora. Second, more analysis, beyond the one based on part-of-speech tags, should be conducted in order to better understand the advantage of such larger-context models. Lastly, it is important to evaluate the impact of the proposed larger-context models in downstream tasks such as machine translation and speech recognition.

Acknowledgments

This work is done as a part of the course DS-GA 1010-001 Independent Study in Data Science at the Center for Data Science, New York University. KC thanks Facebook, Google (Google Faculty Award 2016) and NVidia (GPU Center of Excellence 2015–2016).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

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

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on* 5(2):157–166.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3:993–1022.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder-decoder approaches. In *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8)*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Cindy Chung and James W Pennebaker. 2007. The psychological functions of function words. *Social communication* pages 343–359.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. pages 2067–2075.

Susan T Dumais. 2004. Latent semantic analysis. *Annual review of information science and technology* 38(1):188–230.

Mikel L Forcada and Ramón P Ñeco. 1997. Recursive hetero-associative memories for translation. In *Biological and Artificial Computation: From Neuroscience to Technology*, Springer, pages 453–462.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*. ACM Press, pages 377–384.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. Lstm: A search space odyssey. *arXiv preprint arXiv:1503.04069* .

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*. Sofia, Bulgaria, pages 690–696.

Karl Moritz Hermann, Tomáš Kočiskỳ, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *arXiv preprint arXiv:1506.03340* .

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

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

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. *arXiv preprint arXiv:1511.03962* .

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. pages 2342–2350.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *EMNLP*. pages 1700–1709.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. *arXiv preprint arXiv:1506.06726* .

Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, pages 142–150.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*. pages 1045–1048.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, pages 5528–5531.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *SLT*. pages 234–239.

George A Miller. 1999. On knowing a word. *Annual Review of psychology* 50(1):1–19.

Ronald Rosenfeld. 2000. Two decades of statistical language modeling: where do we go from here. In *Proceedings of the IEEE*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5:3.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808* .

Martin Sundermeyer, Hermann Ney, and Ralf Schluter. 2015. From feedforward to recurrent lstm neural networks for language modeling. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 23(3):517–529.

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

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, pages 173–180.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .

The Creation and Analysis of a Website Privacy Policy Corpus

Shomir Wilson[1], Florian Schaub[1], Aswarth Abhilash Dara[1], Frederick Liu[1], Sushain Cherivirala[1], Pedro Giovanni Leon[2], Mads Schaarup Andersen[1], Sebastian Zimmeck[3], Kanthashree Mysore Sathyendra[1], N. Cameron Russell[4], Thomas B. Norton[4], Eduard Hovy[1], Joel Reidenberg[4] and Norman Sadeh[1]

[1]Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, USA
[2]Stanford University, Center for Internet and Society, Stanford, CA, USA
[3]Columbia University, Department of Computer Science, New York, NY, USA
[4]Fordham University, Law School, New York, NY, USA
shomir@cs.cmu.edu, sadeh@cs.cmu.edu

Abstract

Website privacy policies are often ignored by Internet users, because these documents tend to be long and difficult to understand. However, the significance of privacy policies greatly exceeds the attention paid to them: these documents are binding legal agreements between website operators and their users, and their opaqueness is a challenge not only to Internet users but also to policy regulators. One proposed alternative to the status quo is to automate or semi-automate the extraction of salient details from privacy policy text, using a combination of crowdsourcing, natural language processing, and machine learning. However, there has been a relative dearth of datasets appropriate for identifying data practices in privacy policies. To remedy this problem, we introduce a corpus of 115 privacy policies (267K words) with manual annotations for 23K fine-grained data practices. We describe the process of using skilled annotators and a purpose-built annotation tool to produce the data. We provide findings based on a census of the annotations and show results toward automating the annotation procedure. Finally, we describe challenges and opportunities for the research community to use this corpus to advance research in both privacy and language technologies.

1 Introduction

Privacy policies written in natural language are a nearly pervasive feature of websites and mobile applications. The "notice and choice" legal regimes of many countries require that website operators post a notice of how they gather and process users' information. In theory, users then choose whether to accept those practices or to abstain from using the website or service. In practice, however, the average Internet user struggles to understand the contents of privacy policies (McDonald and Cranor, 2008) and generally does not read them (Federal Trade Commission, 2012; President's Concil of Advisors on Science and Technology, 2014). This disconnect between Internet users and the data practices that affect them has led to the assessment that the notice and choice model is ineffective in the status quo (Reidenberg et al., 2015b; Cate, 2010).

Thus, an opening exists for language technologies to help "bridge the gap" between privacy policies in their current form and representations that serve the needs of Internet users. Such a bridge would also serve unmet needs of policy regulators, who do not have the means to assess privacy policies in large numbers. Legal text is a familiar domain for natural language processing, and the legal community has demonstrated some reciprocal interest (Mahler, 2015). However, the scale of the problem and its significance—i.e., to virtually any Internet user, as well as to website operators and policy regulators—distinguishes it and provides immense motivation (Sadeh et al., 2013).

To this end, we introduce a corpus of 115 website privacy policies annotated with detailed information about the data practices that they describe.[1] This information consists of 23K data practices, 128K practice attributes, and 103K annotated text spans, all produced by skilled anno-

[1]The dataset is available for download at www.usableprivacy.org/data.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1330–1340,
Berlin, Germany, August 7-12, 2016.

tators. To the best of our knowledge, this is the first large-scale effort to annotate privacy policies at such a fine level of detail. It exceeds prior efforts to annotate sentence-level fragments of policy text (Breaux and Schaub, 2014), answer simple overarching questions about privacy policy contents (Wilson et al., 2016; Zimmeck and Bellovin, 2014), or analyze the readability of privacy policies (Massey et al., 2013). We further present analysis that demonstrates the richness of the corpus and the feasibility of partly automating the annotation of privacy policies.

The remainder of this paper is structured as follows. We discuss related work and contextualize the corpus we have created in Section 2. In Section 3 we describe the creation of the corpus, including the collection of a diverse set of policies and the creation of a privacy policy annotation tool. Section 4 presents analysis that illustrates the diversity and complexity of the corpus, and Section 5 shows results on the prediction of policy structure. Finally, in Section 6 we describe some promising avenues for future work.

2 Related Work

Prior attempts on analyzing privacy policies focused largely on manually assessing their usability (Jensen and Potts, 2004) or compliance with self-regulatory requirements (Hoke et al., 2015). Breaux et al. proposed a description logic to analyze and reason about data sharing properties in privacy policies (2013), but rely on a small set of manually annotated privacy policies to instantiate their language. Automated assessments have largely focused on readability scores (Massey et al., 2013; Meiselwitz, 2013; Ermakova et al., 2015). Cranor et al. leveraged the standardized format of privacy notices in the U.S. financial industry to automatically analyze privacy polices of financial institutions (2013). However, in spite of notable efforts such as P3P (Wenning et al., 2006), the majority of privacy policies are unstructured and do not follow standardized formats.

Costante et al. (2012) proposed a supervised learning approach to determine which data practice categories are covered in a privacy policy. Rule-based extraction techniques have been proposed to extract some of a website's data collection practices from its privacy policy (Costante et al., 2013) or to answer certain binary questions about a privacy policy (Zimmeck and Bellovin, 2014). Other approaches leverage topic modeling (Chundi and Subramaniam, 2014; Stamey and Rossi, 2009) or sequence alignment techniques (Liu et al., 2014; Ramanath et al., 2014) to analyze privacy policies or identify similar policy sections and paragraphs. However, the complexity and vagueness of privacy policies makes it difficult to automatically extract complex data practices from privacy policies without substantial gold standard data.

Crowdsourcing has been proposed as a potential approach to obtain annotations for privacy policies (Sadeh et al., 2013; Breaux and Schaub, 2014; Wilson et al., 2016). However, crowdworkers are not trained in understanding and interpreting legal documents, which may result in interpretation discrepancies compared to experts (Reidenberg et al., 2015a). Our policy annotation tool shares some common features with GATE (Bontcheva et al., 2013), although the interface for our tool is simpler to fit the specific requirements of the task.

Few prior efforts, aside from those we cite above, have applied natural language processing to privacy policies or other legal documents purported for the general public to regularly read. More generally, legal text has a history of attention from natural language processing (Bach et al., 2013; Galgani et al., 2012; Francesconi et al., 2010) and from artificial intelligence (Sartor and Rotolo, 2013; Bench-Capon et al., 2012). Classifying legal text into categories has received some interest (Šavelka and Ashley, 2015; Mickevicius et al., 2015), as well as making the contents of legal texts more accessible (Boella et al., 2015; Curtotti and McCreath, 2013).

Compared to prior efforts, our data set is notable for its combination of size, input from experts (for the label scheme) and skilled annotators (for the annotation procedure), and fine-grained detail.

3 Corpus Creation and Structure

In this section we describe our procedure for selecting a diverse set of privacy policies, our annotation scheme, how we obtained annotations, and the structure of the corpus.

3.1 Privacy Policy Selection

Privacy policies vary in length, complexity, legal sophistication, and coverage of services. For instance, privacy policies of large companies may cover multiple services, websites, apps, and even physical stores; such policies are often crafted by legal teams and frequently updated. Privacy poli-

cies of smaller or less popular companies may have narrower focus or vary in employed language, and they may be updated less frequently.

To reflect this diversity, we used a two-step process for policy selection: (1) relevance-based website pre-selection and (2) sector-based sub-sampling. First, we monitored Google Trends (Google, 2015) for one month (May 2015) to collect the top five search queries for each trend. Then, for each query we retrieved the first five websites listed on each of the first 10 pages of results. This process produced a diverse sample of 1,799 unique websites.

Second, we sub-sampled from this website dataset according to DMOZ.org's top-level website sectors.[2] More specifically, we organized the dataset into 15 sectors (e.g., Arts, Shopping, Business, News). We excluded the "World" sector and limited the "Regional" sector to the "U.S." subsector in order to ensure that all privacy policies in our corpus are subject to the same legal and regulatory requirements. We ranked the websites in each sector according to their frequency in the retrieved search results. Then we selected eight websites from each sector by randomly chosing two websites from each rank quartile.

For each selected website, we manually verified that it had an English-language privacy policy and that it pertained to a US company (based on contact information and WHOIS entry) before downloading its privacy policy. Excluded websites were replaced with random re-draws from the same sector rank quartile. Some privacy policies covered more than one of the selected websites (e.g., the Disney privacy policy covered disney.go.com and espn.go.com), resulting in a final dataset of 115 privacy policies across 15 sectors.

3.2 Annotation Scheme and Process

We developed a policy annotation scheme to capture the data practices specified by privacy policies. To ensure the scheme reflected actual policy contents, development occurred as an iterative refinement process, in which a small group of domain experts (privacy experts, public policy experts, and legal scholars) identified different data practice categories and their descriptive attributes from multiple privacy policies. The annotation scheme was then applied to additional policies and refined over multiple iterations during discussions among the experts.

The final annotation scheme consists of ten data practice categories:

1. *First Party Collection/Use*: how and why a service provider collects user information.
2. *Third Party Sharing/Collection*: how user information may be shared with or collected by third parties.
3. *User Choice/Control*: choices and control options available to users.
4. *User Access, Edit, & Deletion*: if and how users may access, edit, or delete their information.
5. *Data Retention*: how long user information is stored.
6. *Data Security*: how user information is protected.
7. *Policy Change*: if and how users will be informed about changes to the privacy policy.
8. *Do Not Track*: if and how Do Not Track signals[3] for online tracking and advertising are honored.
9. *International & Specific Audiences*: practices that pertain only to a specific group of users (e.g., children, Europeans, or California residents).
10. *Other*: additional sub-labels for introductory or general text, contact information, and practices not covered by the other categories.

An individual *data practice* belongs to one of the ten categories above, and it is articulated by a category-specific set of *attributes*. For example, a User Choice/Control data practice is associated with four mandatory attributes (*Choice Type*, *Choice Scope*, *Personal Information Type*, *Purpose*) and one optional attribute (*User Type*). The annotation scheme defines a set of potential values for each attribute. To ground the data practice in the policy text, each attribute also may be associated with a *text span* in the privacy policy.

The set of mandatory and optional attributes reflects the potential level of specificity with which a data practice of a given category may be described. Optional attributes are less common, while mandatory attributes are necessary to represent a data practice. However, privacy policies are often vague or ambiguous on many of these attributes. Therefore, a valid value for each attribute is *Unspecified*, allowing annotators to express an absence of information.

[2]The DMOZ.org website sectors are notable for their use by Alexa.com.

[3]www.w3.org/2011/tracking-protection

Documents	115
Words	266,713
Annotated Data Practices	23,194
Annotated Attributes	128,347
Annotated Text Spans	102,576
Annotators Per Document	3
Annotators Total	10

Table 1: Totalized statistics on the corpus.

We developed a web-based annotation tool, shown in Figure 1, for skilled annotators to apply our annotation scheme to the selected privacy policies.[4] In preparation, privacy policies were divided into paragraph-length *segments* for annotators to read in the tool, one at a time in sequence. For each segment, an annotator may label zero or more data practices from each category. To create a data practice, an annotator first selects a practice category and then specifies values and text spans for each of its attributes. Annotators can see a list of data practices they have created for a segment and selectively duplicate and edit them to annotate practices that differ only slightly, though we omit these features from the figure for brevity.

4 Composition of the OPP-115 Corpus

The annotation process produced a nuanced and diverse dataset, which we describe in detail below. We name the dataset the *OPP-115 Corpus* (Online Privacy Policies, set of 115) for convenience.

4.1 Policy Contents

Table 1 shows some descriptive statistics for the corpus as a whole. Each privacy policy was read by three skilled annotators, who worked independently, and a total of ten annotators participated in the process. All annotators were law students and were compensated for their work at rates appropriate for student employees at their respective universities. They required a mean of 72 minutes per policy, though this number is slightly inflated by outliers when they stepped away from in-progress sessions for extended periods of time. The annotators produced a total of 23K data practices, although this number contains some redundancies between annotators' efforts.[5] In aggregate, these data practices are associated with 128K values for attributes and 103K selected spans of policy text. Note that the annotation tool required the selection of a text span for mandatory attributes, but did not require a text-based justification for optional attributes or attributes marked as "Unspecified".

The corpus allows us to investigate the composition of typical privacy policies in terms of data practices. Privacy policies are known for their length and complexity, but those notions do not necessarily entail a density of pertinent information. Table 2 shows the pre-consolidation quantities of practices that we collected in each of the ten annotation categories, along with the mean and median counts of practices per privacy policy. Intuitively, First Party Collection/Use and Third Party Sharing/Collection dominated the rankings by frequency: the collection, usage, and sharing of user data are the primary concerns that compel the production of privacy policies. Data practices in the Other category, while frequent, were mostly statements that were ostensibly not about user data; 57% were introductory, contact, or generic information. Means were above medians for all categories, reflecting rightward skews for all the distributions.

Table 2 also contains statistics on segment-level category coverage and annotator agreement. Here, *coverage* is meant in an *ipso facto* sense: a practice category covers a policy segment if two of three annotators each identified at least one practice from that category in the segment text. Differences in the category rankings by frequency and by coverage reveal that practices in some categories are less tightly clustered than others. In particular, Data Retention is the second rarest practice category but ranks fourth by segment coverage. Since Kappa is applied here to an artificial task (annotators were not asked to label entire segments) the common conventions for its interpretation (Carletta, 1996; Viera and Garrett, 2005) are not directly applicable. However, Do Not Track and International and Specific Audiences remain standout categories with the greatest segment-level agreement. We hypothesize that these two categories have the most easily recognizable cues for annotation. Do Not Track practices, for example, are associated with the eponymous phrase.

Finally, the pre-consolidation mean and median

[4] Our experts for the annotation scheme development and our skilled annotators were mutually exclusive groups.

[5] We describe a method to *consolidate* annotations (i.e., to eliminate redundancies between annotators' data) in Section 4.2. Here, we analyze policy contents pre-consolidation to avoid propagating the effects of nontrivial assumptions necessary for consolidation.

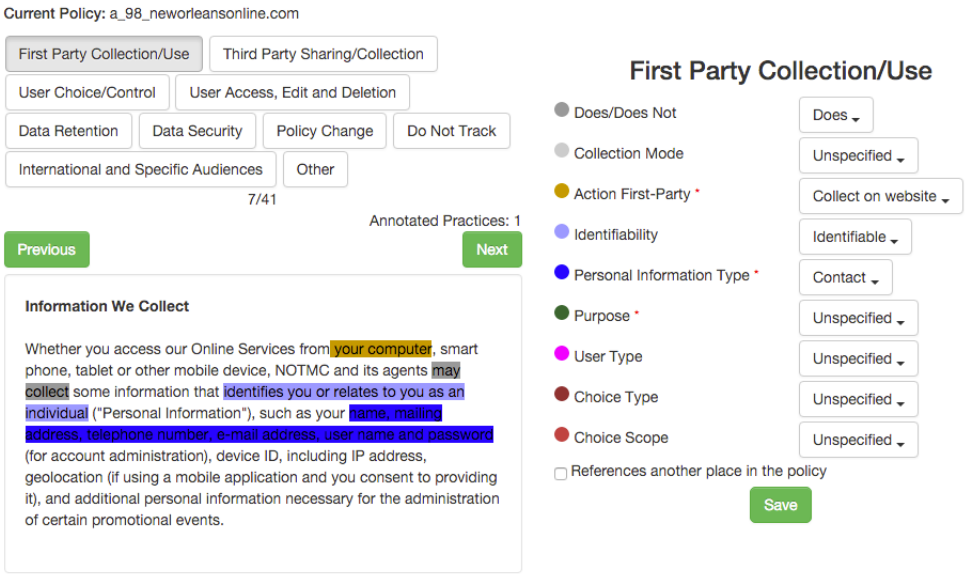


Figure 1: Web-based tool for our skilled annotators of privacy policies.

Category	Freq.	Mean	Median	Coverage	Fleiss' Kappa
First Party Collection/Use	8,956	78	74	.27	.76
Third Party Sharing/Collection	5,230	45	39	.21	.76
Other	3,551	31	25	.24	.49
User Choice/Control	1,791	16	13	.08	.61
Data Security	1,009	9	7	.05	.67
International and Specific Audiences	941	8	6	.07	.87
User Access, Edit and Deletion	747	6	5	.03	.74
Policy Change	550	5	4	.03	.73
Data Retention	370	3	2	.20	.55
Do Not Track	90	1	0	.01	.91

Table 2: By-category descriptive statistics for the data practices in the corpus. These statistics are calculated prior to consolidating multiple annotators' work. Means and medians are calculated across the population of policies in the corpus. Coverage and Kappa are calculated in terms of by-segment contents.

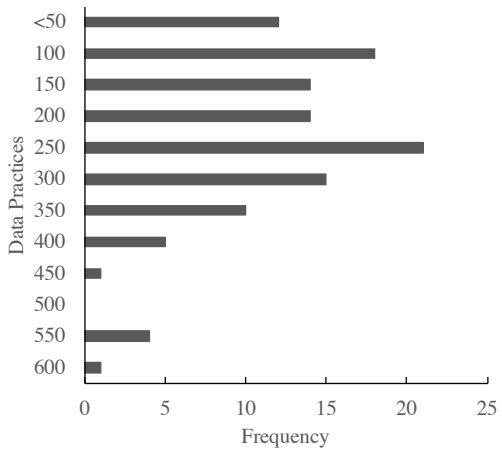


Figure 2: Distribution of data practices per policy.

quantities of data practices per policy were 202 and 200, respectively. These do not correspond to columnar totals that may be calculated from Table 2 because the categories were not equally distributed among the privacy policies. Figure 2 shows the distribution of quantities of data practices per policy. The distribution exhibits a skew toward larger numbers of data practices per policy. Importantly, differences in the number of data practices should not be interpreted as varying levels of data protection or privacy. A privacy policy that contains many data practices may exhibit substantial redundancy among them, and a privacy policy with relatively few data practices could merely be concise. In either case, the data practices may be responsive to users concerns or at odds with them.

4.2 Consolidating Annotators' Work

In this section we discuss the problem of *consolidation*, or merging data practices from multiple annotators if those practices refer to the same underlying practice expressed by the text. The ambiguity and vagueness of privacy policies (Reidenberg et al., 2016) and the sophistication of the annotation scheme are natural limitations on annotator agreement. With that in mind, we present a consolidation procedure to collapse redundant annotations with the proviso that practices labeled by only one or two skilled annotators also have substantial value and merit retention.

First, we institute some basic requirements about locality and topicality to determine which data practices are eligible for consolidation. Given a segment, if annotators $A_1, ..., A_n$ (for $n = 2$ or $n = 3$ in our dataset) respectively produce sets of data practices $P_1, ..., P_n$, then a selection of data practices $p_1 \in P_1, ..., p_n \in P_n$ is eligible to be consolidated into a single data practice only if all of them belong to the same category. Additionally, three implicit assumptions in this requirement are that (1) at least two annotators contribute practices to a consolidation set, (2) all the practices are located in the same policy segment, and (3) each practice must belong to a unique annotator.

For each segment we create an exhaustive list of eligible combinations of data practices to consolidate, score and rank each combination using a method detailed below, prune the list with a score threshold, and finally perform consolidations in order of ranking until no further consolidations are possible. Consolidation sets containing three annotators' practices are considered prior to sets containing practices from only two annotators. The data practices in a chosen consolidation set are removed and replaced by a single "master" data practice. To do this, it is necessary to merge sets of values and sets of text spans respectively associated with each attribute. Sets of values are merged using a majority vote if possible and set to Unspecified if otherwise; the latter case occurs in approximately a third of all mergers. Sets of text spans are merged with a strong bias toward recall, by creating a new text span that begins and ends with respectively the first and last indexes in the set.

Our scoring method is based on the summative overlap between the sets of text spans associated with attributes of data practices, with normalization to account for longer spans. Since the text spans connect data practices to the policy text, we use their overlaps as evidence that two annotators' data practices refer to the same underlying practice in the text. Thus, a score for two data practices that are associated with roughly the same policy text is relatively high, and a score for two data practices that are associated with different text is low.

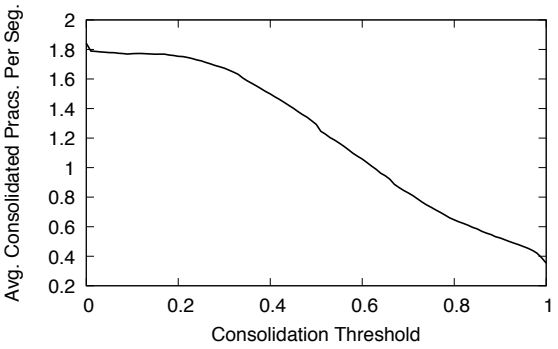


Figure 3: Consolidation threshold value versus the average number of data practices per segment produced by consolidation.

Figure 3 shows the effect of the consolidation threshold on the average number of practices produced by consolidation per policy segment (i.e., excluding those original data practices that were retained because they were not subject to consolidation). Past a threshold value of approximately 0.2, the number of practices steadily decreases. Notably the average number of practices produced by consolidation is substantially less than the average practices per annotator per segment (2.04) at any point on the curve, indicating a relative lack of agreement between annotators in terms of text span selections. As part of the corpus, we release consolidated datasets at threshold values of 0.5, 0.75, and 1.

4.3 Data Exploration Website

The data practice annotations are difficult for human readers to interpret without visual connections to the policy text. To help researchers, policy regulators, and the general public understand the structure and utility of the data set, we created a data exploration website[6] that visually integrates the data practice annotations with the texts of privacy policies. Site interactivity allows users to search for websites in the dataset or browse by DMOZ sectors.

The website also allows users to compare privacy policies by categorical structure, data prac-

[6] explore.usableprivacy.org

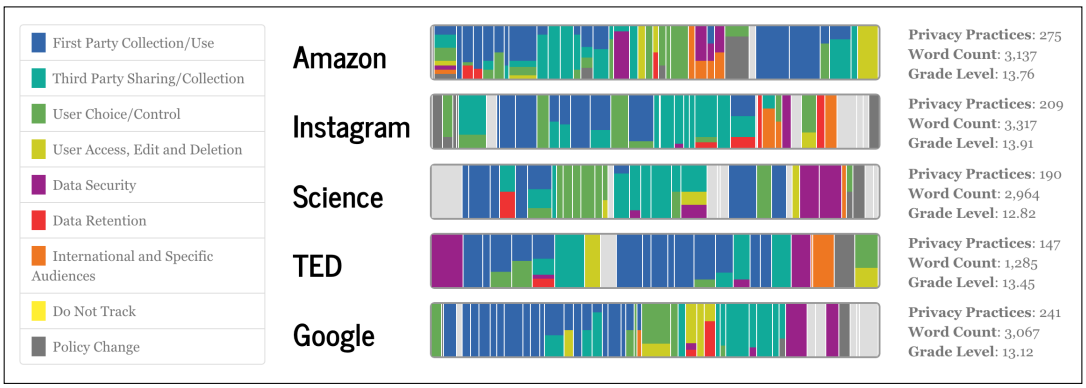


Figure 4: Comparing five policies on the data exploration website.

tice quantities, and reading level.[7] Figure 4 shows a sample comparison between five websites. Each policy's segments are depicted in order from left to right. Segments are colored according to the practice categories that annotators labeled within them. Qualitative patterns are discernible; for example, several of these policies have large blocks of First Party Collection/Use toward the beginning or large blocks of Third Party Sharing/Collection further inward. We discuss exploiting such recurring structures in the next section.

5 Prediction of Policy Structure

The current human annotation procedure is impractical for covering the entire Internet or accounting for changes in privacy policies. This raises the question of whether the process can be partly automated. In this section we describe our experiments to automatically assign category labels to policy segments, which would enable the simplification of the annotation task.

5.1 Experiments

Our dataset consisted of 3,792 segments from 115 privacy policies. We represented the text of each segment as a dense vector using Paragraph2Vec (Le and Mikolov, 2014) and the GENSIM toolkit (Řehůřek and Sojka, 2010). This approach exploited semantic similarities between words in the vocabulary of privacy policies, acknowledging that the vocabulary in this domain is specialized but not completely standardized. We assigned each policy segment a binary vector of category-specific labels, with each element in the vector corresponding to the presence or absence of a data practice category in the segment. We considered a vector with twelve elements, with nine of them coming from existing practice categories (all except Other). The remaining three came from elevating three attributes of Other to category status: *Introductory/Generic*, *Practice Not Covered*, and *Privacy Contact Information*. We created gold standard data for this problem using a simplified consolidation approach: if two or more annotators agreed that a category is present in a segment, then we labeled that segment with the category.

To predict the category labels of privacy policy segments, we tried three approaches. Two were logistic regression and SVM models, for which we treated this as a multi-class classification problem. Since 2^{12} unique category vectors exist, we trimmed the label space to only those that occur in the training set. The third was a sequence labeling approach inspired by prior work to apply hidden Markov models (HMMs) to privacy policy text (Ramanath et al., 2014). Our work differs from this prior work by using labels from an annotation scheme constructed by privacy experts rather than topics developed from an unsupervised method. Additionally, in our formulation, each hidden state corresponds to one of the unique binary vectors that represent classes of category combinations in the training data. The HMM's transition probabilities capture the tendency of privacy policy authors to organize topics (i.e., practice categories in our annotation scheme) in similar sequences. Since each segment is represented by a unique real-valued vector from Paragraph2Vec, it was not possible to directly obtain an emission probability distribution from the training data. Therefore, we ran the K-Means++ algorithm using the scikit-learn toolkit (Pedregosa et al., 2011) on the segment vector representations and assigned each segment to a cluster. The emission probability distribution then captured the tendencies of a given class and generated the segment

[7] `explore.usableprivacy.org/compare`

that is represented as a cluster. These two distributions are estimated empirically from the training data, and we used Viterbi decoding to obtain the best labeling sequence during the prediction.

5.2 Results

We split the set of 115 policies into subsets of 75 for training and 40 for testing. The number of clusters in the HMM approach[8] is set to 100 and the results are shown in Table 3 as means across 10 runs. The standard deviations for these performance figures are generally between 0.01 and 0.05; the one exception is Do Not Track (the least frequent category) with a standard deviation of 0.2. As the table shows, although the HMM does not reach the same performance as SVM, it performs similarly to logistic regression and meets or exceeds its F1-score for five categories.

We interpret the strength of the SVM as indicating the strong potential to partly automate the policy labeling procedure, especially for two categories: First Party Collection/Use (a standout performance and the category for which the most labeled data exists) and Do Not Track (a perfect performance, likely due to the limited vocabulary used to describe practices in this category). Additionally, while the HMM did not perform as well overall, we note that its micro-average F1 was a slight improvement over logistic regression. With relatively little data to train this HMM, we expect that the accumulation of more labeled instances can improve its performance substantially.

6 Future Directions

The OPP-115 Corpus enables research in several directions of interest to natural language processing and usable privacy. We sketch some opportunites for future work below.

A central challenge for this research is the scalability of policy annotation. Although it was necessary to annotate the first 115 policies manually, to ensure the annotations were responsive to the annotation scheme, a less labor-intensive approach will be required for large-scale Web coverage. The OPP-115 Corpus is a valuable dataset for this move toward automated methods. Additionally, a strong potential exists for a combination of automated annotation of coarse information and human annotation of finer details. For example, automated category labeling of policy segments is feasible, as demonstrated in Section 5. Asking a human to label practices in a single category would be a reduction in effort, especially if they are shown text that is relevant to the category. Crowdsourcing also becomes a possibility when the complexity of the task is reduced.

An ambitious goal will be to eliminate human annotators altogether. Our preliminary analysis has shown that the policy vocabularies associated with certain annotations are very distinctive (e.g., the Do Not Track category or financial information as a data type, for example), lending themselves to automatic identification. By producing confidence ratings alongside data practice predictions, an automated system could mitigate its shortcomings.

Separately, data practices must be presented to Internet users in a way that is responsive to their concerns. Text summarization is a possibility, using the annotations as a guide for important details to retain. Internet users have already demonstrated limited patience with text-based privacy policies, which adds a nuance to this challenge and suggests the need for a combination of text and pictorial representations (or chiefly pictorial representations) to communicate data practices (Schaub et al., 2015).

Additional questions of interest include:

- *How can the data practice annotations for a policy be combined into a cohesive interpretation?* The relationships between data practices are not straightforward. Vagueness, contradictions, and unclear scope are all problems for constructing a knowledge base of them.
- *How can the balance between human and automated methods for annotation be optimized?* The model for the ideal combination is subject to parameters such as the availability of resources and the necessary level of confidence for annotations.
- *How can sectoral norms and outliers be identified automatically?* A bank website that collects users' health information, for example, deserves scrutiny. It seems appropriate to address this question with clustering techniques, using features from the data practices and from the policy text.

[8] We tuned the parameters of the HMM approach and SVM after performing a five-fold cross validation on the training data.

Category	LR			SVM			HMM		
	P	R	F	P	R	F	P	R	F
First Party Collection/Use	0.73	0.67	0.70	0.76	0.73	0.75	0.69	0.76	0.72
Third Party Sharing/Collection	0.64	0.63	0.63	0.67	0.73	0.70	0.63	0.61	0.62
User Choice/Control	0.45	0.62	0.52	0.65	0.58	0.61	0.47	0.33	0.39
Introductory/Generic*	0.51	0.50	0.50	0.58	0.49	0.53	0.54	0.49	0.51
Data Security	0.48	0.75	0.59	0.66	0.67	0.67	0.67	0.53	0.59
Internat'l and Specific Audiences	0.49	0.69	0.57	0.70	0.70	0.70	0.67	0.66	0.66
Privacy Contact Information*	0.34	0.72	0.46	0.60	0.68	0.64	0.48	0.59	0.53
User Access, Edit, and Deletion	0.47	0.71	0.57	0.67	0.56	0.61	0.48	0.42	0.45
Practice Not Covered*	0.20	0.47	0.28	0.19	0.26	0.22	0.15	0.12	0.13
Policy Change	0.59	0.83	0.69	0.66	0.88	0.75	0.52	0.68	0.59
Data Retention	0.10	0.35	0.16	0.12	0.12	0.12	0.08	0.12	0.09
Do Not Track	0.45	1.0	0.62	1.0	1.0	1.0	0.45	0.40	0.41
Micro-Average	0.53	0.65	0.58	0.66	0.66	0.66	0.60	0.59	0.60

Table 3: Precision/Recall/F1 for the three models. The three starred categories resulted from the decomposition of the original Other category, which is excluded here. Categories are ordered in this table in descending order by frequency in the dataset.

7 Conclusion

We have described the motivation, creation, and analysis of a unique corpus of 115 privacy policies and 23K fine-grained data practice annotations, and we have demonstrated the feasibility of partly automating the annotation process. The annotations reveal the structure and complexity of these documents, which Internet users are expected to understand and accept. This corpus should serve as a resource for language technologies research to help Internet users understand the privacy practices of businesses and other entities that they interact with online.

Acknowledgements

This work is funded by the National Science Foundation under grants CNS-1330596 and CNS-1330214. The authors would like to acknowledge the law students at Fordham University and the University of Pittsburgh who worked as annotators to make this corpus possible. The authors also wish to acknowledge all members of the Usable Privacy Policy Project (`www.usableprivacy.org`) for their contributions.

References

Ngo Xuan Bach, Nguyen Le Minh, Tran Thi Oanh, and Akira Shimazu. 2013. A two-phase framework for learning logical structures of paragraphs in legal articles. *ACM Transactions on Asian Language Information Processing (TALIP)*, 12(1):3:1–3:32, March.

Trevor Bench-Capon, Michał Araszkiewicz, Kevin Ashley, Katie Atkinson, Floris Bex, Filipe Borges, Daniele Bourcier, Paul Bourgine, Jack G Conrad, Enrico Francesconi, et al. 2012. A History of AI and Law in 50 Papers: 25 Years of the International Conference on AI and Law. *Artificial Intelligence and Law*, 20(3):215–319.

Guido Boella, Luigi Di Caro, Michele Graziadei, Loredana Cupi, Carlo Emilio Salaroglio, Llio Humphreys, Hristo Konstantinov, Kornel Marko, Livio Robaldo, Claudio Ruffini, et al. 2015. Linking legal open data: Breaking the accessibility and language barrier in European legislation and case law. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, pages 171–175. ACM.

Kalina Bontcheva, Hamish Cunningham, Ian Roberts, Angus Roberts, Valentin Tablan, Niraj Aswani, and Genevieve Gorrell. 2013. GATE teamware: a web-based, collaborative text annotation framework. *Language Resources and Evaluation*, 47(4):1007–1029.

Travis D. Breaux and Florian Schaub. 2014. Scaling requirements extraction to the crowd. In *RE'14: Proceedings of the 22nd IEEE International Requirements Engineering Conference*, RE' 14, pages 163–172, Washington, DC, USA, August. IEEE Society Press.

Travis D. Breaux, Hanan Hibshi, and Ashwini Rao. 2013. Eddy, a formal language for specifying and analyzing data flow specifications for conflicting privacy requirements. *Requirements Engineering*, 19(3):281–307, September.

Jean Carletta. 1996. Assessing agreement on classification tasks: The kappa statistic. *Computational Linguistics*, 22(2):249–254, June.

F.H. Cate. 2010. The limits of notice and choice. *IEEE Security Privacy*, 8(2):59–62, March.

Parvathi Chundi and Pranav M. Subramaniam. 2014. An approach to analyze web privacy policy documents. In *KDD Workshop on Data Mining for Social Good*.

Elisa Costante, Yuanhao Sun, Milan Petković, and Jerry den Hartog. 2012. A machine learning solution to assess privacy policy completeness. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, WPES '12, pages 91–96. ACM.

Elisa Costante, Jerry den Hartog, and Milan Petković. 2013. What websites know about you: Privacy policy analysis using information extraction. In Roberto Di Pietro, Javier Herranz, Ernesto Damiani, and Radu State, editors, *Data Privacy Management and Autonomous Spontaneous Security*, volume 7731 of *Lecture Notes in Computer Science*, pages 146–159. Springer.

Lorrie Faith Cranor, Kelly Idouchi, Pedro Giovanni Leon, Manya Sleeper, and Blase Ur. 2013. Are they actually any different? Comparing thousands of financial institutions' privacy practices. In *Workshop on the Economics of Information Security*, WEIS '13, June.

Michael Curtotti and Eric McCreath. 2013. A right to access implies a right to know: An open online platform for research on the readability of law. *J. Open Access L.*, 1:1–56.

Tatiana Ermakova, Benjamin Fabian, and Eleonora Babina. 2015. Readability of privacy policies of healthcare websites. In *12. Internationale Tagung Wirtschaftsinformatik (Wirtschaftsinformatik 2015)*.

Federal Trade Commission. 2012. Protecting consumer privacy in an era of rapid change: Recommendations for businesses and policymakers. `http://www.ftc.gov/reports`. Last accessed: June 22, 2016.

Enrico Francesconi, Simonetta Montemagni, Wim Peters, and Daniela Tiscornia. 2010. *Semantic processing of legal texts: Where the language of law meets the law of language*, volume 6036 of *Lecture Notes in Artificial Intelligence*. Springer.

Filippo Galgani, Paul Compton, and Achim Hoffmann. 2012. Combining different summarization techniques for legal text. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, HYBRID '12, pages 115–123. ACL.

Google. 2015. Google Trends. `https://www.google.com/trends/`. Last accessed: June 22, 2016.

Candice Hoke, Lorrie Cranor, Pedro Leon, and Alyssa Au. 2015. Are they worth reading? An in-depth analysis of online tracker's privacy policies. *I/S: A Journal of Law and Policy for the Information Society*, 11(2):325–404, April.

Carlos Jensen and Colin Potts. 2004. Privacy policies as decision-making tools: An evaluation of online privacy notices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 471–478. ACM.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

Fei Liu, Rohan Ramanath, Norman Sadeh, and Noah A. Smith. 2014. A step towards usable privacy policy: Automatic alignment of privacy statements. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, COLING '14, pages 884–894. ANLP, August.

Lars Mahler. 2015. What is NLP and why should lawyers care? `http://www.lawpracticetoday.org/article/nlp-lawyers/`, February. Last accessed: June 22, 2016.

Aaron K. Massey, Jacob Eisenstein, Annie I. Antón, and Peter P. Swire. 2013. Automated text mining for requirements analysis of policy documents. In *21st IEEE International Requirements Engineering Conference*, RE '13, pages 4–13. IEEE.

Aleecia M. McDonald and Lorrie Faith Cranor. 2008. The cost of reading privacy policies. *I/S: J Law & Policy Info. Soc.*, 4(3):540–561.

Gabriele Meiselwitz. 2013. Readability assessment of policies and procedures of social networking sites. In A. Ant Ozok and Panayiotis Zaphiris, editors, *5th International conference, OCSC 2013, Held as Part of HCI International 2013*, Lecture Notes in Computer Science, pages 67–75. Springer, January.

Vytautas Mickevicius, Tomas Krilavicius, and Vaidas Morkevicius. 2015. Classification of short legal Lithuanian texts. In *The 5th Workshop on Balto-Slavic Natural Language Processing*, BSNLP '15, pages 106–111. ACM.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

President's Concil of Advisors on Science and Technology. 2014. Big data and privacy: a technological perspective. Report to the President, Executive Office of the President, May.

Rohan Ramanath, Fei Liu, Norman Sadeh, and Noah A. Smith. 2014. Unsupervised alignment of privacy policies using hidden markov models. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics*, ACL '14, pages 605–610. ACL, June.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50. ELRA, May.

Joel R. Reidenberg, Travis Breaux, Lorrie Faith Cranor, Brian French, Amanda Grannis, James T. Graves, Fei Liu, Aleecia McDonald, Thomas B. Norton, Rohan Ramanath, N. Cameron Russell, Norman Sadeh, and Florian Schaub. 2015a. Disagreeable privacy policies: Mismatches between meaning and users' understanding. *Berkeley Technology Law Journal*, 30(1):39–88.

Joel R. Reidenberg, N. Cameron Russell, Alexander J. Callen, Sophia Qasir, and Thomas B. Norton. 2015b. Privacy harms and the effectiveness of the notice and choice framework. *I/S: J Law & Policy Info. Soc.*, 11(2).

Joel R. Reidenberg, Jaspreet Bhatia, Travis Breaux, and Thomas B. Norton. 2016. Automated comparisons of ambiguity in privacy policies and the impact of regulation. *Social Science Research Network Working Paper Series*, January.

Norman Sadeh, Alessandro Acquisti, Travis D. Breaux, Lorrie Faith Cranor, Aleecia M. Mcdonald, Joel R. Reidenberg, Noah A. Smith, Fei Liu, N. Cameron Russell, Florian Schaub, and Shomir Wilson. 2013. The usable privacy policy project: Combining crowdsourcing, machine learning and natural language processing to semi-automatically answer those privacy questions users care about. Technical Report CMU-ISR-13-119, Carnegie Mellon University.

Giovanni Sartor and Antonino Rotolo. 2013. AI and law. In *Agreement Technologies*, pages 199–207. Springer.

Jaromír Šavelka and Kevin D Ashley. 2015. Transfer of predictive models for classification of statutory texts in multi-jurisdictional settings. In *Proceedings of the 15th International Conference on Artificial Intelligence and Law*, ICAIL '15, pages 216–220. ACM.

Florian Schaub, Rebecca Balebako, Adam L. Durity, and Lorrie Faith Cranor. 2015. A design space for effective privacy notices. In *Eleventh Symposium On Usable Privacy and Security*, SOUPS '15, pages 1–17. USENIX Association, July.

John W. Stamey and Ryan A. Rossi. 2009. Automatically identifying relations in privacy policies. In *27th ACM International Conference on Design of Communication*, SIGDOC '09, pages 233–238. ACM.

Anthony J. Viera and Joanne M. Garrett. 2005. Understanding interobserver agreement: the kappa statistic. *Family Medicine*, 37(5):360–363, 5.

Rigo Wenning, Matthias Schunter, Lorrie Cranor, B. Dobbs, S. Egelman, G. Hogben, J. Humphrey, M. Langheinrich, M. Marchiori, M. Presler-Marshall, J. Reagle, and D. A. Stampley. 2006. The platform for privacy preferences 1.1 (P3P 1.1) specification.

Shomir Wilson, Florian Schaub, Rohan Ramanath, Norman Sadeh, Fei Liu, Noah A. Smith, and Frederick Liu. 2016. Crowdsourcing annotations for websites' privacy policies: Can it really work? In *Proceedings of the 25th World Wide Web Conference*, WWW '13, pages 133–143.

Sebastian Zimmeck and Steven M. Bellovin. 2014. Privee: An architecture for automatically analyzing web privacy policies. In *23rd USENIX Security Symposium*, USENIX Security '14, pages 1–16. USENIX Association, August.

Sequence-based Structured Prediction for Semantic Parsing

Chunyang Xiao and **Marc Dymetman**
Xerox Research Centre Europe
6-8 Chemin de Maupertuis
Meylan, 38240, France
chunyang.xiao, marc.dymetman
@xrce.xerox.com

Claire Gardent
CNRS, LORIA, UMR 7503
Vandoeuvre-lès-Nancy,
F-54500, France
claire.gardent@loria.fr

Abstract

We propose an approach for semantic parsing that uses a recurrent neural network to map a natural language question into a logical form representation of a KB query. Building on recent work by (Wang et al., 2015), the interpretable logical forms, which are structured objects obeying certain constraints, are enumerated by an underlying grammar and are paired with their canonical realizations. In order to use sequence prediction, we need to sequentialize these logical forms. We compare three sequentializations: a direct linearization of the logical form, a linearization of the associated canonical realization, and a sequence consisting of derivation steps relative to the underlying grammar. We also show how grammatical constraints on the derivation sequence can easily be integrated inside the RNN-based sequential predictor. Our experiments show important improvements over previous results for the same dataset, and also demonstrate the advantage of incorporating the grammatical constraints.

1 Introduction

Learning to map natural language utterances (NL) to logical forms (LF), a process known as semantic parsing, has received a lot of attention recently, in particular in the context of building Question-Answering systems (Kwiatkowski et al., 2013; Berant et al., 2013; Berant and Liang, 2014). In this paper, we focus on such a task where the NL question may be semantically complex, leading to a logical form query with a fair amount of compositionality, in a spirit close to (Pasupat and Liang, 2015).

Given the recently shown effectiveness of RNNs (Recurrent Neural Networks), in particular Long Short Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), for performing sequence prediction in NLP applications such as machine translation (Sutskever et al., 2014) and natural language generation (Wen et al., 2015), we try to exploit similar techniques for our task. However we observe that, contrary to those applications which try to predict intrinsically *sequential* objects (texts), our task involves producing a *structured* object, namely a logical form that is tree-like by nature and also has to respect certain *a priori* constraints in order to be interpretable against the knowledge base.

In our case, building on the work "Building a Semantic Parser Overnight" (Wang et al., 2015), which we will refer to as SPO, the LFs are generated by a grammar which is known *a priori*, and it is this grammar that makes explicit the structural constraints that have to be satisfied by the LFs. The SPO grammar, along with generating logical forms, generates so-called "canonical forms" (CF), which are direct textual realizations of the LF that, although they are not "natural" English, transparently convey the meaning of the LF (see Fig. 1 for an example).

Based on this grammar, we explore three different ways of representing the LF structure through a sequence of items. The first one (LF Prediction, or LFP), and simplest, consists in just linearizing the LF tree into a sequence of individual tokens; the second one (CFP) represents the LF through its associated CF, which is itself a sequence of words; and finally the third one (DSP) represents the LF through a derivation sequence (DS), namely the sequence of grammar rules that were chosen to produce this LF.

We then predict the LF via LSTM-based models that take as input the NL question and map it into

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1341–1350,
Berlin, Germany, August 7-12, 2016.

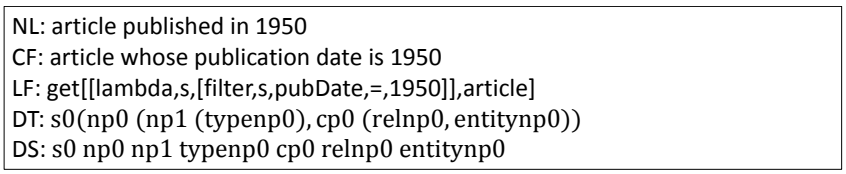


Figure 1: Example of natural language utterance (NL) from the SPO dataset and associated representations considered in this work. CF: canonical form, LF: logical form, DT: derivation tree, DS: derivation sequence.

one of the three sequentializations. In the three cases, the LSTM predictor cannot on its own ensure the grammaticality of the predicted sequence, so that some sequences do not lead to well-formed LFs. However, in the DSP case (in contrast to LFP and CFP), it is easy to integrate inside the LSTM predictor local constraints which guarantee that only grammatical sequences will be produced.

In summary, the contribution of our paper is twofold. Firstly, we propose to use sequence prediction for semantic parsing. Our experimental results show some significant improvements over previous systems. Secondly, we propose to predict derivation sequences taking into account grammatical constraints and we show that the model performs better than sequence prediction models not exploiting this knowledge. These results are obtained without employing any reranking or linguistic features such as POS tags, edit distance, paraphrase features, etc., which makes the proposed methodology even more promising.

2 Background on SPO

The SPO paper (Wang et al., 2015) proposes an approach for quickly developing semantic parsers for new knowledge bases and domains when no training data initially exists. In this approach, a small underlying grammar is used to generate canonical forms and pair them with logical forms. Crowdsourcing is then used to paraphrase each of these canonical forms into several natural utterances. The crowdsourcing thus creates a dataset (SPO dataset in the sequel) consisting of (NL, CF, LF) tuples where NL is a natural language question with CF and LF the canonical and the logical form associated with this question.

SPO learns a semantic parser on this dataset by firstly learning a log-linear similarity model based on a number of features (word matches, ppdb matches, matches between semantic types and POSs, etc.) between NL and the corresponding (CF, LF) pair. At decoding time, SPO parses a natural utterance NL by searching among the derivations of the grammar for one for which the projected (CF, LF) is most similar to the NL based on this log-linear model. The search is based on a so-called "floating parser" (Pasupat and Liang, 2015), a modification of a standard chart-parser, which is able to guide the search based on the similarity features.

In contrast, our approach does not search among the derivations for the one that maximizes a match with the NL, but instead directly tries to predict a decision sequence that can be mapped to the LF.

The SPO system together with its dataset were released to the public[1] and our work exploits this release.

3 Approach

3.1 Grammars and Derivations

```
s0:  s(S) → np(S).
np0: np(get[CP,NP]) → np(NP), cp(CP).
np1: np(NP) → typenp(NP).
cp0: cp([lambda,s,[filter,s,RELNP,=,ENTNP]]) →
       [whose], relnp(RELNP), [is], entitynp(ENTNP).
...
typenp0:   typenp(article) → [article].
relnp0:    relnp(pubDate) → [publication, date]
entitynp0: entitynp(1950) → [1950].
...
```

Figure 2: Some general rules (top) and domain-specific rules (bottom) in DCG format.

The core grammatical resource released by SPO is a generic grammar connecting logical forms with canonical form realizations. They also provide seven domain-specific lexica that can be used in combination with the generic grammar to obtain domain-specific grammars which generate (LF, CF) pairs in each domain, in such a way that LF can then be used to query the corresponding knowledge base. While SPO also released a set of

[1] https://github.com/percyliang/sempre

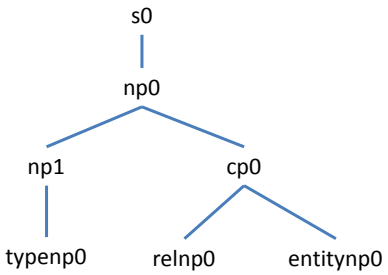


Figure 3: A derivation tree. Its leftmost derivation sequence is [s0, np0, np1, typenp0, cp0, relnp0, entitynp0].

typenp0	*article*
	article
relnp0	*publication date*
	pubDate
entitynp0	*1950*
	1950
cp0	*whose publication date is 1950*
	[lambda,s,[filter,s,pubDate,=,1950]
np1	*article*
	article
np0	*article whose publication date is 1950*
	get[[lambda,s,[filter,s,pubDate,=,1950]],article]
s0	*article whose publication date is 1950*
	get[[lambda,s,[filter,s,pubDate,=,1950]],article]

Figure 4: Projection of the derivation tree nodes into (i) a canonical form and (ii) a logical form.

Java-based parsers and generators for these grammars, for our own purposes we found it convenient to translate the grammars into the formalism of Definite Clause Grammars (Pereira and Warren, 1980), a classical unification-based extension of CFGs, which — through a standard Prolog interpreter such as SWIPL[2] — provide direct support for jointly generating textual realizations and logical forms and also for parsing text into logical forms; we found this translation process to be rather straightforward and we were able to cover all of the SPO grammars.

Figure 2 lists a few DCG rules, general rules first, then lexical rules, for the SPO "publications" domain. Nonterminals are indicated in bold, terminals in italics. We provide each rule with a unique identifier (e.g. s0, np0, ...), which is obtained by concatenating the name of its head nonterminal with a position number relative to the rules that may expand this nonterminal; we can then consider that the nonterminal (e.g. **np**) is the "type" of all its expanding rules (e.g. np0, np1, ...).

According to standard DCG notation, uppercase items `S, NP, CP, RELNP, ENTNP` denote unification variables that become instantiated during processing. In our case unificaion variables range over logical forms and each nonterminal has a single argument denoting a partially instantiated associated logical form. For instance, in the cp0 rule, **`relnp`** is associated with the logical form `RELNP`, **`entitynp`** with the logical form `ENTNP`, and the LHS nonterminal **cp** is then associated with the logical form `[lambda, s, [filter, s, RELNP, =, ENTNP]]`.[3]

In Figure 3, we display a *derivation tree* DT (or simply *derivation*) relative to this grammar, where each node is labelled with a rule identifier. This tree projects on the one hand onto the canonical form *article whose publication date is 1950*, on the other hand onto the logical form `get[[lambda,s,[filter,s,pubDate,=,1950]],article]`.

Figure 4 shows how these projections are obtained by bottom-up composition. For instance, the textual projection of node cp0 is obtained from the textual representations of nodes relnp0 and entitynp0, according to the RHS of the rule cp0, while its logical form projection is obtained by instantiation of the variables `RELNP` and `ENTNP` respectively to the LFs associated with relnp0 and entitynp0.

Relative to these projections, one may note a fundamental difference between derivation trees DT and their projections CF and LF: while the well-formedness of DT can simply be assessed locally by checking that each node expansion is valid according to the grammar, there is in principle no such easy, local, checking possible for the canonical or the logical form; in fact, in order to check the validity of a proposed CF (resp. LF), one needs to *search* for *some* DT that projects onto this CF (resp LF). The first process, of course, is known as "parsing", the second process as "generation". While parsing has polynomial complexity for grammars with a context-free backbone such as the ones considered here, deciding whether a logical form is well-formed or not could in principle be undecidable for certain forms of LF composition.[4]

[2]http://www.swi-prolog.org/

[3]This logical form is written here in DCG list notation; in the more "Lispian" format used by SPO, it would be written `(lambda s (filter s RELNP = ENTNP))`.

[4]The term 'projection' is borrowed from the notion of *bimorphism* in formal language theory (Shieber, 2014) and refers in particular to the fact that the overall logical form is constructed by bottom-up composition of logical forms asso-

To be able to leverage sequence prediction models, we can associate with each derivation tree DT its leftmost *derivation sequence* DS, which corresponds to a preorder traversal of the tree. For the tree of Figure 3, this sequence is [s0, np0, np1, typenp0, cp0, relnp0, entitynp0]. When the grammar is known (in fact, as soon as the CFG core of the grammar is known), two properties of the DS hold (we omit the easy algorithms underlying these properties; they involve using a prefix of the DS for constructing a partial derivation tree in a top-down fashion):

1. knowing the DS uniquely identifies the derivation tree.

2. knowing a prefix of the DS (for instance [s0, np0, np1, typenp0]) completely determines the *type* of the next item (here, this type is **cp**).

The first property implies that if we are able to predict DS, we are also able to predict DT, and therefore also LF and CF. The second property implies that the sequential prediction of DS is strongly constrained by *a priori* knowledge of the underlying grammar: instead of having to select the next item among all the possible rules in the grammar, we only have to select among those rules that are headed by a specific nonterminal. Under a simple condition on the grammar (namely that there are no "unproductive" rules, rules that can never produce an output[5]), following such constrained selection for the next rule guarantees that the derivation sequence will always lead to a valid derivation tree.

At this point, a theoretical observation should be made: *there is no finite-state mechanism on the sequence of rule-names that can control whether the next rule-name is valid or not*.[6] The relevance of that observation for us is that the RNNs that we use are basically finite-state devices (with a huge number of states, but still finite-state), and therefore we do not expect them in principle to be able to always produce valid derivation sequences *unless* they can exploit the underlying grammar for constraining the next choice.

ciated with lower nodes in the derivation tree. In our DCG grammars, this composition actually involves more complex operations (such as "beta-reduction") than the simple copyings illustrated in the small excerpt of Fig. 2.

[5]The general grammar ensures a good coverage of possible logical and canonical forms. However, when this general grammar is used in particular domains, some rules are not relevant any more (i.e. become "unproductive"), but these can be easily eliminated at compile time.

[6]This is easy to see by considering a CFG generating the non finite-state language $a^n b^n$.

3.2 Sequence prediction models

In all these models, we start from a natural utterance NL and we predict a sequence of target items, according to a common sequence prediction architecture that will be described in section 3.3.

3.2.1 Predicting logical form (LFP model)

The most direct approach is to directly predict a linearization of the logical form from NL, the input question. While an LF such as that of Figure 1 is really a structured object respecting certain implicit constraints (balanced parentheses, consistency of the variables bound by lambda expressions, and more generally, conformity with the underlying grammar), the linearization treats it simply as a sequence of tokens: `get [ [ lambda s [ filter s pubDate = 1950 ] ] article ]`. At training time, the LFP model only sees such sequences, and at test time, the next token in the target sequence is then predicted without taking into account any structural constraints. The training regime is the standard one attempting to minimize the cross-entropy of the model relative to the logical forms in the training set.

3.2.2 Predicting derivation sequence (DSP-X models)

Rather than predicting LF directly, we can choose to predict a derivation sequence DS, that is, a sequence of rule-names, and then project it onto LF. We consider three variants of this model.

DSP This basic derivation sequence prediction model is trained on pairs (NL, DS) with the standard training regime. At test time, it is possible for this model to predict ill-formed sequences, which do not correspond to grammatical derivation trees, and therefore do not project onto any logical form.

DSP-C This is a Constrained variant of DSP where we use the underlying grammar to constrain the next rule-name. We train this model exactly as the previous one, but at test time, when sampling the next rule-name inside the RNN, we reject any rule that is not a possible continuation.

DSP-CL This last model is also constrained, but uses a different training regime, with Constrained Loss. In the standard learning regime (used for the

Target sequence	DS	CF	LF
Length	10.5	11.8	47.0
Vocabulary Size	106.0	55.8	59.9

Table 1: Characteristics of different target sequences.

two previous models), the incremental loss when predicting the next item y_t of the sequence is computed as $-\log p(y_t)$, where $p(y_t)$ is the probability of y_t according to the RNN model, normalized (through the computation of a softmax) over *all* the potential values of y_t (namely, here, all the rules in the grammar). By contrast, in the CL learning regime, the incremental loss is computed as $-\log p'(y_t)$, where $p'(y_t)$ is normalized *only over the values of* y_t *that are possible continuations* once the grammar-induced constraints are taken into account, ignoring whatever weights the RNN predictor may (wrongly) believe should be put on impossible continuations. In other words, the DSP-CL model incorporates the prior knowledge about well-formed derivation sequences that we have thanks to the grammar. It computes the actual cross-entropy loss according to the underlying generative process of the model that is used once the constraints are taken into account.

3.2.3 Predicting canonical form (CFP model)

The last possibility we explore is to predict the sequence of words in the canonical form CF, and then use our grammar to parse this CF into its corresponding LF, which we then execute against the knowledge base.[7]

Table 1 provides length and vocabulary-size statistics for the LFP, DSP and CFP tasks.

We see that, typically, for the different domains, DS is a shorter sequence than LF or CF, but its vocabulary size (i.e. number of rules) is larger than that of LF or CF. However DS is unique in allowing us to easily validate grammatical constraints. We also note that the CF is less lengthy than the LF, which uses a number of non "word-like" symbols such as parentheses, lambda variables, and the like.

[7] Although the general intention of SPO is to unambiguously reflect the logical form through the canonical form (which is the basis on which Turkers provide their paraphrases), we do encounter some cases where, although the CF is well-formed and therefore parsable by the grammar, several parses are actually possible, some of which do not correspond to queries for which the KB can return an answer. In these cases, we return the first parse whose logical form does return an answer. Such situations could be eliminated by refining the SPO grammar to a moderate extent, but we did not pursue this.

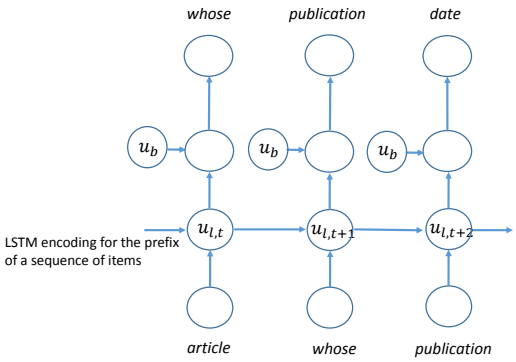


Figure 5: Our neural network model which is shared between all the systems. An MLP encodes the sentence in unigrams and bigrams and produces u_b. An LSTM encodes the prefix of the predicted sequence generating $u_{l,t}$ for each step t. The two representations are then fed into a final MLP to predict the next choice of the target sequence.

3.3 Sequence prediction architecture

3.3.1 Neural network model

The goal of our neural network is to estimate the conditional probability $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$ where $(x_1, \ldots, x_T)$ is a natural language question and $(y_1, \ldots, y_{T'})$ is a target sequence (linearized LF, CF or derivation sequence). In all three cases, we use the same neural network model, which we explain in this subsection.

Suppose that the content of the NL is captured in a real-valued vector u_b, while the prefix of the target sequence up to time t is captured in another real-valued vector $u_{l,t}$. Now, the probability of the target sequence given the input question can be estimated as:

$$p(y_1, \ldots y_{T'}|x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t|u_b, y_1, \ldots y_{t-1})$$
$$= \prod_{t=1}^{T'} p(y_t|u_b, u_{l,t-1})$$

In all our systems, the u_b capturing the content of the NL is calculated from the concatenation of a vector u_1 reading the sentence based on unigrams and another vector u_2 reading the sentence based on bigrams. Mathematically, $u_1 = \tanh(W_1 v_1)$ where v_1 is the 1-hot unigram encoding of the NL

and $u_2 = \tanh(W_2 v_2)$ where v_2 is its 1-hot bigram encoding. Then $u_b = \tanh(Wu)$, where u is the concatenation of u_1 and u_2. W_1, W_2 and W are among the parameters to be learnt. For regularization purposes, a dropout procedure (Srivastava et al., 2014) is applied to u_1 and u_2.

The prefix of the target sequence up to time t is modelled with the vector $u_{l,t}$ generated by the latest hidden state of an LSTM (Hochreiter and Schmidhuber, 1997); LSTM is appropriate here in order to capture the long distance dependencies inside the target sequence. The vector $u_{l,t}$ is then concatenated with u_b (forming u_{bl} in the equation below) before passing through a two-layer MLP (Multi-Layer Perceptron) for the final prediction:

$$p(y_{t+1}|u_{l,t}, u_b) = \text{softmax}(W_2' \tanh(W_1' u_{bl}))$$

Using deep structures such as this MLP for RNN prediction has been shown to be beneficial in previous work (Pascanu et al., 2013).

The overall network architecture is summarized in Figure 5. We train the whole network to minimize the cross entropy between the predicted sequence of items and the reference sequence.

This network architecture can easily support other representations for the input sentence than unigrams and bigrams, as long as they are real-valued vectors of fixed length. We can just concatenate them with u_1 and u_2 and generate u_b as previously. In fact, in initial experiments, we did concatenate an additional representation which reads the sentence through an LSTM, but the performance was not improved.

3.3.2 Decoding the target sequence

We implemented a uniform-cost search algorithm (Russell and Norvig, 2003) to decode the best decision sequence as the sequence with the highest probability. The algorithm finishes in a reasonable time for two reasons: 1) as indicated by Table 1, the vocabulary size of each domain is relatively small, and 2) we found that our model predicts relatively peaked distributions. Of course, it would also be easy to use a beam-search procedure, for situations where these conditions would not hold.

4 Experiments

4.1 Setup

We conduct our experiments on the SPO dataset. To test the overall performance of a semantic parser, the SPO dataset contains seven domains focusing on different linguistic phenomena such as multi-arity relations, sublexical compositionality etc. The utterances in each domain are annotated both with logical forms (LFs) and canonical forms (CFs). The number of such utterances vary from 800 to 4000 depending on the domain. The size of training data is indeed small but as the target vocabulary is always in the domain, thus very small as well, it is actually possible to learn a reasonable semantic parser.

In the SPO dataset, the natural utterances were split randomly into 80%-20% for training and test, and we use the same sets. We perform an additional 80%-20% random split on the SPO training data and keep the 20% as development set to choose certain hyperparameters of our model. Once the hyperparameters are chosen, we retrain on the whole training data before testing.

For LFP experiments, we directly tokenize the LF, as explained earlier, and for CFP experiments we directly use the CF. For DSP experiments (DSP, DSP-C, DSP-CL) where our training data consist of (NL, DS) pairs, the derivation sequences are obtained by parsing each canonical form using the DCG grammar of section 3.

We compare our different systems to SPO. While we only use unigram and bigram features on the NL, SPO uses a number of features of different kinds: linguistic features on NL such as POS tags, lexical features computing the similarity between words in NL and words in CF, semantic features on types and denotations, and also features based on PPDB (Ganitkevitch et al., 2013).

At test time, like SPO, we evaluate our system on the proportion of questions for which the system is able to find the correct answer in the knowledge base.

4.2 Implementation details

We choose the embedding vectors u_1 for unigrams and u_2 for bigrams to have 50 dimensions. The vector u_b representing the sentence content has 200 dimensions. The word embedding layer has 100 dimensions, which is also the case of the hidden layer of the LSTM $u_{l,t}$. Thus u_{bl} which is the concatenation of u_b and $u_{l,t}$ has 300 dimensions and we fix the next layer to u_{bl} to have 100 dimensions. The model is implemented in Keras[8] on top of Theano (Bergstra et al., 2010). For all the exper-

[8] https://github.com/fchollet/keras

iments, we train our models using rmsprop (Tieleman and Hinton., 2012) as the backpropagation algorithm[9]. We use our development set to select the number of training epochs, the dropout factor over unigrams representation and the dropout factor over bigrams representation, by employing a grid search over these hyperparameters: epochs in $\{20, 40, 60\}$, unigrams dropout in $\{0.05, 0.1\}$ and bigrams dropout in $\{0.1, 0.2, 0.3\}$.

4.3 Experimental results

4.3.1 Results on test data

Table 2 shows the test results of SPO and of our different systems over the seven domains.

It can be seen that all of our sequence-based systems are performing better than SPO by a large margin on these tests. When averaging over the seven domains, our 'worst' system DSP scores at 64.7% compared to SPO at 57.1%.

We note that these positive results hold despite the fact that DSP has the handicap that it may generate ungrammatical sequences relative to the underlying grammar, which do not lead to interpretable LFs. The LFP and CFP models, with higher performance than DSP, also may generate ungrammatical sequences.

The best results overall are obtained by the DSP-C system, which does take into account the grammatical constraints. This model performs not only considerably better than its DSP baseline (72.7% over 64.7%), but also better than the models LFP and CFP. Somewhat contrary to our expectations, the DSP-CL model, which exploits constraints not only during decoding, but also during training, performs somewhat worse than the DSP-C, which only exploits them during decoding.

We note that, for all the sequence based models, we strictly base our results on the performance of the *first* sequence predicted by the model. It would probably be possible to improve them further by reranking n-best sequence lists using a set of features similar to those used by SPO.

4.4 Analysis of results

4.4.1 Grammatical errors

We just observed that CFP and LFP perform well on test data although the sequences generated are not guaranteed to be grammatical. We analysed the percentage of grammatical errors made by these models and also by DSP for three domains, which we report in Table 3.[10]

	Basketball	Publication	Housing
LFP	6.6	3.7	1.6
CFP	1.8	1.9	2.2
DSP	9.5	11.8	5.8
DSP-C(L)	0.0	0.0	0.0

Table 3: Grammatical error rate of different systems on test.

The table shows that LFP and especially CFP make few grammatical errors while DSP makes them more frequently. For DSP-C and DSP-CL, the error rate is always 0 since by construction, the derivations must be well-formed. Note that as DSP is not constrained by prior knowledge about the grammar, the grammatical error rate can be high – even higher than CFP or LFP because DSP typically has to choose among more symbols, see Table 1.

4.4.2 Difference between DSP-C and DSP-CL

We observed that the DSP-CL model performs somewhat worse than DSP-C in our experiments. While we were a bit surprised by that behavior, given that the DSP-CL has strong theoretical motivations, let us note that the two models are quite different. To stress the difference, suppose that, for a certain prediction step, only two rules are considered as possible by the grammar, among the many rules of the grammar. Suppose that the LSTM gives probabilities 0.004 and 0.006 respectively to these two rules, the rest of the mass being on the ungrammatical rules. While the DSP-C model associates respective losses of $-\log 0.004, -\log 0.006$ with the two rules, the DSP-CL model normalizes the probabilites first, resulting in smaller losses $-\log 0.4, -\log 0.6$.

As we choose the best complete sequence during decoding, it means that DSP-C will be more likely to prefer to follow a different path in such a case, in order not to incur a loss of at least $-\log 0.006$. Intuitively, this means that DSP-C will prefer paths where the LSTM *on its own*

[9] All the hyperparameters of rmsprop as well as options for initializing the neural network are left at their default values in Keras.

[10] Our DCG permits to compute this error rate directly for canonical forms and derivation sequences. For logical forms, we made an estimation by executing them against the knowledge base and eliminating the cases where the errors are not due to the ungrammaticality of the logical form.

	Basketball	Social	Publication	Blocks	Calendar	Housing	Restaurants	Avg
SPO	46.3	48.2	59.0	41.9	74.4	54.0	75.9	57.1
LFP	73.1	70.2	72.0	55.4	71.4	61.9	76.5	68.6
CFP	80.3	79.5	70.2	54.1	73.2	**63.5**	71.1	70.3
DSP	71.6	67.5	64.0	53.9	64.3	55.0	76.8	64.7
DSP-C	80.5	**80.0**	**75.8**	**55.6**	**75.0**	61.9	**80.1**	**72.7**
DSP-CL	**80.6**	77.6	70.2	53.1	**75.0**	59.3	74.4	70.0

Table 2: Test results over different domains on SPO dataset. The numbers reported correspond to the proportion of cases in which the predicted LF is interpretable against the KB *and* returns the correct answer. LFP = Logical Form Prediction, CFP = Canonical Form Prediction, DSP = Derivation Sequence Prediction, DSP-C = Derivation Sequence constrained using grammatical knowledge, DSP-CL = Derivation Sequence using a loss function constrained by grammatical knowledge.

gives small probability to ungrammatical choices, a property not shared by DSP-CL. However, a more complete understanding of the difference will need more investigation.

5 Related Work and Discussion

In recent work on developing semantic parsers for open-domain and domain-specific question answering, various methods have been proposed to handle the mismatch between natural language questions and knowledge base representations including, graph matching, paraphrasing and embeddings techniques.

Reddy et al. (2014) exploits a weak supervision signal to learn a mapping between the logical form associated by a CCG based semantic parser with the input question and the appropriate logical form in Freebase (Bollacker et al., 2008).

Paraphrase-based approaches (Fader et al., 2013; Berant and Liang, 2014) generate variants of the input question using a simple hand-written grammar and then rank these using a paraphrase model. That is, in their setting, the logical form assigned to the input question is that of the generated sentence which is most similar to the input question.

Finally, Bordes et al. (2014b; 2014a) learn a similarity function between a natural language question and the knowledge base formula encoding its answer.

We depart from these approaches in that we learn a direct mapping between natural language questions and their corresponding logical form or equivalently, their corresponding derivation and canonical form. This simple, very direct approach to semantic parsing eschews the need for complex feature engineering and large external resources required by such paraphrase-based approaches as (Fader et al., 2013; Berant and Liang, 2014). It is conceptually simpler than the two steps, graph matching approach proposed by Reddy et al. (2014). And it can capture much more complex semantic representations than Bordes et al. (2014b; 2014a)'s embeddings based method.[11]

At a more abstract level, our approach differs from previous work in that it exploits the fact that logical forms are structured objects whose shape is determined by an underlying grammar. Using the power of RNN as sequence predictors, we learn to predict, from more or less explicit representations of this underlying grammar, equivalent but different representations of a sentence content namely, its canonical form, its logical form and its derivation sequence.

We observe that the best results are obtained by using the derivation sequence, when also exploiting the underlying grammatical constraints. However the results obtained by predicting directly the linearization of the logical form or canonical form are not far behind; we show that often, the predicted linearizations actually satisfy the underlying grammar. This observation can be related to the results obtained by Vinyals et al. (2014), who use an RNN-based model to map a sentence to the linearization of its parse tree,[12] and find that in most cases, the predicted sequence produces well-balanced parentheses. It would be interest-

[11] In (Bordes et al., 2014b; Bordes et al., 2014a), the logical forms denoting the question answers involve only few RDF triples consisting of a subject, a property and an object i.e., a binary relation and its arguments.

[12] Note a crucial difference with our approach. While in their case the underlying ("syntactic") grammar is only partially and implicitly represented by a set of parse annotations, in our case the explicit ("semantic") grammar is known *a priori* and can be exploited as such.

ing to see if our observation would be maintained for more complex LFs than the ones we tested on, where it might be more difficult for the RNN to predict not only the parentheses, but also the dependencies between several lambda variables inside the overall structure of the LF.

6 Conclusion and Future Work

We propose a sequence-based approach for the task of semantic parsing. We encode the target logical form, a structured object, through three types of sequences: direct linearization of the logical form, canonical form, derivation sequence in an underlying grammar. In all cases, we obtain competitive results with previously reported experiments. The most effective model is one using derivation sequences and taking into account the grammatical constraints.

In order to encode the underlying derivation tree, we chose to use a leftmost derivation sequence. But there are other possible choices that might make the encoding even more easily learnable by the LSTM, and we would like to explore those in future work.

In order to improve performance, other promising directions would involve adding re-reranking techniques and extending our neural networks with attention models in the spirit of (Bahdanau et al., 2015).

Acknowledgements

We would like to thank Guillaume Bouchard for his advice on a previous version of this work, as well as the anonymous reviewers for their constructive feedback. The authors gratefully acknowledge the support of the Association Nationale de la Recherche Technique (ANRT), Convention Industrielle de Formation par la Recherche (CIFRE) No. 2014/0476 .

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

J. Berant and P. Liang. 2014. Semantic parsing via paraphrasing. In *Annual Meeting for the Association for Computational Linguistics (ACL)*.

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, jun. Oral.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD Conference*, pages 1247–1250.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery (ECML-PKDD)*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618.

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

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

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Empirical Methods in Natural Language Processing, (EMNLP)*, pages 1545–1556.

Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *CoRR*, abs/1312.6026.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL Volume 1: Long Papers*.

Fernando C.N. Pereira and David H.D. Warren. 1980. Definite clause grammars for language analysis a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231 – 278.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, 2:377–392.

Stuart J. Russell and Peter Norvig. 2003. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

Stuart M. Shieber. 2014. Bimorphisms and synchronous grammars. *J. Language Modelling*, 2(1):51–104.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958.

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

T. Tieleman and G. E. Hinton. 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2014. Grammar as a foreign language. *Neural Information Processing Systems (NIPS)*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL Volume 1: Long Papers*, pages 1332–1342.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Learning Word Meta-Embeddings

Wenpeng Yin and **Hinrich Schütze**
Center for Information and Language Processing
LMU Munich, Germany
wenpeng@cis.lmu.de

Abstract

Word embeddings – distributed representations of words – in deep learning are beneficial for many tasks in NLP. However, different embedding sets vary greatly in quality and characteristics of the captured information. Instead of relying on a more advanced algorithm for embedding learning, this paper proposes an ensemble approach of combining different public embedding sets with the aim of learning *metaembeddings*. Experiments on word similarity and analogy tasks and on part-of-speech tagging show better performance of metaembeddings compared to individual embedding sets. One advantage of metaembeddings is the increased vocabulary coverage. We release our metaembeddings publicly at http://cistern.cis.lmu.de/meta-emb.

1 Introduction

Recently, deep neural network (NN) models have achieved remarkable results in NLP (Collobert and Weston, 2008; Sutskever et al., 2014; Yin and Schütze, 2015). One reason for these results are *word embeddings*, compact distributed word representations learned in an unsupervised manner from large corpora (Bengio et al., 2003; Mnih and Hinton, 2009; Mikolov et al., 2013a; Pennington et al., 2014).

Some prior work has studied differences in performance of different embedding sets. For example, Chen et al. (2013) show that the embedding sets HLBL (Mnih and Hinton, 2009), SENNA (Collobert and Weston, 2008), Turian (Turian et al., 2010) and Huang (Huang et al., 2012) have great variance in quality and characteristics of the semantics captured. Hill et al. (2014; 2015a) show that embeddings learned by NN machine translation models can outperform three representative monolingual embedding sets: word2vec (Mikolov et al., 2013b), GloVe (Pennington et al., 2014) and CW (Collobert and Weston, 2008). Bansal et al. (2014) find that Brown clustering, SENNA, CW, Huang and word2vec yield significant gains for dependency parsing. Moreover, using these representations together achieved the best results, suggesting their complementarity. These prior studies motivate us to explore an ensemble approach. Each embedding set is trained by a different NN on a different corpus, hence can be treated as a distinct description of words. We want to leverage this *diversity* to learn better-performing word embeddings. Our expectation is that the ensemble contains more information than each component embedding set.

The ensemble approach has two benefits. First, *enhancement* of the representations: metaembeddings perform better than the individual embedding sets. Second, *coverage*: metaembeddings cover more words than the individual embedding sets. The first three ensemble methods we introduce are CONC, SVD and 1TON and they directly only have the benefit of enhancement. They learn metaembeddings on the overlapping vocabulary of the embedding sets. CONC concatenates the vectors of a word from the different embedding sets. SVD performs dimension reduction on this concatenation. 1TON assumes that a metaembedding for the word exists, e.g., it can be a randomly initialized vector in the beginning, and uses this metaembedding to predict representations of the word in the individual embedding sets by projections – the resulting fine-tuned metaembedding is expected to contain knowledge from all individual embedding sets.

To also address the objective of increased coverage of the vocabulary, we introduce 1TON$^+$,

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1351–1360,
Berlin, Germany, August 7-12, 2016.

a modification of 1TON that learns metaembeddings for all words in the *vocabulary union* in one step. Let an out-of-vocabulary (OOV) word w of embedding set ES be a word that is not covered by ES (i.e., ES does not contain an embedding for w).[1] 1TON$^+$ first randomly initializes the embeddings for OOVs and the metaembeddings, then uses a prediction setup similar to 1TON to update metaembeddings *as well as OOV embeddings*. Thus, 1TON$^+$ simultaneously achieves two goals: learning metaembeddings and extending the vocabulary (for both metaembeddings and individual embedding sets).

An alternative method that increases coverage is MUTUALLEARNING. MUTUALLEARNING learns the embedding for a word that is an OOV in embedding set from its embeddings in other embedding sets. We will use MUTUALLEARNING to increase coverage for CONC, SVD and 1TON, so that these three methods (when used together with MUTUALLEARNING) have the advantages of both performance enhancement and increased coverage.

In summary, metaembeddings have two benefits compared to individual embedding sets: *enhancement* of performance and improved *coverage* of the vocabulary. Below, we demonstrate this experimentally for three tasks: word similarity, word analogy and POS tagging.

If we simply view metaembeddings as a way of coming up with better embeddings, then the alternative is to develop a single embedding learning algorithm that produces better embeddings. Some improvements proposed before have the disadvantage of increasing the training time of embedding learning substantially; e.g., the NNLM presented in (Bengio et al., 2003) is an order of magnitude less efficient than an algorithm like word2vec and, more generally, replacing a linear objective function with a nonlinear objective function increases training time. Similarly, fine-tuning the hyperparameters of the embedding learning algorithm is complex and time consuming. In terms of coverage, one might argue that we can retrain an existing algorithm like word2vec on a bigger corpus. However, that needs much longer training time than our simple ensemble approaches which achieve coverage as well as enhancement with less effort. In many cases, it is not possible to retrain using a different algorithm because the corpus is not publicly available. But even if these obstacles could be overcome, *it is unlikely that there ever will be a single "best" embedding learning algorithm*. So the current situation of multiple embedding sets with different properties being available is likely to persist for the forseeable future. Metaembedding learning is a simple and efficient way of taking advantage of this diversity. As we will show below they combine several complementary embedding sets and the resulting metaembeddings are stronger than each individual set.

[1] We do not consider words in this paper that are not covered by any of the individual embedding sets. OOV refers to a word that is covered by a proper subset of ESs.

2 Related Work

Related work has focused on improving performance on specific tasks by using several embedding sets simultaneously. To our knowledge, there is no work that aims to learn generally useful metaembeddings from individual embedding sets.

Tsuboi (2014) incorporates word2vec and GloVe embeddings into a POS tagging system and finds that using these two embedding sets together is better than using them individually. Similarly, Turian et al. (2010) find that using Brown clusters, CW embeddings and HLBL embeddings for Name Entity Recognition and chunking tasks together gives better performance than using these representations individually.

Luo et al. (2014) adapt CBOW (Mikolov et al., 2013a) to train word embeddings on different datasets – a Wikipedia corpus, search clickthrough data and user query data – for web search ranking and for word similarity. They show that using these embeddings together gives stronger results than using them individually.

Both (Yin and Schütze, 2015) and (Zhang et al., 2016) try to incorporate multiple embedding sets into channels of convolutional neural network system for sentence classification tasks. The better performance also hints the complementarity of component embedding sets, however, such kind of incorporation brings large numbers of training parameters.

In sum, these papers show that using multiple embedding sets is beneficial. However, they either use embedding sets trained on the same corpus (Turian et al., 2010) or enhance embedding sets by more training data, not by innovative learning algorithms (Luo et al., 2014), or make the system architectures more complicated (Yin and

	Vocab Size	Dim	Training Data
HLBL (Mnih and Hinton, 2009)	246,122	100	Reuters English newswire August 1996-August 1997
Huang (Huang et al., 2012)	100,232	50	April 2010 snapshot of Wikipedia
Glove (Pennington et al., 2014)	1,193,514	300	42 billion tokens of web data, from Common Crawl
CW (Collobert and Weston, 2008)	268,810	200	Reuters English newswire August 1996-August 1997
word2vec (Mikolov et al., 2013b)	929,022	300	About 100 billion tokens from Google News

Table 1: Embedding Sets (Dim: dimensionality of word embeddings).

Schütze, 2015; Zhang et al., 2016). In our work, we can leverage any publicly available embedding set learned by any learning algorithm. Our metaembeddings (i) do not require access to resources such as large computing infrastructures or proprietary corpora; (ii) are derived by fast and simple ensemble learning from existing embedding sets; and (iii) have much lower dimensionality than a simple concatentation, greatly reducing the number of parameters in any system that uses them.

An alternative to learning metaembeddings from embeddings is the MVLSA method that learns powerful embeddings directly from multiple data sources (Rastogi et al., 2015). Rastogi et al. (2015) combine a large number of data sources and also run two experiments on the embedding sets Glove and word2vec. In contrast, our focus is on metaembeddings, i.e., embeddings that are exclusively based on embeddings. The advantages of metaembeddings are that they outperform individual embeddings in our experiments, that few computational resources are needed, that no access to the original data is required and that embeddings learned by new powerful (including nonlinear) embedding learning algorithms in the future can be immediately taken advantage of without any changes being necessary to our basic framework. In future work, we hope to compare MVLSA and metaembeddings in effectiveness (Is using the original corpus better than using embeddings in some cases?) and efficiency (Is using SGD or SVD more efficient and in what circumstances?).

3 Experimental Embedding Sets

In this work, we use five released embedding sets. (i) **HLBL.** Hierarchical log-bilinear (Mnih and Hinton, 2009) embeddings released by Turian et al. (2010);[2] 246,122 word embeddings, 100 dimensions; training corpus: RCV1 corpus (Reuters English newswire, August 1996 – August 1997). (ii) **Huang.**[3] Huang et al. (2012) incorporate global context to deal with challenges raised by words with multiple meanings; 100,232 word embeddings, 50 dimensions; training corpus: April 2010 snapshot of Wikipedia. (iii) **GloVe**[4] (Pennington et al., 2014). 1,193,514 word embeddings, 300 dimensions; training corpus: 42 billion tokens of web data, from Common Crawl. (iv) **CW** (Collobert and Weston, 2008). Released by Turian et al. (2010);[5] 268,810 word embeddings, 200 dimensions; training corpus: same as HLBL. (v) **word2vec** (Mikolov et al., 2013b) CBOW;[6] 929,022 word embeddings (we discard phrase embeddings), 300 dimensions; training corpus: Google News (about 100 billion words). Table 1 gives a summary of the five embedding sets.

The intersection of the five vocabularies has size 35,965, the union has size 2,788,636.

4 Ensemble Methods

This section introduces the four ensemble methods: CONC, SVD, 1TON and 1TON$^+$.

4.1 CONC: Concatenation

In CONC, the metaembedding of w is the concatenation of five embeddings, one each from the five embedding sets. For GloVe, we perform L2 normalization for each dimension across the vocabulary as recommended by the GloVe authors. Then each embedding of each embedding set is L2-normalized. This ensures that each embedding set contributes equally (a value between -1 and 1) when we compute similarity via dot product.

We would like to make use of prior knowledge and give more weight to well performing embedding sets. In this work, we give GloVe and word2vec weight $i > 1$ and weight 1 to the other three embedding sets. We use MC30 (Miller and Charles, 1991) as dev set, since all embedding sets fully cover it. We set $i = 8$, the value in Figure 1

[2] metaoptimize.com/projects/wordreprs

[3] ai.stanford.edu/~ehhuang

[4] nlp.stanford.edu/projects/glove

[5] metaoptimize.com/projects/wordreprs

[6] code.google.com/p/Word2Vec

where performance reaches a plateau. After L2 normalization, GloVe and word2vec embeddings are multiplied by i and remaining embedding sets are left unchanged.

The dimensionality of CONC metaembeddings is $k = 100+50+300+200+300 = 950$. We also tried equal weighting, but the results were much worse, hence we skip reporting it. It nevertheless gives us insight that simple concatenation, without studying the difference among embedding sets, is unlikely to achieve enhancement. The main disadvantage of simple concatenation is that word embeddings are commonly used to initialize words in DNN systems; thus, the high-dimensionality of concatenated embeddings causes a great increase in training parameters.

4.2 SVD: Singular Value Decomposition

We do SVD on above weighted concatenation vectors of dimension $k = 950$.

Given a set of CONC representations for n words, each of dimensionality k, we compute an SVD decomposition $C = USV^T$ of the corresponding $n \times k$ matrix C. We then use U_d, the first d dimensions of U, as the SVD metaembeddings of the n words. We apply L2-normalization to embeddings; similarities of SVD vectors are computed as dot products.

d denotes the dimensionality of metaembeddings in SVD, 1TON and 1TON$^+$. We use $d = 200$ throughout and investigate the impact of d below.

4.3 1TON

Figure 2 depicts the simple neural network we employ to learn metaembeddings in 1TON. White rectangles denote known embeddings. The target to learn is the metaembedding (shown as shaded rectangle). Metaembeddings are initialized randomly.

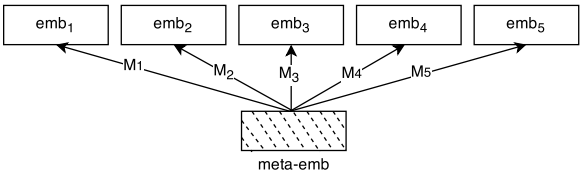


Figure 2: 1toN

Let c be the number of embedding sets under consideration, $V_1, V_2, \ldots, V_i, \ldots, V_c$ their vocabularies and $V^{\cap} = \cap_{i=1}^{c} V_i$ the intersection, *used as training set*. Let V_* denote the metaembedding space. We define a projection f_{*i} from space V_* to space V_i ($i = 1, 2, \ldots, c$) as follows:

$$\hat{\mathbf{w}}_i = \mathbf{M}_{*i}\mathbf{w}_* \quad (1)$$

where $\mathbf{M}_{*i} \in \mathbb{R}^{d_i \times d}$, $\mathbf{w}_* \in \mathbb{R}^d$ is the metaembedding of word w in space V_* and $\hat{\mathbf{w}}_i \in \mathbb{R}^{d_i}$ is the projected (or learned) representation of word w in space V_i. The training objective is as follows:

$$E = \sum_{i=1}^{c} k_i \cdot \Big(\sum_{\mathbf{w} \in V^{\cap}} |\hat{\mathbf{w}}_i - \mathbf{w}_i|^2 + l_2 \cdot |\mathbf{M}_{*i}|^2 \Big) \quad (2)$$

In Equation 2, k_i is the weight scalar of the i^{th} embedding set, determined in Section 4.1, i.e, $k_i = 8$ for GloVe and word2vec embedding sets, otherwise $k_i = 1$; l_2 is the weight of L2 normalization.

The principle of 1TON is that we treat each individual embedding as a projection of the metaembedding, similar to principal component analysis. An embedding is a description of the word based on the corpus and the model that were used to create it. The metaembedding tries to recover a more comprehensive description of the word when it is trained to predict the individual descriptions.

1TON can also be understood as a sentence modeling process, similar to DBOW (Le and Mikolov, 2014). The embedding of each word in a sentence s is a partial description of s. DBOW combines all partial descriptions to form a comprehensive description of s. DBOW initializes the sentence representation randomly, then uses this representation to predict the representations of individual words. The sentence representation of s corresponds to the metaembedding in 1TON; and the representations of the words in s correspond to the five embeddings for a word in 1TON.

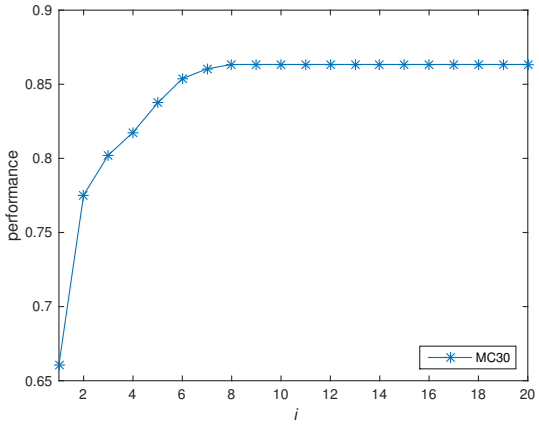


Figure 1: Performance vs. Weight scalar i

4.4 1TON$^+$

Recall that an OOV (with respect to embedding set ES) is defined as a word unknown in ES. 1TON$^+$ is an extension of 1TON that learns embeddings for OOVs; thus, it does not have the limitation that it can only be run on overlapping vocabulary.

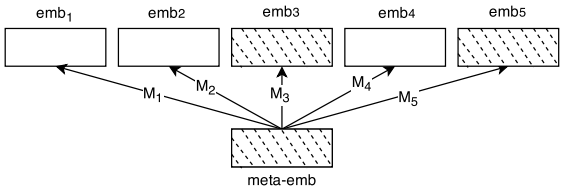


Figure 3: 1toN$^+$

Figure 3 depicts 1TON$^+$. In contrast to Figure 2, we assume that the current word is an OOV in embedding sets 3 and 5. Hence, in the new learning task, embeddings 1, 2, 4 are known, and embeddings 3 and 5 and the metaembedding are targets to learn.

We initialize all OOV representations and metaembeddings randomly and use the same mapping formula as for 1TON to connect a metaembedding with the individual embeddings. Both metaembedding and initialized OOV embeddings are updated during training.

Each embedding set contains information about only a part of the overall vocabulary. However, it can predict what the remaining part should look like by comparing words it knows with the information other embedding sets provide about these words. Thus, 1TON$^+$ learns a model of the dependencies between the individual embedding sets and can use these dependencies to infer what the embedding of an OOV should look like.

CONC, SVD and 1TON compute metaembeddings only for the intersection vocabulary. 1TON$^+$ computes metaembeddings for the union of all individual vocabularies, thus greatly increasing the coverage of individual embedding sets.

5 MUTUALLEARNING

MUTUALLEARNING is a method that extends CONC, SVD and 1TON such that they have increased coverage of the vocabulary. With MUTUALLEARNING, all four ensemble methods – CONC, SVD, 1TON and 1TON$^+$ – have the benefits of both performance enhancement and increased coverage and we can use criteria like performance, compactness and efficiency of training to select the best ensemble method for a particular application.

	bs	lr	l_2
1TON	200	0.005	5×10^{-4}
MUTUALLEARNING (ml)	200	0.01	5×10^{-8}
1TON$^+$	2000	0.005	5×10^{-4}

Table 2: Hyperparameters. bs: batch size; lr: learning rate; l_2: L2 weight.

MUTUALLEARNING is applied to learn OOV embeddings for all c embedding sets; however, for ease of exposition, let us assume we want to compute embeddings for OOVs for embedding set j only, based on known embeddings in the other $c-1$ embedding sets, with indexes $i \in \{1 \dots j-1, j+1 \dots c\}$. We do this by learning $c-1$ mappings f_{ij}, each a projection from embedding set E_i to embedding set E_j.

Similar to Section 4.3, we train mapping f_{ij} on the intersection $V_i \cap V_j$ of the vocabularies covered by the two embedding sets. Formally, $\hat{\mathbf{w}}_j = f_{ij}(\mathbf{w}_i) = \mathbf{M}_{ij}\mathbf{w}_i$ where $\mathbf{M}_{ij} \in \mathbb{R}^{d_j \times d_i}$, $\mathbf{w}_i \in \mathbb{R}^{d_i}$ denotes the representation of word w in space V_i and $\hat{\mathbf{w}}_j$ is the projected metaembedding of word w in space V_j. Training loss has the same form as Equation 2 except that there is no "$\sum_{i=1}^{c} k_i$" term. A total of $c-1$ projections f_{ij} are trained to learn OOV embeddings for embedding set j.

Let w be a word unknown in the vocabulary V_j of embedding set j, but known in $V_1, V_2, \dots, V_k$. To compute an embedding for w in V_j, we first compute the k projections $f_{1j}(\mathbf{w}_1)$, $f_{2j}(\mathbf{w}_2)$, ..., $f_{kj}(\mathbf{w}_k)$ from the source spaces $V_1, V_2, \dots, V_k$ to the target space V_j. Then, the element-wise average of $f_{1j}(\mathbf{w}_1)$, $f_{2j}(\mathbf{w}_2)$, ..., $f_{kj}(\mathbf{w}_k)$ is treated as the representation of w in V_j. Our motivation is that – assuming there is a true representation of w in V_j and assuming the projections were learned well – we would expect all the projected vectors to be close to the true representation. Also, each source space contributes potentially complementary information. Hence averaging them is a balance of knowledge from all source spaces.

6 Experiments

We train NNs by back-propagation with AdaGrad (Duchi et al., 2011) and mini-batches. Table 2 gives hyperparameters.

We report results on three tasks: word similarity, word analogy and POS tagging.

		Model	SL999	WS353	MC30	MEN	RW	sem.	syn.	tot.
ind-full	1	HLBL	22.1 (1)	35.7 (3)	41.5 (0)	30.7 (128)	19.1 (892)	27.1 (423)	22.8 (198)	24.7
	2	Huang	9.7 (3)	61.7 (18)	65.9 (0)	30.1 (0)	6.4 (982)	8.4 (1016)	11.9 (326)	10.4
	3	GloVe	45.3 (0)	75.4 (18)	83.6 (0)	81.6 (0)	48.7 (21)	81.4 (0)	70.1 (0)	75.2
	4	CW	15.6 (1)	28.4 (3)	21.7 (0)	25.7 (129)	15.3 (896)	17.4 (423)	5.0 (198)	10.5
	5	W2V	44.2 (0)	69.8 (0)	78.9 (0)	78.2 (54)	53.4 (209)	77.1 (0)	74.4 (0)	75.6
ind-overlap	6	HLBL	22.3 (3)	34.8 (21)	41.5 (0)	30.4 (188)	22.2 (1212)	13.8 (8486)	15.4 (1859)	15.4
	7	Huang	9.7 (3)	62.0 (21)	65.9 (0)	30.7 (188)	3.9 (1212)	27.9 (8486)	9.9 (1859)	10.7
	8	GloVe	45.0 (3)	75.5 (21)	83.6 (0)	81.4 (188)	59.1 (1212)	91.1 (8486)	68.2 (1859)	69.2
	9	CW	16.0 (3)	30.8 (21)	21.7 (0)	24.7 (188)	17.4 (1212)	11.2 (8486)	2.3 (1859)	2.7
	10	W2V	44.1 (3)	69.3 (21)	78.9 (0)	77.9 (188)	61.5 (1212)	89.3 (8486)	72.6 (1859)	73.3
discard	11	CONC (-HLBL)	**46.0** (3)	**76.5** (21)	**86.3** (0)	**82.2** (188)	**63.0** (1211)	**93.2** (8486)	**74.0** (1859)	**74.8**
	12	CONC (-Huang)	**46.1** (3)	**76.5** (21)	**86.3** (0)	**82.2** (188)	**62.9** (1212)	**93.2** (8486)	**74.0** (1859)	**74.8**
	13	CONC (-GloVe)	44.0 (3)	69.4 (21)	79.1 (0)	77.9 (188)	61.5 (1212)	89.3 (8486)	**72.7** (1859)	**73.4**
	14	CONC (-CW)	**46.0** (3)	**76.5** (21)	**86.6** (0)	**82.2** (188)	**62.9** (1212)	**93.2** (8486)	**73.9** (1859)	**74.7**
	15	CONC (-W2V)	45.0 (3)	75.5 (21)	83.6 (0)	**81.6** (188)	59.1 (1212)	90.9 (8486)	68.3 (1859)	69.2
	16	SVD (-HLBL)	**48.5** (3)	**76.1** (21)	**85.6** (0)	**82.5** (188)	61.5 (1211)	90.6 (8486)	69.5 (1859)	70.4
	17	SVD (-Huang)	**48.8** (3)	**76.5** (21)	**85.4** (0)	**83.0** (188)	**61.7** (1212)	**91.4** (8486)	69.8 (1859)	70.7
	18	SVD (-GloVe)	**46.2** (3)	66.9 (21)	81.6 (0)	78.8 (188)	59.1 (1212)	88.8 (8486)	67.3 (1859)	68.2
	19	SVD (-CW)	**48.5** (3)	**76.1** (21)	**85.7** (0)	**82.5** (188)	61.5 (1212)	90.6 (8486)	69.5 (1859)	70.4
	20	SVD (-W2V)	**49.4** (3)	**79.0** (21)	**87.3** (0)	**83.1** (188)	59.1 (1212)	90.3 (8486)	66.0 (1859)	67.1
	21	1TON (-HLBL)	**46.3** (3)	**75.8** (21)	83.0 (0)	**82.1** (188)	60.5 (1211)	**91.9** (8486)	**75.9** (1859)	**76.5**
	22	1TON (-Huang)	**46.5** (3)	**75.8** (21)	82.3 (0)	**82.4** (188)	60.5 (1212)	**93.5** (8486)	**76.3** (1859)	**77.0**
	23	1TON (-GloVe)	43.4 (3)	67.5 (21)	75.6 (0)	76.1 (188)	57.3 (1212)	89.0 (8486)	**73.8** (1859)	**74.5**
	24	1TON (-CW)	**47.4** (3)	**76.5** (21)	**84.8** (0)	**82.9** (188)	**62.3** (1212)	**91.4** (8486)	**73.1** (1859)	**73.8**
	25	1TON (-W2V)	**46.3** (3)	**76.2** (21)	80.0 (0)	**81.5** (188)	56.8 (1212)	**92.2** (8486)	72.2 (1859)	73.0
	26	1TON$^+$ (-HLBL)	**46.1** (3)	**75.8** (21)	**85.5** (0)	**82.1** (188)	**62.3** (1211)	**92.2** (8486)	**76.2** (1859)	**76.9**
	27	1TON$^+$ (-Huang)	**46.2** (3)	**76.1** (21)	**86.3** (0)	**82.4** (188)	**62.2** (1212)	**93.8** (8486)	**76.1** (1859)	**76.8**
	28	1TON$^+$ (-GloVe)	**45.3** (3)	71.2 (21)	80.0 (0)	78.8 (188)	**62.5** (1212)	90.0 (8486)	**73.3** (1859)	**74.0**
	29	1TON$^+$ (-CW)	**46.9** (3)	**78.1** (21)	**85.5** (0)	**82.5** (188)	**62.7** (1212)	**91.8** (8486)	**73.3** (1859)	**74.1**
	30	1TON$^+$ (-W2V)	**45.8** (3)	**76.2** (21)	**84.4** (0)	81.3 (188)	60.9 (1212)	**92.4** (8486)	72.4 (1859)	73.2
ensemble	31	CONC	**46.0** (3)	**76.5** (21)	**86.3** (0)	**82.2** (188)	**62.9** (1212)	**93.2** (8486)	**74.0** (1859)	**74.8**
	32	SVD	**48.5** (3)	**76.0** (21)	**85.7** (0)	**82.5** (188)	61.5 (1212)	90.6 (8486)	69.5 (1859)	70.4
	33	1TON	**46.4** (3)	74.5 (21)	80.7 (0)	**81.6** (188)	60.1 (1212)	**91.9** (8486)	**76.1** (1859)	**76.8**
	34	1TON$^+$	**46.3** (3)	75.3 (21)	**85.2** (0)	80.8 (188)	**61.6** (1212)	**92.5** (8486)	**76.3** (1859)	**77.0**
	35	state-of-the-art	68.5	81.0	–	–	–	–	–	–

Table 3: Results on five word similarity tasks (Spearman correlation metric) and analogical reasoning (accuracy). The number of OOVs is given in parentheses for each result. "ind-full/ind-overlap": individual embedding sets with respective full/overlapping vocabulary; "ensemble": ensemble results using all five embedding sets; "discard": one of the five embedding sets is removed. If a result is better than all methods in "ind-overlap", then it is bolded. Significant improvement over the best baseline in "ind-overlap" is underlined (online toolkit from `http://vassarstats.net/index.html` for Spearman correlation metric, test of equal proportions for accuracy, $p < .05$).

		RW(21)				semantic				syntactic				total			
		RND	AVG	ml	1TON$^+$	RND	AVG	ml	1TON$^+$	RND	AVG	ml	1TON$^+$	RND	AVG	ml	1TON$^+$
ind	HLBL	7.4	6.9	17.3	17.5	26.3	26.4	26.3	26.4	22.4	22.4	22.7	22.9	24.1	24.2	24.4	24.5
	Huang	4.4	4.3	6.4	6.4	1.2	2.7	21.8	22.0	7.7	4.1	10.9	11.4	4.8	3.3	15.8	16.2
	CW	7.1	10.6	17.3	17.7	17.2	17.2	16.7	18.4	4.9	5.0	5.0	5.5	10.5	10.5	10.3	11.4
ensemble	CONC	14.2	16.5	48.3	–	4.6	18.0	88.1	–	62.4	15.1	74.9	–	36.2	16.3	81.0	–
	SVD	12.4	15.7	47.9	–	4.1	17.5	87.3	–	54.3	13.6	70.1	–	31.5	15.4	77.9	–
	1TON	16.7	11.7	48.5	–	4.2	17.6	88.2	–	60.0	15.0	76.8	–	34.7	16.1	82.0	–
	1TON$^+$	–	–	–	48.8	–	–	–	88.4	–	–	–	76.3	–	–	–	81.1

Table 4: Comparison of effectiveness of four methods for learning OOV embeddings. RND: random initialization. AVG: average of embeddings of known words. ml: MUTUALLEARNING. RW(21) means there are still 21 OOVs for the vocabulary union.

6.1 Word Similarity and Analogy Tasks

We evaluate on SimLex-999 (Hill et al., 2015b), WordSim353 (Finkelstein et al., 2001), MEN (Bruni et al., 2014) and RW (Luong et al., 2013). For completeness, we also show results for MC30, the validation set.

The word analogy task proposed in (Mikolov et al., 2013b) consists of questions like, "a is to b as c is to _ ?". The dataset contains 19,544 such questions, divided into a semantic subset of size 8869 and a syntactic subset of size 10,675. Accuracy is reported.

We also collect the state-of-the-art report for each task. SimLex-999: (Wieting et al., 2015), WS353: (Halawi et al., 2012). Not all state-of-the-art results are included in Table 3. One reason is that a fair comparison is only possible on the shared vocabulary, so methods without released embeddings cannot be included. In addition, some prior systems can possibly generate better performance, but those literature reported lower results than ours because different hyperparameter setup, such as smaller dimensionality of word embeddings or different evaluation metric. In any case, our main contribution is to present ensemble frameworks which show that a combination of complementary embedding sets produces better-performing metaembeddings.

Table 3 reports **results on similarity and analogy**. Numbers in parentheses are the sizes of words in the datasets that are uncovered by intersection vocabulary. We do not consider them for fair comparison. Block "ind-full" (1-5) lists the performance of individual embedding sets on the *full vocabulary*. Results on lines 6-34 are for the intersection vocabulary of the five embedding sets: "ind-overlap" contains the performance of individual embedding sets, "ensemble" the performance of our four ensemble methods and "discard" the performance when one component set is removed.

The four ensemble approaches are very promising (31-34). For CONC, discarding HLBL, Huang or CW does not hurt performance: CONC (31), CONC(-HLBL) (11), CONC(-Huang) (12) and CONC(-CW) (14) beat each individual embedding set (6-10) in all tasks. GloVe contributes most in SimLex-999, WS353, MC30 and MEN; word2vec contributes most in RW and word analogy tasks.

SVD (32) reduces the dimensionality of CONC from 950 to 200, but still gains performance in SimLex-999 and MEN. GloVe contributes most in SVD (larger losses on line 18 vs. lines 16-17, 19-20). Other embeddings contribute inconsistently.

1TON performs well only on word analogy, but it gains great improvement when discarding CW embeddings (24). 1TON$^+$ performs better than 1TON: it has stronger results when considering all embedding sets, and can still outperform individual embedding sets while discarding HLBL (26), Huang (27) or CW (29).

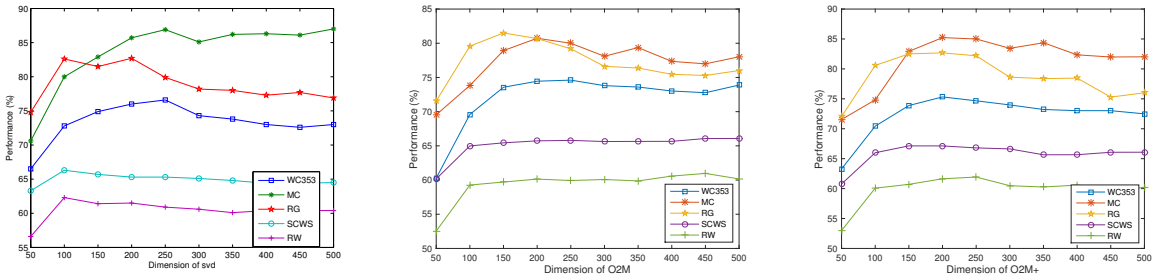
These results demonstrate that ensemble methods using multiple embedding sets produce stronger embeddings. However, *it does not mean the more embedding sets the better. Whether an embedding set helps, depends on the complementarity of the sets and on the task.*

CONC, the simplest ensemble, has robust performance. However, size-950 embeddings as input means a lot of parameters to tune for DNNs. The other three methods (SVD, 1TON, 1TON$^+$) have the advantage of smaller dimensionality. SVD reduces CONC's dimensionality dramatically and still is competitive, especially on word similarity. 1TON is competitive on analogy, but weak on word similarity. 1TON$^+$ performs consistently strongly on word similarity and analogy.

Table 3 uses the metaembeddings of intersection vocabulary, hence it shows directly the *quality enhancement* by our ensemble approaches; this enhancement is not due to bigger coverage.

System comparison of learning OOV embeddings. In Table 4, we extend the vocabularies of each individual embedding set ("ind" block) and our ensemble approaches ("ensemble" block) to the vocabulary union, reporting results on RW and analogy – these tasks contain the most OOVs. As both word2vec and GloVe have full coverage on analogy, we do not rereport them in this table. This subtask is specific to "coverage" property. Apparently, our mutual learning and 1TON$^+$ can cover the union vocabulary, which is bigger than each individual embedding sets. But the more important issue is that we should keep or even improve the embedding quality, compared with their original embeddings in certain component sets.

For each embedding set, we can compute the representation of an OOV (i) as a randomly initialized vector (RND); (ii) as the average of embeddings of all known words (AVG); (iii) by MUTUALLEARNING (ml) and (iv) by 1TON$^+$. 1TON$^+$ learns OOV embeddings for individual embedding sets and metaembeddings simultaneously, and it



(a) Performance vs. d of SVD (b) Performance vs. d of 1TON (c) Performance vs. d of 1TON$^+$

Figure 4: Influence of dimensionality

		newsgroups		reviews		weblogs		answers		emails		wsj	
		ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV	ALL	OOV
baselines	TnT	88.66	54.73	90.40	56.75	93.33	74.17	88.55	48.32	88.14	58.09	95.76	88.30
	Stanford	89.11	56.02	91.43	58.66	94.15	77.13	88.92	49.30	88.68	58.42	96.83	90.25
	SVMTool	89.14	53.82	91.30	54.20	94.21	76.44	88.96	47.25	88.64	56.37	96.63	87.96
	C&P	89.51	57.23	91.58	59.67	94.41	78.46	89.08	48.46	88.74	58.62	96.78	88.65
	FLORS	90.86	66.42	92.95	75.29	94.71	83.64	90.30	62.15	89.44	62.61	96.59	90.37
+indiv	FLORS+HLBL	90.01	62.64	92.54	74.19	94.19	79.55	90.25	62.06	89.33	62.32	96.53	91.03
	FLORS+Huang	90.68	68.53	92.86	77.88	94.71	84.66	90.62	65.04	89.62	64.46	96.65	91.69
	FLORS+GloVe	90.99	70.64	92.84	78.19	94.69	86.16	90.54	65.16	89.75	65.61	96.65	92.03
	FLORS+CW	90.37	69.31	92.56	77.65	94.62	84.82	90.23	64.97	89.32	65.75	96.58	91.36
	FLORS+W2V	90.72	72.74	92.50	77.65	94.75	86.69	90.26	64.91	89.19	63.75	96.40	91.03
+meta	FLORS+CONC	**91.87**	72.64	92.92	**78.34**	**95.37**	86.69	**90.69**	**65.77**	**89.94**	**66.90**	**97.31**	**92.69**
	FLORS+SVD	90.98	70.94	92.47	77.88	94.50	86.49	**90.75**	64.85	**89.88**	**65.99**	96.42	90.36
	FLORS+1TON	**91.53**	**72.84**	**93.58**	78.19	**95.65**	**87.62**	**91.36**	**65.36**	**90.31**	**66.48**	**97.66**	**92.86**
	FLORS+1TON$^+$	**91.52**	72.34	**93.14**	**78.32**	**95.65**	**87.29**	**90.77**	**65.28**	**89.93**	**66.72**	**97.14**	**92.55**

Table 5: POS tagging results on six target domains. "baselines" lists representative systems for this task, including FLORS. "+indiv / +meta": FLORS with individual embedding set / metaembeddings. Bold means higher than "baselines" and "+indiv".

would not make sense to replace these OOV embeddings computed by 1TON$^+$ with embeddings computed by "RND/AVG/ml". Hence, we do not report "RND/AVG/ml" results for 1TON$^+$.

Table 4 shows four interesting aspects. (i) MUTUALLEARNING helps much if an embedding set has lots of OOVs in certain task; e.g., MUTUALLEARNING is much better than AVG and RND on RW, and outperforms RND considerably for CONC, SVD and 1TON on analogy. However, it cannot make big difference for HLBL/CW on analogy, probably because these two embedding sets have much fewer OOVs, in which case AVG and RND work well enough. (ii) AVG produces bad results for CONC, SVD and 1TON on analogy, especially in the syntactic subtask. We notice that those systems have large numbers of OOVs in word analogy task. If for analogy "a is to b as c is to d", all four of a, b, c, d are OOVs, then they are represented with the same average vector. Hence, similarity between $b - a + c$ and each OOV is 1.0. In this case, it is almost impossible to predict the correct answer d. Unfortunately, methods CONC, SVD and 1TON have many OOVs, resulting in the low numbers in Table 4. (iii) MUTUALLEARNING learns very effective embeddings for OOVs. *CONC-ml,* 1TON*-ml and SVD-ml all get better results than word2vec and GloVe on analogy (e.g., for semantic analogy: 88.1, 87.3, 88.2 vs. 81.4 for GloVe).* Considering further their bigger vocabulary, these ensemble methods are very strong representation learning algorithms. (iv) The performance of 1TON$^+$ for learning embeddings for OOVs is competitive with MUTUALLEARNING. For HLBL/Huang/CW, 1TON$^+$ performs slightly better than MUTUALLEARNING in all four met-

rics. Comparing 1TON-ml with 1TON$^+$, 1TON$^+$ is better than "ml" on RW and semantic task, while performing worse on syntactic task.

Figure 4 shows the **influence of dimensionality** d for SVD, 1TON and 1TON$^+$. Peak performance for different data sets and methods is reached for $d \in [100, 500]$. There are no big differences in the averages across data sets and methods for high enough d, roughly in the interval $[150, 500]$. In summary, as long as d is chosen to be large enough (e.g., ≥ 150), performance is robust.

6.2 Domain Adaptation for POS Tagging

In this section, we test the quality of those individual embedding embedding sets and our metaembeddings in a Part-of-Speech (POS) tagging task. For POS tagging, we add word embeddings into FLORS[7] (Schnabel and Schütze, 2014) which is the state-of-the-art POS tagger for unsupervised domain adaptation.

FLORS tagger. It treats POS tagging as a window-based (as opposed to sequence classification), multilabel classification problem using LIBLINEAR,[8] a linear SVM. A word's representation consists of four feature vectors: one each for its suffix, its shape and its left and right distributional neighbors. Suffix and shape features are standard features used in the literature; our use of them in FLORS is exactly as described in (Schnabel and Schütze, 2014).

Let $f(w)$ be the concatenation of the two distributional and suffix and shape vectors of word w. Then FLORS represents token v_i as follows:
$f(v_{i-2}) \oplus f(v_{i-1}) \oplus f(v_i) \oplus f(v_{i+1}) \oplus f(v_{i+2})$
where $\oplus$ is vector concatenation. Thus, token v_i is tagged based on a 5-word window.

FLORS is trained on sections 2-21 of Wall Street Journal (WSJ) and evaluate on the development sets of six different target domains: five SANCL (Petrov and McDonald, 2012) domains – newsgroups, weblogs, reviews, answers, emails – and sections 22-23 of WSJ for in-domain testing.

Original FLORS mainly depends on distributional features. We insert word's embedding as the *fifth* feature vector. All embedding sets (except for 1TON$^+$) are extended to the union vocabulary by MUTUALLEARNING. We test if this additional feature can help this task.

Table 5 gives results for some representative systems ("baselines"), FLORS with individual embedding sets ("+indiv") and FLORS with metaembeddings ("+meta"). Following conclusions can be drawn. (i) Not all individual embedding sets are beneficial in this task; e.g., HLBL embeddings make FLORS perform worse in 11 out of 12 cases. (ii) However, in most cases, embeddings improve system performance, which is consistent with prior work on using embeddings for this type of task (Xiao and Guo, 2013; Yang and Eisenstein, 2014; Tsuboi, 2014). (iii) Metaembeddings generally help more than the individual embedding sets, except for SVD (which only performs better in 3 out of 12 cases).

[7] `cistern.cis.lmu.de/flors` (Yin et al., 2015)
[8] `liblinear.bwaldvogel.de` (Fan et al., 2008)

7 Conclusion

This work presented four ensemble methods for learning metaembeddings from multiple embedding sets: CONC, SVD, 1TON and 1TON$^+$. Experiments on word similarity and analogy and POS tagging show the high quality of the metaembeddings; e.g., they outperform GloVe and word2vec on analogy. The ensemble methods have the added advantage of increasing vocabulary coverage. We make our metaembeddings available at `http://cistern.cis.lmu.de/meta-emb`.

Acknowledgments

We gratefully acknowledge the support of Deutsche Forschungsgemeinschaft (DFG): grant SCHU 2246/8-2.

References

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proceedings of ACL*, pages 809–815.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *JMLR*, 3:1137–1155.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *JAIR*, 49(1-47).

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of ICML*, pages 160–167.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *Proceedings of WWW*, pages 406–414.

Guy Halawi, Gideon Dror, Evgeniy Gabrilovich, and Yehuda Koren. 2012. Large-scale learning of word relatedness with constraints. In *Proceedings of KDD*, pages 1406–1414.

Felix Hill, KyungHyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2014. Not all neural embeddings are born equal. In *NIPS Workshop on Learning Semantics*.

Felix Hill, Kyunghyun Cho, Sebastien Jean, Coline Devin, and Yoshua Bengio. 2015a. Embedding word similarity with neural machine translation. In *Proceedings of ICLR Workshop*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015b. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, pages 665–695.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*, pages 873–882.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*, pages 1188–1196.

Yong Luo, Jian Tang, Jun Yan, Chao Xu, and Zheng Chen. 2014. Pre-trained multi-view word embedding using two-side neural network. In *Proceedings of AAAI*, pages 1982–1988.

Minh-Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*, volume 104, pages 104–113.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*, pages 3111–3119.

George A Miller and Walter G Charles. 1991. Contextual correlates of semantic similarity. *Language and cognitive processes*, 6(1):1–28.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Proceedings of NIPS*, pages 1081–1088.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of EMNLP*, 12:1532–1543.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Proceedings of SANCL*, volume 59.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *Proceedings of NAACL*, pages 556–566.

Tobias Schnabel and Hinrich Schütze. 2014. FLORS: Fast and simple domain adaptation for part-of-speech tagging. *TACL*, 2:15–26.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Yuta Tsuboi. 2014. Neural networks leverage corpus-wide information for part-of-speech tagging. In *Proceedings of EMNLP*, pages 938–950.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of ACL*, pages 384–394.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*, 3:345–358.

Min Xiao and Yuhong Guo. 2013. Domain adaptation for sequence labeling tasks with a probabilistic language adaptation model. In *Proceedings of ICML*, pages 293–301.

Yi Yang and Jacob Eisenstein. 2014. Unsupervised domain adaptation with feature embeddings. In *Proceedings of ICLR Workshop*.

Wenpeng Yin and Hinrich Schütze. 2015. Multichannel variable-size convolution for sentence classification. In *Proceedings of CoNLL*, pages 204–214.

Wenpeng Yin, Tobias Schnabel, and Hinrich Schütze. 2015. Online updating of word representations for part-of-speech taggging. In *Proceedings of EMNLP*, pages 1329–1334.

Ye Zhang, Stephen Roller, and Byron Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. In *Proceedings of NAACL-HLT*.

Towards Constructing Sports News from Live Text Commentary

Jianmin Zhang **Jin-ge Yao** **Xiaojun Wan**
Institute of Computer Science and Technology, Peking University, Beijing 100871, China
Key Laboratory of Computational Linguistic (Peking University), MOE, China
{zhangjianmin2015, yaojinge, wanxiaojun}@pku.edu.cn

Abstract

In this paper, we investigate the possibility to automatically generate sports news from live text commentary scripts. As a preliminary study, we treat this task as a special kind of document summarization based on sentence extraction. We formulate the task in a supervised learning to rank framework, utilizing both traditional sentence features for generic document summarization and novelly designed task-specific features. To tackle the problem of local redundancy, we also propose a probabilistic sentence selection algorithm. Experiments on our collected data from football live commentary scripts and corresponding sports news demonstrate the feasibility of this task. Evaluation results show that our methods are indeed appropriate for this task, outperforming several baseline methods in different aspects.

1 Introduction

There are a huge number of sports games played each day. It is demanding and challenging to write corresponding news reports instantly after various games. Meanwhile, live text commentary services are available on the web and becoming increasingly popular for sports fans who do not have access to live video streams due to copyright reasons. Some people may also prefer live texts on portable devices. The emergence of live texts has produced huge amount of text commentary data. To the best of our knowledge, there exists few studies about utilizing this rich data source.

Manually written sports news for match report usually share the same information and vocabulary as live texts for the corresponding sports game. Sports news and commentary texts can be treated as two different sources of descriptions for the same sports events. It is tempting to investigate whether we can utilize the huge amount of live texts to automatically construct sports news, typically in a form of match report. Building such a system will largely relax the burden of sports news editors, making them free from repetitive tedious efforts for writing while producing sports news more efficiently.

In this work, we study the possibility to construct sports news in the form of match reports from given live text commentary scripts. As a concrete example we collect live text data and corresponding news reports for football (called soccer more often in the United States) games and conduct our study thereby. However, our methods and discussions made in this paper can be trivially adapted to other types of sports games as well.

As a preliminary study, we treat this task as a special kind of document summarization: extracting sentences from live texts to form a match report as generated news. However, generating sports news from live texts is still challenging due to some unique properties of live text commentary scripts. For almost every minute of the game there are usually several sentences describing various kinds of events. Texts are ordered and organized by the timeline, without apparent highlights for many important events [1]. Descriptions are usually in short sentences, which is not helpful for sentence scoring and selection in general. The commentators may tend to use similar, repeated words describing the same type of key events, which may bring additional challenges to traditional summarization methods that are designed to avoid literal repetitions in nature. As a result, naively treating the task as an ordinary document summarization

[1] Some live texts services may use different textual format for scoring events, which is not enough for our more general purposes.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1361–1371,
Berlin, Germany, August 7-12, 2016.

problem can hardly lead to the construction of reasonable sports news reports.

To overcome these difficulties, we explore some specific features of live text commentary scripts and formulate a system based on supervised learning to rank models for this task. In order to tackle the local redundancy issue, we also propose a probabilistic sentence selection strategy.

We summarize our contributions as follows:

- We originally study the task of sports news construction from live text commentary and we build datasets for supervised learning and evaluation for this task.
- We formulate the task in a learning to rank framework, utilizing both traditional features for document summarization and novel task-specific features during supervised learning.
- We propose a probabilistic sentence selection algorithm to address the issue of local redundancy in description.
- We conduct a series of experiments on a real dataset and the evaluation results verify the performance of our system. Results suggest that constructing sports news from live texts is feasible and our proposed methods can outperform a few strong baselines.

2 Problem Statement

2.1 Task Description

In this work, we treat the task of constructing sports news from live text commentary as a special kind of document summarization: extracting sentences from live text scripts to form a match report.

Formally, given a piece of live text commentary containing a collection of candidate sentences $S = \{s_1, s_2, \ldots, s_n\}$ describing a particular sports game G, we need to extract sentences to form a summary of G which are suitable to be formed as sports news. The total length should not exceed a pre-specified length budget B.

The overall framework of generic document summarization can still be retained for this preliminary study. We first rank all candidate sentences according to a sentence scoring scheme and then select a few sentences according to certain criteria to form the final generated news.

2.2 Data Collection

To the best of our knowledge, there does not exist off-the-shelf datasets for evaluating sports news construction. Therefore we have to build a new dataset for this study. We will focus on live text scripts for football (soccer) games as a concrete instance, since football live texts are the easiest to collect. Note that the methods and discussions described in this paper can trivially generalize to other types of sports games.

Meanwhile, live text commentary services are extremely popular in China, where sports fans in many cases do not have access to live video streams due to copyright reasons. The most influential football live services are Sina Sports Live [2] and 163 Football Live [3]. For evaluation purposes we need to simultaneously collect both live texts and news texts describing the same sports games. Due to the convenience and availability of parallel data collection, we build our dataset from Chinese websites. For most football games, there exist both live text scripts recorded after the games and human-written news reports on both Sina Sports and 163 Football. We crawl live text commentary scripts for 150 football matches on Sina Sports Live. Figure 1 displays an example of the format of the live texts, containing the main commentary text along with information of the current timeline and scoreline.

Text Commentary	Timeline	Scoreline
莱万多夫斯基右路传球给到穆勒 (Lewandowski passes the ball to the right and finds Müller)	上半场 42' (first half 42')	2-0
穆勒停球后直接射门 (Müller stops the ball and gets a direct shot)	上半场 43' (first half 43')	2-0
切赫这边反应很快将球托出横梁 (Fast reaction from Cech to tip the ball over the bar)	上半场 43' (first half 43')	2-0

Figure 1: Illustration of the live text format

For every match, two different corresponding sports news reports are collected from Sina Sports Live and 163 Football Matches Live, respectively. These news reports are manually written by professional editors and therefore suitable to be treated as gold-standard news for our task. The average number of sentences in the live texts for one match is around 242, containing around 4,590 Chinese characters for that match. The gold-standard news reports contain 1,185 Chinese characters on average, forming around 32 sentences.

For both the gold-standard news and live text commentary scripts, we split them into sentences

[2] http://match.sports.sina.com.cn/
[3] http://goal.sports.163.com/

and then use a Chinese word segmentation tool [4] to segment the sentences into word sequences. For each sentence, we compute its TFIDF vector for calculating literal cosine similarity when used.

3 Constructing Sports News via Sentence Extraction

We build a system to automatically construct match reports from live text commentary. Since we have described the new challenges for this task, we may design a number of relevant features to address them. In this work, we cast the problem into supervised sentence extraction. Supervised approaches, especially those based on learning to rank (LTR), can better utilize the power of various task-dependent features (Shen and Li, 2011; Wang et al., 2013). For a given specific sports game, we extract features from all candidate sentences in the corresponding live texts and score the sentences using a learning to rank (LTR) model learned from the training data (Section 3.1). Then we select a few of them according to the ranking scores to form the constructed news (Section 3.3).

3.1 Training Data Format

Supervised sentence scoring models based on LTR require input training data in the format of (x_i, y_i) for each candidate sentence s_i, where x_i is the feature vector and y_i is the preference score. The feature vector x is described in Section 3.2. The score y will be defined to reflect the importance, or the tendency to be included in the final news report, of the candidate sentence. In this work we first calculate a group of ROUGE-2 F-scores (cf. Section 4.4.1) of the candidate sentence, treating each sentence in the gold-standard news as reference. The score y of the candidate sentence is then set to be the maximum among those ROUGE-2 F-scores. Later we will see that this scores can indeed serve as good learning targets.

3.2 Features

In this work, we extract both common features which have been widely used for generic document summarization (Shen and Li, 2011; Wang et al., 2013) and novel task-specific features aiming at proper sports news generation from live broadcast script. The features are described as follows.

3.2.1 Basic Features

Position: The position of each candidate sentence. Suppose there are n sentences in a document. For the i-th sentence, its position feature is computed as $1 - \frac{i-1}{n}$.

Length: The number of words contained in the sentence after stopwords removal.

Number of stopwords: The Number of stopwords contained in each sentence. Sentences with many stopwords should be treated as less important candidates.

Sum of word weights: The sum of TF-IDF weights for each word in a sentence.

Similarity to the Neighboring Sentences: We calculate the average cosine similarity of a candidate sentence to its previous N and the next N neighboring sentences. We set N as 1 and 2 here to get two different features.

3.2.2 Task-specific Features

The task we study has some unique properties compared with generic document summarization. For instance, in live text commentary for sports games such as football matches, the scripts not only contain descriptive texts but also the scoreline and timeline information. Such information can be utilized to judge the quality of candidate sentences as well. We extract a rich set of new features, which can be grouped into four types:

Explicit highlight markers: Explicit highlight marker words in a sentence are usually good indicators for its importance. Sentences with more marker words are more probable to be extracted and contained in news or reports for the games. For example, words such as "破门 (scores)" and "红牌 (red card)" in a sentence may indicate that the sentence is describing important events and will be more likely to be extracted. We collect a short list of 25 explicit highlight marker words [5]. For each marker word we create a binary feature to denote the presence or absence of that markers in each candidate sentence. We also use the number of markers as one feature, with the intuition that containing more marker words typically suggests more important sentences.

Scoreline features: An audience of sports games typically pays more attention on scoreline changes, especially those deadlock-breaking scores that break the game from ties. We use three

[4] We use the ICTCLAS toolkit for word segmentation in this work: http://ictclas.nlpir.org/

[5] We include the full list of marker words in the supplementary materials due to the space limit.

binary features to describe the scoreline information of each candidate sentence:

- An indicator feature on whether there was a change of scoreline when the narrator or commentator was producing that sentence.
- An indicator feature on whether the distance between the candidate sentence and the previous closest sentence with a change of scoreline is less than or equal to 5.
- An indicator feature showing whether the game was a draw or not at that time.

To better describe these features we give an example in Figure 2, where S1-S3 corresponds to the above three binary features, respectively.

Text Commentary	Timeline	Scoreline	S1	S2	S3
Both sides take advantages of counter attacks.	32'	1-1	0	0	0
1-2!!	33'	1-2	1	1	1
Alexis!!	33'	1-2	0	1	1
Özil finds the teammate byline followed by a low cross to far post, Alexis sends the ball into the net!	34'	1-2	0	1	1
Leicester players are unhappy.	34'	1-2	0	1	1

Figure 2: An example of scoreline features

Timeline features: The timestamp on each sentence can reflect the progress of a sports game. We divide a match into five different stages as "未赛 (not started)", "上半场(first half)", "中场休息(half-time)", "下半场(second half)" and "完赛(full-time)". Then we use five binary features to represent whether the sentence was describing a specific stage. We also use the specific time-stamp (in integral minutes) of the candidate sentence in the match as an additional feature. Suppose there are n minutes of the match (typically 90 minutes for football), for sentences on the time-stamp of the i-th minute , this feature is computed as $\frac{i}{n}$.

Player popularity: Sports fans usually focus more on the performance of the star players or inform players during the games. We design two features to utilize player information described in a candidate sentence: the number of players contained in the sentence and the sum of their popularity measurements. In this work the popularity of a player is measured using search engines for news: we use the name of a certain player as input query to Baidu News [6], and use the number of recent news retrieved to measure this player's popularity.

[6]http://news.baidu.com/

3.3 Sentence Selection

Once we have the trained LTR model, we can immediately construct news reports by selecting sentences with the highest scores. Unfortunately this simple strategy will suffer from redundancy in commentary, since the LTR scores are predicted independently for each sentence and assigning high scores for repeated commentary texts describing the same key event. Therefore, special care is needed in sentence selection. In principle, any In this work we propose a probabilistic approach based on determinantal point processes (Kulesza and Taskar, 2012, DPPs). This approach can naturally integrate the predicted scores from the LTR model while trying to avoid certain redundancy by producing more diverse extractions [7]. We first review some background knowledge on the model. More details can be found in the comprehensive survey (Kulesza and Taskar, 2012) covering this topic.

3.3.1 Determinantal Point Processes

Determinantal point processes (DPPs) are distributions over subsets that jointly prefer quality of each item and diversity of the whole subset. Formally, a DPP is a probability measure defined on all possible subsets of a group of items $\mathcal{Y} = \{1, 2, \ldots, N\}$. For every $Y \subseteq \mathcal{Y}$ we have:

$$\mathcal{P}(Y) = \frac{\det(L_Y)}{\det(L + I)}$$

where L is a positive semidefinite matrix typically called an *L-ensemble*. $L_Y \equiv [L_{ij}]_{i,j \in Y}$ denotes the restriction of L to the entries indexed by elements of Y, and $\det(L_\emptyset) = 1$. The term $\det(L + I)$ is the normalization constant which has a succinct closed-form and easy to compute. We can define the entries of L as follows:

$$L_{ij} = q_i \phi_i^\top \phi_j q_j = q_i \cdot \mathrm{sim}(i, j) \cdot q_j \qquad (1)$$

where we can think of $q_i \in \mathbb{R}^+$ as the *quality* of an item i and $\phi_i \in \mathbb{R}^n$ with $\|\phi_i\|_2 = 1$ denotes a normalised feature vector such that $\mathrm{sim}(i, j) \in [-1, 1]$ measures *similarity* between item i and item j. This simple definition gives rise to a distribution that places most of its mass on sets that are both high quality and diverse. This is intuitive in a

[7]Many other approaches can also be used to achieve similar effect, such as submodular maximization (Lin and Bilmes, 2010). We leave the comparison with these alternatives for future work study.

geometric sense since determinants are closely related to volumes; in particular, $\det(L_Y)$ is proportional to the volume spanned by the vectors $q_i\phi_i$ for $i \in Y$. Thus, item sets with both high-quality and diverse items will have the highest probability (Figure 3).

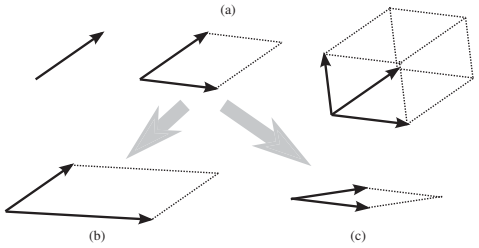


Figure 3: (a) The DPP probability of a set Y depends on the volume spanned by vectors $q_i\phi_i$ for $i \in Y$ (b) As length increases, so does volume. (c) As similarity increases, volume decreases.

3.3.2 Sentence Selection

In this work we formulate the sentence selection problem as maximum a posteriori (MAP) inference for DPPs, i.e. finding $\text{argmax}_Y \log\det(L_Y)$. It is known that MAP inference for DPPs is NP-hard (Gillenwater et al., 2012). Therefore we adopt the greedy approximate inference procedure used by Kulesza and Taskar (2011) which is fast and performs reasonably well in practice.

The remaining question is how to define the L-ensemble matrix L, or equivalently how to define itemwise quality q_i and pairwise similarity $\text{sim}(i, j)$, where each item corresponds to a candidate sentence. Since we have predicted scores for all candidates with the LTR model, we simply set q_i to be the ranking score for sentence i.

The definition of $\text{sim}(i, j)$ is more subtle since it directly address specific types of redundancy. The most straightforward definition is to use literal cosine similarity. This is used for traditional summarization problems (Kulesza and Taskar, 2011). However, the problem for constructing sports news from live broadcast script is rather different. A live broadcast script may use literally similar sentences to describe similar types of events happened at different time stamps. Simply removing sentences that are similar in content may become harmful to the preservation of important events [8].

One typical redundancy that we found in this study is local description redundancy. In live texts, an important event (such as goals) may be stressed multiple times consecutively by the commentator. Therefore in this study we use local literal similarity as a first attempt. Formally, the pairwise similarity is defined as:

$$\text{sim}(i,j) = \begin{cases} 0, \text{if } \max\{|i_p - j_p|, |i_t - j_t|\} > 1, \\ \cos(i,j), \text{ otherwise,} \end{cases}$$

where the subscripts i_p and i_t denotes position and timestamp for sentence i, respectively. In other words we treat sentences written consecutively within one minute as local descriptions and only calculate literal cosine similarity for them.

4 Experimental Setup

4.1 Data Preparation

As described earlier in Section 2.2, we evaluate the performance of different systems on our collected dataset. To utilize the dataset more sufficiently and draw more reliable conclusions, we perform cross-validation during evaluation. Specifically, we randomly divide the dataset into three parts with equal sizes, i.e. each has 50 pairs of live texts and gold-standard news. Each time we set one of them as the test set and use the remaining two parts for training and validation. We will mainly report the averaged results from all three folds. For unsupervised baselines the results are calculated similarly via averaging the performance on the test set.

4.2 Learning to Rank

For predicting ranking scores we use the Random Forest (RF) (Breiman, 2001) ensemble ranker of LambdaMart (Wu et al., 2010), implemented in RankLib [9]. We set the number of iterations to 300 and the sampling rate to 0.3. Using different values did not show real differences.

4.3 Compared Baseline Methods

Our system is compared with several baselines, typically traditional summarization approaches:

HeadTail: Using head and tail sentences only. Commentators usually describe some basic information of the two sides at the beginning and summarize the scoring events in the end of commentary. This baseline resembles the baseline of leading sentences for traditional summarization.

[8] Using cosine similarity for all similarity-dependent methods performs poorly in our experiments. Therefore we will not discuss cosine similarity in more details later.

[9] http://sourceforge.net/p/lemur/wiki/RankLib/; In preliminary experiments, we contrasted RF with support vector regression predictor as well as other pairwise and listwise LTR models. We found that RF consistently outperformed others.

Centroid: In centroid-based summarization (Radev et al., 2000), a pseudo-sentence of the document called centroid is calculated. The centroid consists of words with TFIDF scores above a predefined threshold. The score of each sentence is defined by summing the scores based on different features including cosine similarity of sentences with the centroid, position weight and cosine similarity with the first sentence.

LexRank: LexRank (Erkan and Radev, 2004) computes sentence importance based on the concept of eigenvector centrality in a graph representation of sentences. In this model, a connectivity matrix based on intra-sentence cosine similarity is used as the adjacency matrix of the graph representation of sentences.

ILP: Integer linear programming (ILP) approaches (Gillick et al., 2008) cast document summarization as combinatorial optimization. An ILP model selects sentences by maximizing the sum of frequency-induced weights of bigram concepts [10] contained in the summary.

Highlight: This method is designed to show the effect of using merely the explicit highlight markers described in Section 3.2.2. The importance of a sentence is represented by the number of highlight markers it includes.

For fair comparisons the length of each constructed news report is limited to be no more than 1,000 Chinese characters, roughly the same with the average length of the gold-standard news. Note that we do not use the traditional MMR redundancy removal algorithm based on literal similarity (Carbonell and Goldstein, 1998) since we find only ignorable differences between using MMR or not for all systems.

4.4 Evaluation Methods and Metrics

4.4.1 Automatic Evaluation

Similar to the evaluation for traditional summarization tasks, we use the ROUGE metrics (Lin and Hovy, 2003) to automatically evaluate the quality of produced summaries given the gold-standard reference news. The ROUGE metrics measure summary quality by counting the precision, recall and F-score of overlapping units, such as n-grams and skip grams, between a candidate summary and the reference summaries.

We use the ROUGE-1.5.5 toolkit to perform the evaluation. In this paper we report the F-scores of the following metrics in the experimental results: ROUGE-1 (unigram-based), ROUGE-2 (bigram-based) and ROUGE-SU4 (based on skip bigrams with a maximum skip distance of 4).

4.4.2 Pyramid Evaluation

We also conduct manual pyramid evaluation in this study. Specifically, we use the modified pyramid scores as described in (Passonneau et al., 2005) to manually evaluate the summaries generated by different methods. We randomly sample 20 games from the data set and manually annotate facts on the gold-standard news. The annotated facts are mostly describing specific events happened during the game, e.g. "伊万被黄牌警告" (Ivanovic is shown the yellow card) and "内马尔开出角球" (Neymar takes the corner). Each fact is treated as a Summarization Content Unit, (SCU) (Nenkova and Passonneau, 2004). The number of occurrences for each SCU in the gold-standard news is regarded as the weight of this SCU.

5 Results and Analysis

5.1 Comparison with Baseline Methods

The average performance on all three folds of different methods are displayed in Table 1.

Method	R-1	R-2	R-SU4
HeadTail	0.30147	0.07779	0.10336
Centroid	0.32508	0.08113	0.11245
LexRank	0.31284	0.06159	0.09376
ILP	0.32552	0.07285	0.10378
Highlight	0.34687	0.08748	0.11924
RF	0.38559	0.11887	0.14907
RF+DPP	0.39391	0.11986	0.15097

Table 1: Comparison results of different methods

As we can see from the results, our learning to rank approach based on RF achieves significantly (< 0.01 significance level for pairwise-t testing) better results compared with traditional unsupervised summarization approaches [11]. The ILP model, which is believed to be suitable for multi-document summarization, did not perform well in our settings. Head and tail sentences are informative but merely using them lacks specific descriptions for procedural events, therefore not

[10] We also tried words rather than bigrams but found slightly worse performance.

[11] We also conducted experiments on using our proposed features to calculate LexRank, but did not observe real difference compared with normal LexRank. This suggest that the performance gain comes from supervised learning to rank approach, not merely from the features.

providing competitive results either.

The comparison between RF and RF+DPP shows the effectiveness of our sentence selection strategy. However, the increase is still limited [12]. This may become reasonable later when we discuss more about the errors from our systems.

Merely using highlight markers to construct news also provides competitive results, but inferior to supervised models. This suggests that the highlight marker features are relatively strong indicators for good sentences while merely using these features may not be sufficient.

Table 2 shows the average pyramid scores for the systems in comparison. The "Gold-standard" row denotes manually written news report and is listed for reference. We can see our learning to rank systems based on RF constructs news with the highest pyramid scores.

Method	Pyramid scores
HeadTail	0.13657
Centroid	0.30663
LexRank	0.28756
ILP	0.20867
Highlight	0.41121
RF	0.53766
RF+DPP	0.62500
Gold-standard	0.88329

Table 2: Average Pyramid scores

Overall, the experimental results indicate that our system can generate much better news than the baselines in both automatic and manual evaluations. We include examples of our constructed news reports in the supplementary materials.

5.2 Feature Validation

Different groups of features may play different roles in the LTR models. In order to validate the impact of both the traditional features and the novel task-specific features, we conduct experiments with different combinations by removing each group of features respectively. Table 3 shows the results, with "w/o" denotes experiments without the corresponding group of features.

Method	R-1	R-2	R-SU4
RF	0.38559	0.11887	0.14907
RF-w/o novel	0.37297	0.10964	0.14021
RF-w/o trad.	0.36314	0.09910	0.13102

Table 3: Results of feature validation

[12] Significance level < 0.05 for pairwise-t testing only for ROUGE-1.

We can observe that both the traditional features and the novel features contribute useful information for learning to rank models. Due to the nature of the sentence extraction approach, features designed for traditional document summarization are still playing an indispensable role for our task, although they might be important in this work for different reasons. For example, position features are indicative for traditional summarization since sentences appearing in the very beginning or the end are more probable as summarizing sentences. For sports commentary, positions are closely related to timeline in a more coarse fashion. Certain types of key events, for example player substitutions and even scores, may tend to happen in certain period in a game rather than uniformly spread out in every minute.

5.3 Room for Improvements

5.3.1 Upper Bounds

To get a rough estimate of what is actually achievable in terms of the final ROUGE scores, we looked at different "upper bounds" under various scenarios (Table 4). We first evaluate one *reference* news with the other reference news served as the gold-standard result. The results are given in the row labeled *reference* of Table 4. This provides a reasonable estimate of human performance.

Second, in sentence extraction we restrict the constructed news to sentences from the original commentary texts themselves. We use the greedy algorithm to *extract* sentences that maximize ROUGE-2F scores. The resulting performance is given in the row *extract* of Table 4. We observe numerically superior scores compared with *reference*. This is not strange since we are intentionally optimizing ROUGE scores. And also this suggests that the sentence extraction approach for sports news construction is rather reasonable, in terms of information overlap.

Method	R-1	R-2	R-SU4
reference	0.44725	0.15265	0.18064
extract	0.43270	0.16872	0.18622
target	0.40987	0.15901	0.17941
target+DPP	0.41536	0.15994	0.18232
RF+DPP	0.39391	0.11986	0.15097

Table 4: Upper bounds on ROUGE scores

Third, we use the partial ROUGE-2 values, i.e. the targets used to train LTR models (cf. Section 3.1) for greedy selection and DPP selection, with results listed in the row *target* and *tar-*

Time	Live Text Commentary Script
55	内马尔 为 巴 萨 制造 了 一个 位置 不错 的 定位球 Neymar wins a free kick in a good position.
56	内马尔 ~~~ Neymar!!!!!!!
56	内马尔 的 定位球 直接 打入 了 球门 左上 角 的 死角 ！ ！ ！ 门将 无能为力 The free kick from Neymar goes directly into the top left corner! The keeper can do nothing.
56	球 在 飞 向 球门 的 过程 中 下坠 速度 非常 快 The ball drops quickly and flies to the goal.

Figure 4: Case I: short and noisy sentences

Time	Live Text Commentary Script
FT	切尔西中场快发任意球，科斯塔打进全场唯一入球！！！ From a quick free kick from Chelsea, Costa scored the only goal of the game!!!
FT	整场比赛切尔西占据了主动，可面对诺维奇的铁桶阵，办法不多 Chelsea dominated the game but found it difficult against Norwich's defense.
FT	下半场利用对方的一次疏忽，阿扎尔中场被放倒，威廉快发任意球，科斯塔完成致命一击 From an error from the opponent, Hazard was fouled. Willian launches a quick free kick and assists Costa for the lethal strike.

Figure 5: Case II: summarizing sentences

get+DPP of Table 4. This validates that using partial ROUGE-2 as the training target for LTR models is somewhat reasonable for this study.

5.3.2 Error Analysis

In this preliminary study, we use LTR models and probabilistic sentence selection procedure. While reasonable performance has been achieved, there exist certain types of errors as we found in the constructed news results.

Error I: First, sentences in live commentary are mostly short, and sometimes noisy. Sometimes an important event has been described using a number of consecutive short sentences. Our LTR models failed to generate high scores for such sentences and therefore will cause some lack of information. Figure 4 illustrates an example of this type of error in the constructed news report. All the sentences are describing a key scoring event. However, none of them were selected to construct the news because our LTR model assigns low scores for these short sentences. Meanwhile the second sentence can be treated as noisy.

Error II: Second, commentators are likely to summarize important events during the game, not at the point when the event happens. Our sentence selection algorithm can only address local redundancy, while this issue is more global. Figure 5 illustrates an example of this case in the constructed news report. The only goal of the match is described during full-time (FT). Our method redundantly included this in the final constructed news even it had already selected that event.

These two issues are highly non-trivial and have not been well addressed in the method we explored in this paper. We leave them for further study in the future.

5.3.3 Readability Assessment

In this work we only consider sentence extraction. Unlike traditional summarization tasks, sports commentary texts are describing a different specific action in almost every sentence. Descriptive coherence becomes a more difficult challenge in this scenario. We conduct manual evaluation on systems in comparison along with manually written news reports (gold-standard). Three volunteers who are fluent in Chinese were asked to perform manual ratings on three factors: coherence (Coh.), non-redundancy (NR) and overall readability (Read.). The ratings are in the format of 1-5 numerical scores (not necessarily integral), with higher scores denote better quality. The results are shown in Table 5.

Method	Coh.	NR	Read.
HeadTail	3.47	3.07	3.56
Centroid	2.87	3.72	2.66
LexRank	2.90	3.23	2.43
ILP	2.87	3.23	2.50
Highlight	3.36	3.72	3.06
RF	3.23	3.64	3.13
RF+DPP	3.23	3.87	3.06
Gold-Standard	4.67	4.23	4.77

Table 5: Manual readability ratings

The differences between systems in terms of readability factors are not as large as information coverage suggested by ROUGE metrics and pyramid scores. Meanwhile, while we can observe that our approaches outperforms the unsupervised extractive summarization approaches in coherence and readability for certain level, the results also clearly suggest that there still exists large room for improvements in terms of the readability factors.

6 Discussions

The general challenges for the particular task of sports news generation are mostly addressed in those designed features in the learning to rank framework. We utilize the timeline and scoreline information, while also keep traditional features such as sentence length. Experimental results show that our framework indeed outperforms strong traditional summarization baselines, while still having much room for improvement.

We might also notice that there may exist some issues if merely using automatic metrics to evaluate the overall quality of the generated news reports. The ROUGE metrics are mainly based on

ngram overlaps. For sports texts most of the proportions are dominated by proper names, certain types of actions or key events, etc. Compared with traditional summarization tasks, it might be easier to achieve high ROUGE scores with an emphasize on selecting important entities. In our experiments, methods with higher ROUGE scores can indeed achieve better coverage of important units such as events, as shown in pyramid scores in Table 2. However, we can also observe from Table 5 that automatic metrics currently cannot reflect readability factors very well. Generally speaking, while big difference in ROUGE may suggest big difference in overall quality, smaller ROUGE differences may not be that indicative enough. Therefore, it is interesting to find alternative automatic metrics in order to better reflect the general quality for this task.

7 Related Work

To the best of our knowledge, generation of sports news from live text commentary is not a well-studied task in related fields. One related study focused on generating textual summaries for sports events from status updates in Twitter (Nichols et al., 2012). There also exists earlier work on generation of sports highlight frames from sports videos, focusing on a very different type of data (Tjondronegoro et al., 2004). Bouayad-Agha et al. (2011) and Bouayad-Agha et al. (2012) constructed an ontology-based knowledge base for the generation of football summaries, using predefined extraction templates.

Our task is closely related to document summarization, which has been studied quite intensively. Various approaches exist to challenge the document summarization task, including centroid-based methods, link analysis and graph-based algorithms (Erkan and Radev, 2004; Wan et al., 2007), combinatorial optimization techniques such as integer linear programming (Gillick et al., 2008) and submodular optimization (Lin and Bilmes, 2010). Supervised models including learning to rank models (Metzler and Kanungo, 2008; Shen and Li, 2011; Wang et al., 2013) and regression (Ouyang et al., 2007; Galanis and Malakasiotis, 2008; Hong and Nenkova, 2014) have also been adapted in the scenario of document summarization.

Since sports live texts contain timeline information, summarization paradigms that utilize timeline and temporal information (Yan et al., 2011; Ng et al., 2014; Li et al., 2015) are also conceptually related. Supervised approaches related to this work have also been applied for timeline summarization, including linear regression for important scores (Tran et al., 2013a) and learning to rank models (Tran et al., 2013b). In this preliminary work we only use the timestamps in the definition of similarity for sentence selection. More crafted usages will be explored in the future.

8 Conclusion and Future Work

In this paper we study a challenging task to automatically construct sports news from live text commentary. Using football live texts as an instance, we collect training data jointly from live text commentary services and sports news portals. We develop a system based on learning to rank models, with several novel task-specific features. To generate the final news summary and tackle the local redundancy problem, we also propose a probabilistic sentence selection method. Experimental results demostrate that this task is feasible and our proposed methods are appropriate.

As a preliminary work, we only perform sentence extraction in this work. Since sports news and live commentary are in different genres, some post-editing rewritings will make the system generating more natural descriptions for sports news. We would like to extend our system to produce sports news beyond pure sentence extraction.

Another important direction is to focus on the construction of datasets in larger scale. One feasible approach is to use a speech recognition system on live videos or broadcasts of sports games to collect huge amount of transcripts as our raw data source. Although more data can be easily collected in this case, the noisiness of audio transcripts may bring some additional challenges, therefore worthwhile for further study.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61331011), National Hi-Tech Research and Development Program (863 Program) of China (2015AA015403) and IBM Global Faculty Award Program. We thank the anonymous reviewers for helpful comments and Kui Xu from our group for his help in calculating player popularity features. Xiaojun Wan is the corresponding author of this paper.

References

Nadjet Bouayad-Agha, Gerard Casamayor, and Leo Wanner. 2011. Content selection from an ontology-based knowledge base for the generation of football summaries. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 72–81. Association for Computational Linguistics.

Nadjet Bouayad-Agha, Gerard Casamayor, Simon Mille, and Leo Wanner. 2012. Perspective-oriented generation of football match summaries: Old tasks, new challenges. *ACM Transactions on Speech and Language Processing (TSLP)*, 9(2):3.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Jaime Carbonell and Jade Goldstein. 1998. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Dimitrios Galanis and Prodromos Malakasiotis. 2008. Aueb at tac 2008. In *Proceedings of the TAC 2008 Workshop*.

Jennifer Gillenwater, Alex Kulesza, and Ben Taskar. 2012. Near-optimal map inference for determinantal point processes. In *Advances in Neural Information Processing Systems*, pages 2735–2743.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tur. 2008. The icsi summarization system at tac 2008. In *Proceedings of the Text Understanding Conference*.

Kai Hong and Ani Nenkova. 2014. Improving the estimation of word importance for news multi-document summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 712–721, Gothenburg, Sweden, April. Association for Computational Linguistics.

Alex Kulesza and Ben Taskar. 2011. Learning determinantal point processes. In *UAI*.

Alex Kulesza and Ben Taskar. 2012. Determinantal point processes for machine learning. *Foundations and Trends in Machine Learning*, 5(2–3).

Chen Li, Yang Liu, and Lin Zhao. 2015. Improving update summarization via supervised ilp and sentence reranking. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1317–1322, Denver, Colorado, May–June. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 912–920. Association for Computational Linguistics.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 71–78. Association for Computational Linguistics.

Donald Metzler and Tapas Kanungo. 2008. Machine learned sentence selection strategies for query-biased summarization. In *SIGIR Learning to Rank Workshop*, pages 40–47.

Ani Nenkova and Rebecca Passonneau. 2004. Evaluating content selection in summarization: The pyramid method.

Jun-Ping Ng, Yan Chen, Min-Yen Kan, and Zhoujun Li. 2014. Exploiting timelines to enhance multi-document summarization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 923–933, Baltimore, Maryland, June. Association for Computational Linguistics.

Jeffrey Nichols, Jalal Mahmud, and Clemens Drews. 2012. Summarizing sporting events using twitter. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 189–198. ACM.

You Ouyang, Sujian Li, and Wenjie Li. 2007. Developing learning strategies for topic-based summarization. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 79–86. ACM.

Rebecca J Passonneau, Ani Nenkova, Kathleen McKeown, and Sergey Sigelman. 2005. Applying the pyramid method in duc 2005. In *Proceedings of the Document Understanding Conference (DUC 05), Vancouver, BC, Canada*.

Dragomir R Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic summarization*, pages 21–30. Association for Computational Linguistics.

Chao Shen and Tao Li. 2011. Learning to rank for query-focused multi-document summarization. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*, pages 626–634. IEEE.

Dian Tjondronegoro, Yi-Ping Phoebe Chen, and Binh Pham. 2004. Integrating highlights for more complete sports video summarization. *IEEE multimedia*, 11(4):22–37.

Giang Binh Tran, Mohammad Alrifai, and Dat Quoc Nguyen. 2013a. Predicting relevant news events for timeline summaries. In *Proceedings of the 22nd international conference on World Wide Web Companion*, pages 91–92. International World Wide Web Conferences Steering Committee.

Giang Binh Tran, Tuan A Tran, Nam-Khanh Tran, Mohammad Alrifai, and Nattiya Kanhabua. 2013b. Leveraging learning to rank in an optimization framework for timeline summarization. In *SIGIR 2013 Workshop on Time-aware Information Access (TAIA*.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, volume 7, pages 2903–2908.

Lu Wang, Hema Raghavan, Vittorio Castelli, Radu Florian, and Claire Cardie. 2013. A sentence compression based framework to query-focused multi-document summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1384–1394, Sofia, Bulgaria, August. Association for Computational Linguistics.

Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270.

Rui Yan, Liang Kong, Congrui Huang, Xiaojun Wan, Xiaoming Li, and Yan Zhang. 2011. Timeline generation through evolutionary trans-temporal summarization. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 433–443, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

A Continuous Space Rule Selection Model for Syntax-based Statistical Machine Translation

Jingyi Zhang[1,2], **Masao Utiyama**[1], **Eiichro Sumita**[1]
Graham Neubig[2], **Satoshi Nakamura**[2]
[1]National Institute of Information and Communications Technology,
3-5Hikaridai, Keihanna Science City, Kyoto 619-0289, Japan
[2]Graduate School of Information Science, Nara Institute of Science and Technology,
Takayama, Ikoma, Nara 630-0192, Japan
jingyizhang/mutiyama/eiichiro.sumita@nict.go.jp
neubig/s-nakamura@is.naist.jp

Abstract

One of the major challenges for statistical machine translation (SMT) is to choose the appropriate translation rules based on the sentence context. This paper proposes a continuous space rule selection (CSRS) model for syntax-based SMT to perform this context-dependent rule selection. In contrast to existing maximum entropy based rule selection (MERS) models, which use discrete representations of words as features, the CSRS model is learned by a feed-forward neural network and uses real-valued vector representations of words, allowing for better generalization. In addition, we propose a method to train the rule selection models only on minimal rules, which are more frequent and have richer training data compared to non-minimal rules. We tested our model on different translation tasks and the CSRS model outperformed a baseline without rule selection and the previous MERS model by up to 2.2 and 1.1 points of BLEU score respectively.

1 Introduction

In syntax-based statistical machine translation (SMT), especially tree-to-string (Liu et al., 2006; Graehl and Knight, 2004) and forest-to-string (Mi et al., 2008) SMT, a source tree or forest is used as input and translated by a series of tree-based translation rules into a target sentence. A tree-based translation rule can perform reordering and translation jointly by projecting a source subtree into a target string, which can contain both terminals and nonterminals.

One of the difficulties in applying this model is the ambiguity existing in translation rules: a source subtree can have different target translations extracted from the parallel corpus as shown in Figure 1. Selecting correct rules during decoding is a major challenge for SMT in general, and syntax-based models are no exception.

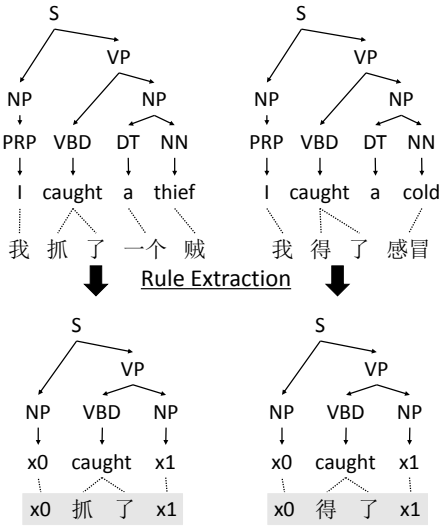


Figure 1: An ambiguous source subtree with different translations (English-to-Chinese).

There have been several methods proposed to resolve this ambiguity. The most simple method, used in the first models of tree-to-string translation (Liu et al., 2006), estimated the probability of a translation rule by relative frequencies. For example, in Figure 1, the rule that occurs more times in the training data will have a higher score. Later, Liu et al. (2008) proposed a maximum entropy based rule selection (MERS, Section 2) model for syntax-based SMT, which used contextual information for rule selection, such as words surrounding a rule and words covered by nonterminals in a rule. For example, to choose the correct rule from the two rules in Figure 1 for decoding a particular input sentence, if the source phrase covered by "x1" is "a thief" and this child phrase

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1372–1381,
Berlin, Germany, August 7-12, 2016.

has been seen in the training data, then the MERS model can use this information to determine that the first rule should be applied. However, if the source phrase covered by "x1" is a slightly different phrase, such as "a gunman", it will be hard for the MERS model to select the correct rule, because it treats "thief" and "gunman" as two different and unrelated words.

In this paper, we propose a continuous space rule selection (CSRS, Section 3) model, which is learned by a feed-forward neural network and replaces the discrete representations of words used in the MERS model with real-valued vector representations of words for better generalization. For example, the CSRS model can use the similarity of word representations for "gunman" and "thief" to infer that "a gunman" is more similar with "a thief" than "a cold".

In addition, we propose a new method, applicable to both the MERS and CSRS models, to train rule selection models only on minimal rules. These minimal rules are more frequent and have richer training data compared to non-minimal rules, making it possible to further relieve the data sparsity problem.

In experiments (Section 4), we validate the proposed CSRS model and the minimal rule training method on English-to-German, English-to-French, English-to-Chinese and English-to-Japanese translation tasks.

2 Tree-to-String SMT and MERS

2.1 Tree-to-String SMT

In tree-to-string SMT (Liu et al., 2006), a parse tree for the source sentence F is transformed into a target sentence E using translation rules R. Each tree-based translation rule $r \in R$ translates a source subtree $\tilde{t}$ into a target string $\tilde{e}$, which can contain both terminals and nonterminals. During decoding, the translation system examines different derivations for each source sentence and outputs the one with the highest probability,

$$\hat{E} = \arg\max_{E,R} \Pr(E, R|F). \quad (1)$$

For a translation E of a source sentence F with derivation R, the translation probability is calculated as follows,

$$\Pr(E, R|F) \approx \frac{\exp\left(\sum_{k=1}^{K} \lambda_k h_k(E, R, F)\right)}{\sum_{E',R'} \exp\left(\sum_{k=1}^{K} \lambda_k h_k(E', R', F)\right)}. \quad (2)$$

Here, h_k are features used in the translation system and λ_k are feature weights. Features used in Liu et al. (2006)'s model contain a language model and simple features based on relative frequencies, which do not consider context information.

One of the most important features used in this model is based on the log conditional probability of the target string given the input source subtree $\log \Pr(\tilde{e}|\tilde{t})$. This allows the model to determine which target strings are more likely to be used in translation. However, as the correct translation of the rules may depend on context that is not directly included in the rule, this simple context-independent estimate is inherently inaccurate.

2.2 Maximum Entropy Based Rule Selection

To perform context-dependent rule selection, Liu et al. (2008) proposed the MERS model for syntax-based SMT. They built a maximum entropy classifier for each ambiguous source subtree $\tilde{t}$, which introduced contextual information C and estimated the conditional probability using a log-linear model as shown below,

$$\Pr(\tilde{e}|\tilde{t}, C) = \frac{\exp\left(\sum_{k=1}^{K} \lambda_k h_k(\tilde{e}, C)\right)}{\sum_{\tilde{e}'} \exp\left(\sum_{k=1}^{K} \lambda_k h_k(\tilde{e}', C)\right)}. \quad (3)$$

The target strings $\tilde{e}$ are treated as different classes for the classifier.

Supposing that,

- r covers source span $[f_\varphi, f_\vartheta]$ and target span $[e_\gamma, e_\sigma]$,
- $\tilde{t}$ contains K nonterminals $\{X_k | 0 \le k \le K-1\}$,
- X_k covers source span $[f_{\varphi_k}, f_{\vartheta_k}]$ and target span $[e_{\gamma_k}, e_{\sigma_k}]$,

the MERS model used 5 kinds of source-side features as follows,

1. Lexical features: words around a rule (e.g. $f_{\varphi-1}$) and words covered by nonterminals in a rule (e.g. f_{φ_0}).

2. Part-of-speech features: part-of-speech (POS) of context words that are used as lexical features.

3. Span features: span lengths of source phrases covered by nonterminals in r.

4. Parent features: the parent node of $\tilde{t}$ in the parse tree of the source sentence.

5. Sibling features: the siblings of the root of $\tilde{t}$.

Note that the MERS model does not use features of the source subtree $\tilde{t}$, because the source subtree $\tilde{t}$ is fixed for each classifier.

The MERS model was integrated into the translation system as two additional features in Equation 2. Supposing that the derivation R contains M rules $r_1, ..., r_M$ with ambiguous source subtrees, then these two MERS features are as follows,

$$\begin{aligned} h_1(E,R,F) &= \sum_{m=1}^{M} \log \Pr\left(\tilde{e}_m|\tilde{t}_m, C_m\right) \\ h_2(E,R,F) &= M, \end{aligned} \tag{4}$$

where $\tilde{t}_m$ and $\tilde{e}_m$ are the source subtree and the target string contained in r_m, and C_m is the context of r_m. h_1 is the MERS probability feature, and, h_2 is a penalty feature counting the number of predictions made by the MERS model.

3 Our CSRS Approach

3.1 Modeling

The proposed CSRS model differs from the MERS model in three ways.

1. Instead of learning a single classifier for each source subtree $\tilde{t}$, it learns a single classifier for all rules.

2. Instead of hand-crafted features, it uses a feed-forward neural network to induce features from context words.

3. Instead of one-hot representations, it uses distributed representations to exploit similarities between words.

First, with regard to training, our CSRS model follows Zhang et al. (2015) in approximating the posterior probability by a binary classifier as follows,

$$\Pr\left(\tilde{e}|\tilde{t}, C\right) \approx \Pr\left(v = 1|\tilde{e}, \tilde{t}, C\right), \tag{5}$$

where $v \in \{0, 1\}$ is an indicator of whether $\tilde{t}$ is translated into $\tilde{e}$. This is in contrast to the MERS model, which treated the rule selection problem as a multi-class classification task. If instead we attempted to estimate output probabilities for all different $\tilde{e}$, the cost of estimating the normalization coefficient would be prohibitive, as the number of unique output-side word strings $\tilde{e}$ is large. There are a number of remedies to this, including noise contrastive estimation (Vaswani et al., 2013), but the binary approximation method has been reported to have better performance (Zhang et al., 2015).

To learn this model, we use a feed-forward neural network with structure similar to neural network language models (Vaswani et al., 2013). The input of the neural rule selection model is a vector representation for $\tilde{t}$, another vector representation for $\tilde{e}$, and a set of ξ vector representations for both source-side and target-side context words of r:

$$C(r) = w_1, ..., w_\xi \tag{6}$$

In our model, $C(r)$ is calculated differently depending on the number of nonterminals included in the rule. Specifically, Equation 7 defines $C_{out}(r, n)$ to be context words (n-grams) around r and $C_{in}(r, n, X_k)$ to be boundary words (n-grams) covered by nonterminal X_k in r.[1]

$$\begin{aligned} &C_{out}(r, n) \\ &= f_{\varphi-n}^{\varphi-1}, f_{\theta+1}^{\theta+n}, e_{\gamma-n}^{\gamma-1}, e_{\sigma+1}^{\sigma+n} \\ &C_{in}(r, n, X_k) \\ &= f_{\varphi_k}^{\varphi_k+n-1}, f_{\theta_k-(n-1)}^{\theta_k}, e_{\gamma_k}^{\gamma_k+n-1}, e_{\sigma_k-(n-1)}^{\sigma_k} \end{aligned} \tag{7}$$

The context words used for a translation rule r with K nonterminals are shown as below.

K	$C(r)$
$= 0$	$C_{out}(r, 6)$
$= 1$	$C_{out}(r, 4), C_{in}(r, 2, X_0)$
> 1	$C_{out}(r, 2), C_{in}(r, 2, X_0), C_{in}(r, 2, X_1)$

We can see that rules with different numbers of nonterminals K use different context words.[2]

[1] Note that when extracting C_{out}, we use "$\langle s \rangle$" and "$\langle /s \rangle$" for context words that exceed the length of the sentence; When extracting C_{in}, we use "$\langle non \rangle$" for context words that exceed the length of the nonterminal. Words that occur less than twice in the training data are replaced by "$\langle unk \rangle$".

[2] In most cases, restrictions on extracted rules will ensure that rules will only contain two nonterminals. However, when using minimal rules as described in the next section, more than two nonterminals are possible, and in these cases, only contextual information covered by the first two nonterminals is used in the input. These cases are sufficiently rare, however, that we chose to consider only the first two.

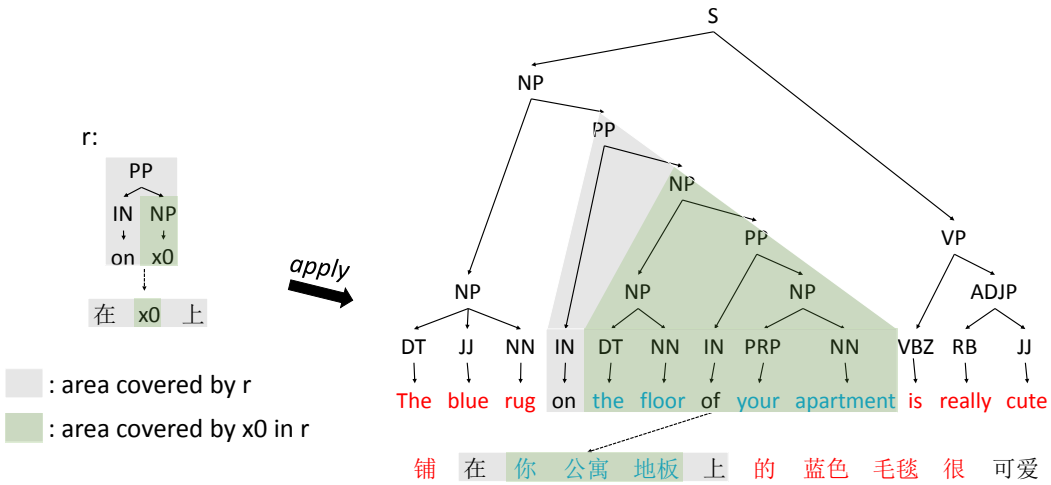


Figure 2: Context word examples. The red words are contained in $C_{out}(r, 4)$ and the blue words are contained in $C_{in}(r, 2, X_0)$.

For example, if r does not contain nonterminals, then C_{in} is not used. Besides, we use more context words surrounding the rule ($C_{out}(r, 6)$) for rules with $K = 0$ than rules that contain nonterminals ($C_{out}(r, 4)$ for $K = 1$ and $C_{out}(r, 2)$ for $K > 1$). This is based on the intuition that rules with $K = 0$ can only use the context words surrounding the rule as information for rule selection, hence this information is more important than for other rules. Figure 2 gives an example of context words when applying the rule r to the example sentence.

Note that we use target-side context because source-side context is not enough for selecting correct rules. Since it is not uncommon for one source sentence to have different correct translations, a translation rule used in one correct derivation may be incorrect for other derivations. In these cases, target-side context is useful for selecting appropriate translation rules. [3]

The vector representations for $\tilde{t}$, $\tilde{e}$ and C are obtained by using a projection matrix to project each one-hot input into a real-valued embedding vector. This projection is another key advantage over the MERS model. Because the CSRS model learns one unified model for all rules and can share all training data to learn better vector representations of words and rules, and the similarities between vectors can be used to generalize in cases such as the "thief/gunman" example in the introduction.

[3]It is also possible to consider target-side context in a framework like the MERS model, but we show in experiments that a linear model using the same features as the CSRS model did not improve accuracy.

After calculating the projections, two hidden layers are used to combine all inputs. Finally, the neural network has two outputs $\Pr(v = 1|\tilde{e}, \tilde{t}, C)$ and $\Pr(v = 0|\tilde{e}, \tilde{t}, C)$.

To train the CSRS model, we need both positive and negative training examples. Positive examples, $\langle \tilde{e}, \tilde{t}, C, 1 \rangle$, can be extracted directly from the parallel corpus. For each positive example, we generate one negative example, $\langle \tilde{e}', \tilde{t}, C, 0 \rangle$. Here, $\tilde{e}'$ is randomly generated according to the translation distribution (Zhang et al., 2015),

$$\Pr(\tilde{e}|\tilde{t}) = \frac{Count(\tilde{e}, \tilde{t})}{\sum_{\tilde{e}'} Count(\tilde{e}', \tilde{t})}, \quad (8)$$

where, $Count(\tilde{e}, \tilde{t})$ is how many times $\tilde{t}$ is translated into $\tilde{e}$ in the parallel corpus.

During translating, following the MERS model, the CSRS model only calculates probabilities for rules with ambiguous source subtrees. These predictions are converted into two CSRS features for the translation system similar to the two MERS features in Equation 4: one is the product of probabilities calculated by the CSRS model and the other one is a penalty feature that stands for how many rules with ambiguous source subtrees are contained in one translation.

3.2 Usage of Minimal Rules

Despite the fact that the CSRS model can share information among instances using distributed word representations, it still poses an extremely sparse learning problem. Specifically, the numbers of unique subtrees $\tilde{t}$ and strings $\tilde{e}$ are extremely large,

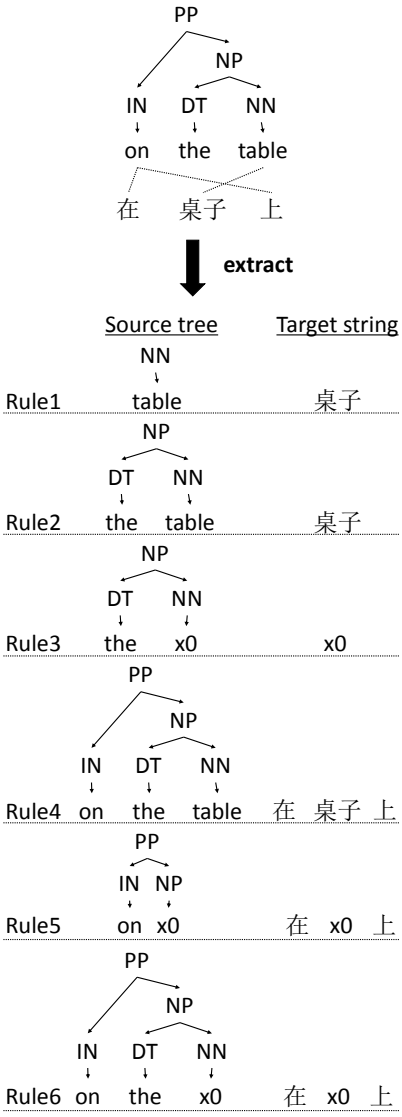


Figure 3: Rules.

and many may only appear a few times in the corpus. To reduce these problems of sparsity, we propose another improvement to the model, specifically through the use of minimal rules.

Minimal rules (Galley et al., 2004) are translation rules that cannot be split into two smaller rules. For example, in Figure 3, Rule2 is not a minimal rule, since Rule2 can be split into Rule1 and Rule3. In the same way, Rule4 and Rule6 are not minimal while Rule1, Rule3 and Rule5 are minimal.

Minimal rules are more frequent than non-minimal rules and have richer training data. Hence, we can expect that a rule selection model trained on minimal rules will suffer less from data sparsity problems. Besides, without non-minimal rules, the rule selection model will need less memory and can be trained faster.

To take advantage of this fact, we train another version of the CSRS model (CSRS-MINI) over only minimal rules. The probability of a non-minimal rule is then calculated using the product of the probability of minimal rules contained therein.

Note that for both the standard CSRS and CSRS-MINI models, we use the same baseline translation system which can use non-minimal translation rules. The CSRS-MINI model will break translation rules used in translations down into minimal rules and multiply all probabilities to calculate the necessary features.

4 Experiments

4.1 Setting

We evaluated the proposed approach for English-to-German (ED), English-to-French (EF), English-to-Chinese (EC) and English-to-Japanese (EJ) translation tasks. For the ED and EF tasks, the translation systems are trained on Europarl v7 parallel corpus and tested on the WMT 2015 translation task.[4] The test sets for the WMT 2014 translation task were used as development sets in our experiments. For the EC and EJ tasks, we used datasets provided for the patent machine translation task at NTCIR-9 (Goto et al., 2011).[5] The detailed statistics for training, development and test sets are given in Table 1. The word segmentation was done by BaseSeg (Zhao et al., 2006) for Chinese and Mecab[6] for Japanese.

For each translation task, we used Travatar (Neubig, 2013) to train a forest-to-string translation system. GIZA++ (Och and Ney, 2003) was used for word alignment. A 5-gram language model was trained on the target side of the training corpus using the IRST-LM Toolkit[7] with modified Kneser-Ney smoothing. Rule extraction was

[4]The WMT tasks provided other training corpora. We used only the Europarl corpus, because training a large-scale system on the whole data set requires large amounts of time and computational resources.

[5]Note that NTCIR-9 only contained a Chinese-to-English translation task. Because we want to test the proposed approach with a similarly accurate parsing model across our tasks, we used English as the source language in our experiments. In NTCIR-9, the development and test sets were both provided for the CE task while only the test set was provided for the EJ task. Therefore, we used the sentences from the NTCIR-8 EJ and JE test sets as the development set in our experiments.

[6]http://sourceforge.net/projects/mecab/files/

[7]http://hlt.fbk.eu/en/irstlm

			SOURCE	TARGET
ED	TRAIN	#Sents	1.90M	
		#Words	52.2M	49.7M
		#Vocab	113K	376K
	DEV	#Sents	3,003	
		#Words	67.6K	63.0K
	TEST	#Sents	2,169	
		#Words	46.8K	44.0K
EF	TRAIN	#Sents	1.99M	
		#Words	54.4M	60.4M
		#Vocab	114K	137K
	DEV	#Sents	3,003	
		#Words	71.1K	81.1K
	TEST	#Sents	1.5K	
		#Words	27.1K	29.8K
EC	TRAIN	#Sents	954K	
		#Words	40.4M	37.2M
		#Vocab	504K	288K
	DEV	#Sents	2K	
		#Words	77.5K	75.4K
	TEST	#Sents	2K	
		#Words	58.1K	55.5K
EJ	TRAIN	#Sents	3.14M	
		#Words	104M	118M
		#Vocab	273K	150K
	DEV	#Sents	2K	
		#Words	66.5K	74.6K
	TEST	#Sents	2K	
		#Words	70.6K	78.5K

Table 1: Data sets.

performed using the GHKM algorithm (Galley et al., 2006) and the maximum numbers of nonterminals and terminals contained in one rule were set to 2 and 10 respectively. Note that when extracting minimal rules, we release this limit. The decoding algorithm is the bottom-up forest-to-string decoding algorithm of Mi et al. (2008). For English parsing, we used Egret[8], which is able to output packed forests for decoding.

We trained the CSRS models (CSRS and CSRS-MINI) on translation rules extracted from the training set. Translation rules extracted from the development set were used as validation data for model training to avoid over-fitting. For different training epochs, we resample negative examples for each positive example to make use of different negative examples. The embedding dimension was set to be 50 and the number of hidden nodes was 100. The initial learning rate was set to be 0.1. The learning rate was halved each time the validation likelihood decreased. The number of epoches was set to be 20. A model was saved after each epoch and the model with highest validation likelihood was used in the translation system.

We implemented Liu et al. (2008)'s MERS model to compare with our approach. The training instances for their model were extracted from the training set. Following their work, the iteration number was set to be 100 and the Gaussian prior was set to be 1. We also compared the original MERS model and the MERS model trained only on minimal rules (MERS-MINI) to test the benefit of using minimal rules for model training.

[8]https://code.google.com/archive/p/egret-parser

	ED	EF	EC	EJ
Base	15.00	26.76	29.42	37.10
MERS	15.62	27.33	29.75	37.76
CSRS	16.15	28.05	30.12	37.83
MERS-MINI	15.77	28.13	30.53	38.14
CSRS-MINI	**16.49**	**28.30**	**31.63**	**38.32**

Table 2: Translation results. The bold numbers stand for the best systems.

	ED	EF	EC	EJ
CSRS vs. MERS	$>>$	$>>$	$>$	$-$
CSRS-MINI vs. MERS-MINI	$>>$	$-$	$>>$	$-$
MERS-MINI vs. MERS	$-$	$>>$	$>>$	$>>$
CSRS-MINI vs. CSRS	$>$	$-$	$>>$	$>>$

Table 3: Significance test results. The symbol $>>$ ($>$) represents a significant difference at the $p < 0.01$ ($p < 0.05$) level and the symbol - represents no significant difference at the $p < 0.05$ level.

The MERS and CSRS models were both used to calculate features used to rerank unique 1,000-best outputs of the baseline system. Tuning is performed to maximize BLEU score using minimum error rate training (Och, 2003).

4.2 Results

Table 2 shows the translation results and Table 3 shows significance test results using bootstrap resampling (Koehn, 2004): "Base" stands for the baseline system without any; "MERS", "CSRS", "MERS-MINI" and "CSRS-MINI" means the outputs of the baseline system were reranked using features from the MERS, CSRS, MERS-MINI and CSRS-MINI models respectively. Generally, the CSRS model outperformed the MERS model and the CSRS-MINI model outperformed the MERS-MINI model on different translation tasks. In addition, using minimal rules for model training benefitted both the MERS and CSRS models.

Table 4 shows translation examples in the EC task to demonstrate the reason why our approach improved accuracy. Among all translations, $T_{CSRS-MINI}$ is basically the same as the reference with only a few paraphrases that do not alter the meaning of the sentence. In con-

Source	typical dynamic response rate of an optical gap sensor as described above is approximately 2 khz , or 0.5 milliseconds .
Reference	上述(described above) 光学(optical) 间隙(gap) 传感器(sensor) 的 典型(typical) 动态(dynamic) 响应(response) 率(rate) 约(approximately) 为(is) 2KHz 或(or) 为 0.5 毫秒(milliseconds) 。
T_{Base}	典 型(typical) 的 动 态(dynamic) 响 应(response) 速 率(rate) 间隙(gap) 传感器(sensor) 的 光学(optical) 如上(above) 描述(described) 的 是(is) 约(approximately) 2 千赫(khz) 兹 ， 或(or) 0.5 毫秒(milliseconds) 。
T_{MERS}	典型(typical) 的 动态(dynamic) 响应(response) 率(rate) 光学(optical) 传感器(sensor) ， 如(as) 以上(above) 所 述(described) 间隙(gap) 的 约(approximately) 2 千赫(khz) ， 或(or) 0.5 毫秒(milliseconds) 。
T_{CSRS}	光学(optical) 传感器(sensor) ， 如(as) 以上(above) 所 述(described) 间隙(gap) 的 典型(typical) 的 动态(dynamic) 响应(response) 速率(rate) 为(is) 约(approximately) 2 千赫(khz) 兹 ， 或(or) 0.5 毫秒(milliseconds) 。
$T_{MERS-MINI}$	典 型(typical) 的 动 态(dynamic) 响 应(response) 率(rate) 间隙(gap) 传感器(sensor) 的 光学(optical) 如 上(above) 描 述(described) 的 是(is) 约(approximately) 2 千 赫(khz) 兹 ， 或(or) 0.5 毫秒(milliseconds) 。
$T_{CSRS-MINI}$	如 上(above) 描 述(described) 的 光学(optical) 间隙(gap) 传感器(sensor) 典 型(typical) 的 动 态(dynamic) 响应(response) 速率(rate) 为(is) 约(approximately) 2 千赫(khz) 兹 ， 或(or) 0.5 毫秒(milliseconds) 。

Table 4: Translation examples.

R_1: T_{MERS}&T_{CSRS}	PP (IN (“of”) NP (NP (DT (“an”) NP’ (JJ (“optical”) x0:NN)) x1:NP’)) → “光学(optical)” x1 x0 “的”
R_2: $T_{MERS-MINI}$	PP (IN (“of”) NP (NP (DT (“an”) NP’ (JJ (“optical”) x0:NP’)) x1:SBAR)) → x0 “的” “光学(optical)” x1 “的”
R_3 : $T_{CSRS-MINI}$	NP’ (JJ (“optical”) x0:NP’) → “光学(optical)” x0

Table 5: Rules used to translate the source word “optical” in different translations. Shadows (R_3) stand for ambiguous rules.

trast, T_{Base}, T_{MERS}, T_{CSRS} and $T_{MERS-MINI}$ all contain apparent mistakes. For example, the source phrase “optical gap sensor” (covered by gray shadows in Table 4) is wrongly translated in T_{Base}, T_{MERS}, T_{CSRS} and $T_{MERS-MINI}$ due to incorrect reorderings.

Table 5 shows rules used to translate the source word “optical” in different translations: R_1 is used in T_{MERS} and T_{CSRS}; R_2 is used in $T_{MERS-MINI}$; R_3 is used in $T_{CSRS-MINI}$. Although the source word “optical” is translated to the correct translation “光学(optical)” in all translations, R_1, R_2 and R_3 cause different reorderings for the source phrase “optical gap sensor”. R_3 reorders this source phrase correctly while R_1 and R_2 cause wrong reorderings for this source phrase.

We can see that R_1 is umambiguous, so the MERS and CSRS models will give probability 1 to R_1, which could make the MERS and CSRS models prefer T_{MERS} and T_{CSRS}. This is a typical translation error caused by sparse rules since the source subtree in R_1 does not have other translations in the training corpus.

To compare the MERS-MINI and CSRS-MINI models, Table 6 shows minimal rules (R_{2a}, R_{2b}, R_{3a} and R_{3b}) contained in R_2 and R_3. Table 7 shows probabilities of these minimal rules calculated by the MERS-MINI and CSRS-MINI models respectively. We can see that the CSRS-MINI model gave higher scores for the correct translation rules R_{3a} and R_{3b} than the MERS-MINI model, while the MERS-MINI model gave a higher score to the incorrect rule R_{2b} than the CSRS-MINI model.

Note that R_{2b} and R_{3b} are the same rule, but the target-side context in $T_{MERS-MINI}$ and $T_{CSRS-MINI}$ is different. The CSRS-MINI model will give R_{2b} and R_{3b} different scores because the CSRS-MINI model used target-side context. However, the MERS-MINI model only used source-side features and gave R_{2b} and R_{3b} the same score. The fact that the CSRS-MINI model

R_{2a}	PP (IN (“of”) NP (NP (DT (“an”) NP’ (x0:JJ x1:NP’)) x2:SBAR)) → x1 “的” x0 x2 “的”
R_{2b}	JJ (“optical”) → “光学(optical)”
R_{3a}	NP’ (x0:JJ x1:NP’) → x0 x1
R_{3b}	JJ (“optical”) → “光学(optical)”

Table 6: Minimal rules contained in R_2 and R_3. Shadows (R_{2b}, R_{3a} and R_{3b}) stand for ambiguous rules.

	MERS-MINI	CSRS-MINI
R_{2a}	1	1
R_{2b}	0.5441	0.09632
R_{3a}	0.9943	0.9987
R_{3b}	0.5441	0.7317

Table 7: Scores of minimal rules.

gave a higher score for R_{3b} than R_{2b} means that the CSRS-MINI model predicted the target string in R_{2b} and R_{3b} is a good translation in the context of $T_{CSRS-MINI}$ but not so good in the context of $T_{MERS-MINI}$. As we can see, the target phrase “如上(above) 描述(described) 的(of) 光学(optical) 间隙(gap) 传感器(sensor)” around “光学(optical)” in $T_{CSRS-MINI}$ is a reasonable Chinese phrase while the target phrase “间隙(gap) 传感器(sensor) 的(of) 光学(optical) 如上(above) 描述(described) 的(of)” around “光学(optical)” in $T_{MERS-MINI}$ does not make sense. Namely, the CSRS model trained with target-side context can perform rule selection considering target sentence fluency, which is the reason why target-side context can help in the rule selection task.

4.3 Analysis

To analyze the influence of different features, we trained the MERS model using source-side and target-side n-gram lexical features similar to the CSRS model. When using this feature set, the performance of the MERS model dropped significantly. This indicates that the syntactic, POS and span features used in the original MERS model are important for their model, since these features can generalize better. Purely lexical features are less effective due to sparsity problems when training one maximum entropy based classifier for each ambiguous source subtree and training data for each classifier is quite limited. In contrast, the CSRS model is trained in a continuous space and does not split training data, which relieves the sparsity problem of lexical features. As a result, the CSRS model achieved better performance using only lexical features compared to the MERS model. We also tried to use pre-trained word embedding features for the MERS model, but it did not improve the performance of the MERS model, which indicates that the log-linear model is not able to benefit from distributed representations as well as the neural network model.

We also tried reranking with both the CSRS and MERS models added as features, but it did not achieve further improvement compared to only using the CSRS model. This indicates that although these two models use different type of features, the information contained in these features are similar. For example, the POS features used in the MERS model and the distributed representations used in the CSRS model are both used for better generalization.

In addition, using both the CSRS and CSRS-MINI models did not improve over using only the CSRS-MINI model in our experiments. There are two main differences between the CSRS and CSRS-MINI models. First, minimal rules are more frequent and have more training data than non-minimal rules, which is why the CSRS-MINI model is more robust than the CSRS model. Second, non-minimal rules contain more information than minimal rules. For example, in Figure 3, Rule4 contains more information than Rule1, which could be an advantage for rule selection. However, the information contained in Rule4 will be considered as context features for Rule1. Therefore, this is no longer an advantage for the CSRS model as long as we use rich enough context features, which could be the reason why using both the CSRS and CSRS-MINI models cannot further improve the translation quality compared to using only the CSRS-MINI model.

5 Related Work

The rule selection problem for syntax-based SMT has received much attention. He et al. (2008) proposed a lexicalized rule selection model to perform context-sensitive rule selection for hierarchical phrase-base translation. Cui et al. (2010) introduced a joint rule selection model for hierarchical phrase-based translation, which also approximated the rule selection problem by a binary classification problem like our approach. However, these two models adopted linear classifiers similar to those used in the MERS model (Liu et al., 2008), which suffers more from the data sparsity

problem compared to the CSRS model.

There are also existing works that exploited neural networks to learn translation probabilities for translation rules used in the phrase-based translation model. Namely, these methods estimated translation probabilities for phrase pairs extracted from the parallel corpus. Schwenk (2012) proposed a continuous space translation model, which calculated the translation probability for each word in the target phrase and then multiplied the probabilities together as the translation probability of the phrase pair. Gao et al. (2014) and Zhang et al. (2014) proposed methods to learn continuous space phrase representations and use the similarity between the source and target phrases as translation probabilities for phrase pairs. All these three methods can only be used for the phrase-based translation model, not for syntax-based translation models.

There are also works that used minimal rules for modeling. Vaswani et al. (2011) proposed a rule Markov model using minimal rules for both training and decoding to achieve a slimmer model, a faster decoder and comparable performance with using non-minimal rules. Durrani et al. (2013) proposed a method to model with minimal translation units and decode with phrases for phrase-based SMT to improve translation performances. Both of these two methods do not use distributed representations as used in our model for better generalization.

In addition, neural machine translation (NMT) has shown promising results recently (Sutskever et al., 2014; Bahdanau et al., 2014; Luong et al., 2015a; Jean et al., 2015; Luong et al., 2015b). NMT uses a recurrent neural network to encode the whole source sentence and then produce the target words one by one. These models can be trained on parallel corpora and do not need word alignments to be learned in advance. There are also neural translation models that are trained on word-aligned parallel corpus (Devlin et al., 2014; Meng et al., 2015; Zhang et al., 2015; Setiawan et al., 2015), which use the alignment information to decide which parts of the source sentence are more important for predicting one particular target word. All these models are trained on plain source and target sentences without considering any syntactic information while our neural model learns rule selection for tree-based translation rules and makes use of the tree structure of natural language for better translation. There is also a new syntactic NMT model (Eriguchi et al., 2016), which extends the original sequence-to-sequence NMT model with the source-side phrase structure. Although this model takes source-side syntax into consideration, it still produces target words one by one as a sequence. In contrast, the tree-based translation rules used in our model can take advantage of the hierarchical structures of both source and target languages.

6 Conclusion

In this paper, we propose a CSRS model for syntax-based SMT, which is learned by a feed-forward neural network on a continuous space. Compared with the previous MERS model that used discrete representations of words as features, the CSRS model uses real-valued vector representations of words and can exploit similarity information between words for better generalization. In addition, we propose to use only minimal rules for rule selection to further relieve the data sparsity problem, since minimal rules are more frequent and have richer training data. In our experiments, the CSRS model outperformed the previous MERS model and the usage of minimal rules benefitted both CSRS and MERS models on different translation tasks.

For future work, we will explore more sophisticated features for the CSRS model, such as syntactic dependency relationships and head words, since only simple lexical features are used in the current incarnation.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Lei Cui, Dongdong Zhang, Mu Li, Ming Zhou, and Tiejun Zhao. 2010. A joint rule selection model for hierarchical phrase-based translation. In *Proc. ACL*, pages 6–11.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proc. ACL*, pages 1370–1380.

Nadir Durrani, Alexander Fraser, and Helmut Schmid. 2013. Model with minimal translation units, but decode with phrases. In *Proc. HLT-NAACL*, pages 1–11.

Akiko Eriguchi, Kazuma Hashimoto, and Yoshimasa Tsuruoka. 2016. Tree-to-sequence attentional neural machine translation. *arXiv preprint arXiv:1603.06075*.

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

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. COLING-ACL*, pages 961–968.

Jianfeng Gao, Xiaodong He, Wen-tau Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proc. ACL*, pages 699–709.

Isao Goto, Bin Lu, Ka Po Chow, Eiichiro Sumita, and Benjamin K Tsou. 2011. Overview of the patent machine translation task at the NTCIR-9 workshop. In *Proc. NTCIR-9*, pages 559–578.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *Proc. HLT-NAACL*, pages 105–112.

Zhongjun He, Qun Liu, and Shouxun Lin. 2008. Improving statistical machine translation using lexicalized rule selection. In *Proc. Coling*, pages 321–328.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proc. ACL-IJCNLP*, pages 1–10.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*, pages 388–395.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proc. COLING-ACL*, pages 609–616.

Qun Liu, Zhongjun He, Yang Liu, and Shouxun Lin. 2008. Maximum entropy based rule selection model for syntax-based statistical machine translation. In *Proc. EMNLP*, pages 89–97.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proc. EMNLP*, pages 1412–1421.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proc. ACL-IJCNLP*, pages 11–19.

Fandong Meng, Zhengdong Lu, Mingxuan Wang, Hang Li, Wenbin Jiang, and Qun Liu. 2015. Encoding source language with convolutional neural network for machine translation. In *Proc. ACL-IJCNLP*, pages 20–30.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proc. HLT-ACL*, pages 192–199.

Graham Neubig. 2013. Travatar: A forest-to-string machine translation engine based on tree transducers. In *Proc. ACL*, pages 91–96.

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

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*, pages 160–167.

Holger Schwenk. 2012. Continuous space translation models for phrase-based statistical machine translation. In *Proc. COLING*, pages 1071–1080.

Hendra Setiawan, Zhongqiang Huang, Jacob Devlin, Thomas Lamar, Rabih Zbib, Richard Schwartz, and John Makhoul. 2015. Statistical machine translation features with multitask tensor networks. In *Proc. ACL-IJCNLP*, pages 31–41.

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

Ashish Vaswani, Haitao Mi, Liang Huang, and David Chiang. 2011. Rule markov models for fast tree-to-string translation. In *Proc. HLT-ACL*, pages 856–864.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proc. EMNLP*, pages 1387–1392.

Jiajun Zhang, Shujie Liu, Mu Li, Ming Zhou, and Chengqing Zong. 2014. Bilingually-constrained phrase embeddings for machine translation. In *Proc. ACL*, pages 111–121.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2015. A binarized neural network joint model for machine translation. In *Proc. EMNLP*, pages 2094–2099.

Hai Zhao, Chang-Ning Huang, and Mu Li. 2006. An improved chinese word segmentation system with conditional random field. In *Proc. SIGHAN*, pages 162–165.

Probabilistic Graph-based Dependency Parsing with Convolutional Neural Network

Zhisong Zhang[1,2], **Hai Zhao**[1,2,*], **Lianhui Qin**[1,2]
[1]Department of Computer Science and Engineering,
Shanghai Jiao Tong University, Shanghai, 200240, China
[2]Key Laboratory of Shanghai Education Commission for Intelligent Interaction
and Cognitive Engineering, Shanghai Jiao Tong University, Shanghai, 200240, China
{zzs2011,qinlianhui}@sjtu.edu.cn,zhaohai@cs.sjtu.edu.cn

Abstract

This paper presents neural probabilistic parsing models which explore up to third-order graph-based parsing with maximum likelihood training criteria. Two neural network extensions are exploited for performance improvement. Firstly, a convolutional layer that absorbs the influences of all words in a sentence is used so that sentence-level information can be effectively captured. Secondly, a linear layer is added to integrate different order neural models and trained with perceptron method. The proposed parsers are evaluated on English and Chinese Penn Treebanks and obtain competitive accuracies.

1 Introduction

Neural network methods have shown great promise in the field of parsing and other related natural language processing tasks, exploiting more complex features with distributed representation and non-linear neural network (Wang et al., 2013; Wang et al., 2014; Cai and Zhao, 2016; Wang et al., 2016). In transition-based dependency parsing, neural models that can represent the partial or whole parsing histories have been explored (Weiss et al., 2015; Dyer et al., 2015). While for graph-based parsing, on which we focus in this work, Pei et al. (2015) also show the effectiveness of neural methods.

*Corresponding author. This paper was partially supported by Cai Yuanpei Program (CSC No. 201304490199 and No. 201304490171), National Natural Science Foundation of China (No. 61170114 and No. 61272248), National Basic Research Program of China (No. 2013CB329401), Major Basic Research Program of Shanghai Science and Technology Committee (No. 15JC1400103), Art and Science Interdisciplinary Funds of Shanghai Jiao Tong University (No. 14JCRZ04), and Key Project of National Society Science Foundation of China (No. 15-ZDA041).

The graph-based parser generally consists of two components: one is the parsing algorithm for inference or searching the most likely parse tree, the other is the parameter estimation approach for the machine learning models. For the former, classical dynamic programming algorithms are usually adopted, while for the latter, there are various solutions. Like some previous neural methods (Socher et al., 2010; Socher et al., 2013), to tackle the structure prediction problems, Pei et al. (2015) utilize a max-margin training criterion, which does not include probabilistic explanations. Re-visiting the traditional probabilistic criteria in log-linear models, this work utilizes maximum likelihood for neural network training. Durrett and Klein (2015) adopt this method for constituency parsing, which scores the anchored rules with neural models and formalizes the probabilities with tree-structured random fields. Motivated by this work, we utilize the probabilistic treatment for dependency parsing: scoring the edges or high-order sub-trees with a neural model and calculating the gradients according to probabilistic criteria. Although scores are computed by a neural network, the existing dynamic programming algorithms for gradient calculation remain the same as those in log-linear models.

Graph-based methods search globally through the whole space for trees and get the highest-scored one, however, the scores for the sub-trees are usually locally decided, considering only surrounding words within a limited-sized window. Convolutional neural network (CNN) provides a natural way to model a whole sentence. By introducing a distance-aware convolutional layer, sentence-level representation can be exploited for parsing. We will especially verify the effectiveness of such representation incorporated with window-based representation.

Graph-based parsing has a natural extension

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1382–1392,
Berlin, Germany, August 7-12, 2016.

through raising its order and higher-order parsers usually perform better. In previous work on high-order graph-parsing, the scores of high-order sub-trees usually include the lower-order parts in their high-order factorizations. In traditional linear models, combining scores can be implemented by including low-order features. However, for neural models, this is not that straightforward because of nonlinearity. A straightforward strategy is simply adding up all the scores, which in fact works well; another way is stacking a linear layer on the top of the representation from various already-trained neural parsing models of different orders.

This paper presents neural probabilistic models for graph-based projective dependency parsing, and explores up to third-order models. Here are the three highlights of the proposed methods:

- Probabilistic criteria for neural network training. (Section 2.2)
- Sentence-level representation learned from a convolutional layer. (Section 3.2)
- Ensemble models with a stacked linear output layer. (Section 3.3)

Our main contribution is exploring sub-tree scoring models which combine local features with a window-based neural network and global features from a distance-aware convolutional neural network. A free distribution of our implementation is publicly available[1].

The remainder of the paper is organized as follows: Section 2 explains the probabilistic model for graph-based parsing, Section 3 describes our neural network models, Section 4 presents our experiments and Section 5 discusses related work, we summarize this paper in Section 6.

2 Probabilistic Graph-based Dependency Parsing

2.1 Graph-based Dependency Parsing

Dependency parsing aims to predict a dependency tree, in which all the edges connect head-modifier pairs. In graph-based methods, a dependency tree is factored into sub-trees, from single edge to multiple edges with different patterns; we will call these specified sub-trees *factors* in this paper. According to the sub-tree size of the factors, we can define the order of the graph model. Three different ordered factorizations considered in this work and their sub-tree patterns are shown in Figure 1.

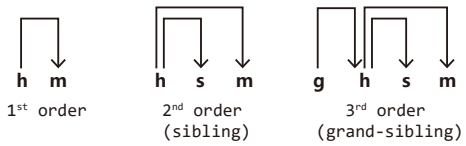


Figure 1: The decompositions of factors.

The score for a dependency tree (T) is defined as the sum of the scores of all its factors (p):

$$Score(T) = \sum_{p \in T} Score(p)$$

In this way, the dependency parsing task is to find a max-scoring tree. For projective dependency parsing considered in this work, this searching problem is conquered by dynamic programming algorithms with the key assumption that the factors are scored independently. Previous work (Eisner, 1996; McDonald et al., 2005; McDonald and Pereira, 2006; Koo and Collins, 2010; Ma and Zhao, 2012) explores ingenious algorithms for decoding ranging from first-order to higher-orders. Our proposed parsers also take these algorithms as backbones and use them for inference.

2.2 Probabilistic Model

With the graph factorization and inference, the remaining problems are how to obtain the scores and how to train the scoring model. For the scoring models, traditional linear methods utilize manually specified features and linear scoring models, while we adopt neural network models, which may exploit better feature representations.

For the training methods, in recent neural graph-based parsers, non-probabilistic margin-based methods are usually used. However, following the maximum likelihood criteria in traditional log-linear models, we can treat it in a probabilistic way. In fact, the probabilistic treatment still utilizes the scores of sub-tree factors in graph models. As in log-linear models like Conditional Random Field (CRF) (Lafferty et al., 2001), the exponentials of scores are taken before re-normalizing, and the probability distribution over trees condi-

[1] https://github.com/zzsfornlp/nnpgdparser

tioned on a sentence X is defined as follows:

$$Pr(T|X,\theta) = \frac{1}{Z(X)}\exp(Score(T|\theta))$$
$$Z(X) = \sum_{T'}\exp(Score(T'|\theta))$$

where θ represents the parameters and $Z(X)$ is the re-normalization partition function. The intuition is that the higher the score is, the more potential or mass it will get, leading to higher probability.

The training criteria will be log-likelihood in the classical setting of maximum likelihood estimation, and we define the loss for a parse tree as negative log-likelihood:

$$\begin{aligned} L(\theta) &= -\log Pr(T_g|X,\theta) \\ &= -Score(T_g|\theta) + \log(Z(X)) \end{aligned}$$

where T_g stands for the golden parse tree. Now we need to calculate the gradients of θ according to gradient-based optimization. Focusing on the second term, we have (some conditions are left out for simplicity):

$$\begin{aligned} \frac{\partial \log(Z(X))}{\partial\theta} &= \sum_{T'} Pr(T') \sum_{p\in T'} \frac{\partial Score(p)}{\partial\theta} \\ &= \sum_{p} \frac{\partial Score(p)}{\partial\theta} \sum_{T'\in T(p)} Pr(T') \end{aligned}$$

Here, $T(p)$ is the set of trees that contain the factor p, and the inner summation is defined as the marginal probability $m(p)$:

$$m(p) = \sum_{T'\in T(p)} Pr(T')$$

which can be viewed as the mass of all the trees containing the specified factor p. The calculation of $m(p)$ (Paskin, 2001; Ma and Zhao, 2015) is solved by a variant of inside-outside algorithm, which is of the same complexity compared with the corresponding inference algorithms. Finally, the gradients can be represented as:

$$\frac{\partial L(\theta)}{\partial\theta} = \sum_{p} \frac{\partial Score(p)}{\partial\theta}\Big(-\big[p\in T_g\big] + m(p)\Big)$$

where $[p \in T_g]$ is a binary value which indicates whether p is in tree T_g.

Traditional models usually utilize linear functions for the *Score* function, which might need carefully feature engineering such as (Zhao et al., 2009a; Zhao et al., 2009b; Zhao et al., 2009c; Zhao, 2009; Zhao et al., 2013), while we adopt neural models with the probabilistic training criteria unchanged.

2.3 Training Criteria

We take a further look between the maximum-likelihood criteria and the max-margin criteria. For the max-margin method, the loss is the difference between the scores of the golden tree and a predicted tree, and its sub-gradient can be written in a similar form:

$$\frac{\partial L_m(\theta)}{\partial\theta} = \sum_{p} \frac{\partial Score(p)}{\partial\theta}\Big(-\big[p\in T_g\big] + \big[p\in T_b\big]\Big)$$

Here, the predicted tree T_b is the best-scored tree with a structured margin loss in the score.

Comparing the derivatives, we can see that the one of probabilistic criteria can be viewed as a soft version of the max-margin criteria, and all the possible factors are considered when calculating gradients for the probabilistic way, while only wrongly predicted factors have non-zero sub-gradients for max-margin training. This observation is not new and Gimpel and Smith (2010) provide a good review of several training criteria. It might be interesting to explore the impacts of different training criteria on the parsing performance, and we will leave it for future research.

2.4 Labeled Parsing

In a dependency tree, each edge can be given a label indicating the type of the dependency relation, this labeling procedure can be integrated directly into the parsing task, instead of a second pass after obtaining the structure.

For the probabilistic model, integrating labeled parsing only needs some extensions for the inference procedure and marginal probability calculations. For the simplicity, we only consider a single label for each factor (even for high-order ones) which corresponds to Model 1 in (Ma and Hovy, 2015): the label of the edge between head and modifier word, which will only multiply $O(l)$ to the complexity. We find this direct approach not only achieves labeled parsing in one pass, but also improves unlabeled attachment accuracies (see Section 4.3), which may benefit from the joint learning with the labels.

3 Neural Model

The task for the neural models is computing the labeled scores of the factors. The inputs are the words in a factor with contexts, and the outputs are the scores for this factor to be valid in the dependency tree. We propose neural models to in-

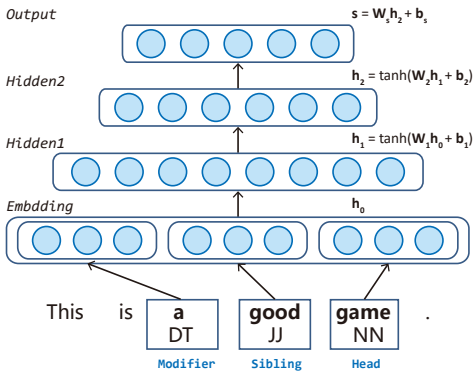


Figure 2: The architecture for the basic model (second order parsing).

tegrate features from both local word-neighboring windows and the entire sentence, and furthermore explore ensemble models with different orders.

3.1 Basic Local Model

Architecture The basic model uses a window-based approach, which includes only surrounding words for the contexts. Figure 2 illustrates a second-order sibling model and models of other orders adopt similar structures. It is simply a standard feed-forward neural network with two hidden layers ($\mathbf{h}_1$ and $\mathbf{h}_2$) above the embedding layer ($\mathbf{h}_0$), the hidden layers all adopt $tanh$ activation function, and the output layer (noted as $\mathbf{s}$) directly represents the scores for different labels.

Feature Sets All the features representing the input factor are atomic and projected to embeddings, then the embedding layer is formed by concatenating them. There are three categories of features: word forms, POS (part-of-speech) tags and distances. For each node in the factor, word forms and POS tags of the surrounding words in a specified window are also considered. Special tokens for start or end of sentences, root node and unknown words are added for both word forms and POS tags. Distances can be negative or positive to represent the relative positions between the factor nodes in surface string. Take the situation for the second-order model as an example, there are three nodes in a factor: h for head, m for modifier and s for sibling. When considering three-word windows, there will be three word forms and three tags for each node and its surrounding context. m and s both have one distance feature while h does not have one as its parent does not exist in the factor.

Training As stated in Section 2.2, we use the maximum likelihood criteria. Moreover, we add two L2-regularizations: one is for all the weights θ' (biases and embeddings not included) to avoid over-fitting and another is for preventing the final output scores from growing too large. The former is common practice for neural network, while the latter is to set soft limits for the norms of the scores. Although the second term is not usually adopted, it directly puts soft constraints on the scores and improves the accuracies (about 0.1% for UAS/LAS overall) according to our primary experiments. So the final loss function will be:

$$L'(\theta) = \sum_p \Big(Score(p) \cdot \big(- \big[p \in T_g \big] + m(p) \big) + \lambda_s \cdot Score(p)^2 \Big) + \lambda_m \cdot \|\theta'\|^2$$

where λ_m and λ_s respectively represent regularization parameters for model and scores. The training process utilizes a mini-batched stochastic gradient descent method with momentum.

Comparisons Our basic model resembles the one of Pei et al. (2015), but with some major differences: probabilistic training criteria are adopted, the structures of the proposed networks are different and direction information is encoded in distance features. Moreover, they simply average embeddings in specified regions for phrase-embedding, while we will include sentence-embedding in convolutional model as follows.

3.2 Convolutional Model

To encode sentence-level information and obtain sentence embeddings, a convolutional layer of the whole sentence followed by a max-pooling layer is adopted. However, we intend to score a factor in a sentence and the position of the nodes should also be encoded. The scheme is to use the distance embedding for the whole convolution window as the position feature.

We will take the second-order model as an example to introduce the related operations. Figure 3 shows the convolution operation for a convolution window, the input atomic features are the word forms and POS tags for each word inside the window, and the distances of only the center word (assuming an odd-sized window) to specified nodes in the factor are adopted as position features. In the example, "*game-good-a*" is to be scored as a second-order sibling factor, and for a

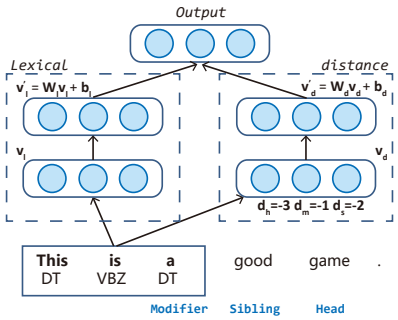


Figure 3: The operations for one convolution window (second order parsing).

convolution window of "*This is a*", word forms and corresponding POS tags are projected to embeddings and concatenated as the lexical vector $\mathbf{v}_l$, the distances of the center word "*is*" to all the three nodes in the factor are also projected to embeddings and concatenated as the distance vector $\mathbf{v}_d$, then these two vectors go through difference linear transformations into the same dimension and are combined together through element-wise addition or multiplication.

In general, assuming after the projection layer, embeddings of the word forms and POS tags of the sentence are represented as $[\mathbf{w}_0, \mathbf{w}_1, ..., \mathbf{w}_{n-1}]$ and $[\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_{n-1}]$. Those embeddings in the basic model may be reused here by sharing the embedding look-up table. The second-order sibling factor to be scored has nodes with indexes of m (modifier), h (head) and s (sibling). The distance embeddings are denoted by $\mathbf{d}$, which can be either negative or positive. These distance embeddings are different from the ones in the basic model, because here we measure the distances between the convolution window (its center word) and factor nodes, while the distances between nodes inside the factors are measured in the basic model.

For a specified window $[i : j]$, always assuming an odd number sized window, and the center token is indexed to $c = \frac{i+j}{2}$, the $\mathbf{v}_l$ and $\mathbf{v}_d$ are obtained through simple concatenation:

$$\mathbf{v}_l = [\mathbf{w}_i, \mathbf{p}_i, \mathbf{w}_{i+1}, \mathbf{p}_{i+1}, ..., \mathbf{w}_j, \mathbf{p}_j]$$
$$\mathbf{v}_d = [\mathbf{d}_{c-h}, \mathbf{d}_{c-m}, \mathbf{d}_{c-s}]$$

then $\mathbf{v}_l$ and $\mathbf{v}_d$ go through difference linear transformations into same dimension space: $\mathbf{v'}_l, \mathbf{v'}_d \in \mathbb{R}^n$, where n is also the dimension of the output vector $\mathbf{v}_o$ for the window. The linear operations can be expressed as:

$$\mathbf{v'}_l = \mathbf{W}_l \cdot \mathbf{v}_l + \mathbf{b}_l$$
$$\mathbf{v'}_d = \mathbf{W}_d \cdot \mathbf{v}_d + \mathbf{b}_d$$

The final vector $\mathbf{v}_o$ is obtained by element-wise operations of $\mathbf{v'}_l$ and $\mathbf{v'}_d$. We consider two strategies: (1) add: simple element-wise addition, (2) mul: element-wise multiplication with $\mathbf{v'}_d$ activated by $tanh$. They can be formalized as:

$$\mathbf{v}_{o\text{-}add} = \mathbf{v'}_l \oplus \mathbf{v'}_d$$
$$\mathbf{v}_{o\text{-}mul} = \mathbf{v'}_l \odot tanh(\mathbf{v'}_d)$$

All the windows whose center-located word is valid (exists) in the sentence are considered and we will get a sequence of convolution outputs whose number is the same as the sentence length. The convolution outputs (all $\mathbf{v}_o$) are collapsed into one global vector $\mathbf{v}_g$ using a standard max-pooling operation. Finally, for utilizing the sentence-level representation in the basic model, we can either replace the original first hidden layer $\mathbf{h}_1$ with $\mathbf{v}_g$ or concatenate $\mathbf{v}_g$ to $\mathbf{h}_1$ for combining local and global features.

3.3 Ensemble Models

For higher-order dependency parsing, it is a standard practice to include the impact of lower-order parts in the scoring of higher-order factors, which actually is an ensemble method of different order models for scoring.

A simple adding scheme is often used. For nonlinear neural models, we use an explicit adding method. For example, in third-order parsing, the final score for the factor (g, h, m, s) will be:

$$\mathbf{s}_{add}(g, h, m, s) = \mathbf{s}_{o3}(g, h, m, s) + \mathbf{s}_{o2}(h, m, s) + \mathbf{s}_{o1}(h, m)$$

Here, g, h, m and s represent the grandparent, head, modifier and sibling nodes in the grand-sibling third-order factor; $\mathbf{s}_{o1}$, $\mathbf{s}_{o2}$ and $\mathbf{s}_{o3}$ stand for the corresponding lower-order scores from first, second and third order models, respectively.

We notice that ensemble or stacking methods for dependency parsing have explored in previous work (Nivre and McDonald, 2008; Torres Martins et al., 2008). Recently, Weiss et al. (2015) stack a linear layer for the final scoring in a single model, and we extend this method to combine multiple models by stacking a linear layer on their output and hidden layers. The simple adding scheme can

be viewed as adopting a final layer with specially fixed weights.

For each model to be combined, we concatenate the output layer and all hidden layers (except embedding layer $\mathbf{h}_0$):

$$\mathbf{v}_{all} = [\mathbf{s}, \mathbf{h}_1, \mathbf{h}_2]$$

All $\mathbf{v}_{all}$ from different models are again concatenated to form the input for the final linear layer and the final scores are obtained through a linear transformation (no bias adding):

$$\mathbf{v}_{combine} = [\mathbf{v}_{all\text{-}o1}, \mathbf{v}_{all\text{-}o2}, \mathbf{v}_{all\text{-}o3}]$$
$$\mathbf{s}_{combine} = \mathbf{W}_{combine} \cdot \mathbf{v}_{combine}$$

We no longer update weights for the underlying neural models, and the learning of the final layer is equally training a linear model, for which structured average perceptron (Collins, 2002; Collins and Roark, 2004) is adopted for simplicity.

This ensemble scheme can be extended in several ways which might be explored in future work: (1) feed-forward network can be stacked rather than a single linear layer, (2) traditional sparse features can also be concatenated to $\mathbf{v}_{combine}$ to combine manually specified representations with distributed neural representations as in (Zhang and Zhang, 2015).

4 Experiments

The proposed parsers are evaluated on English Penn Treebank (PTB) and Chinese Penn Treebank (CTB). Unlabeled attachment scores (UAS), labeled attachment scores (LAS) and unlabeled complete matches (CM) are the metrics. Punctuations[2] are ignored as in previous work (Koo and Collins, 2010; Zhang and Clark, 2008).

For English, we follow the splitting convention for *PTB3*: sections 2-21 for training, 22 for developing and 23 for test. We prepare three datasets of PTB, using different conversion tools: (1) Penn2Malt[3] and the head rules of Yamada and Matsumoto (2003), noted as *PTB-Y&M*; (2) dependency converter in Stanford parser v3.3.0 with Stanford Basic Dependencies (De Marneffe et al., 2006), noted as *PTB-SD*; (3) LTH Constituent-to-Dependency Conversion Tool[4] (Johansson and Nugues, 2007), noted as *PTB-LTH*. We use Stanford POS tagger (Toutanova et al., 2003) to get predicted POS tags for development and test sets, and the accuracies for their tags are 97.2% and 97.4%, respectively.

For Chinese, we adopt the splitting convention for *CTB5* described in (Zhang and Clark, 2008). The dependencies (noted as *CTB*), are converted with the Penn2Malt converter. Gold segmentation and POS tags are used as in previous work.

4.1 Settings

Settings of our models will be described in this sub-section, including pre-processing and initializations, hyper-parameters, and training details.

We ignore the words that occur less than 3 times in the training treebank and use a special token to replace them. For English parsing, we initialize word embeddings with word vectors trained on Wikipedia using *word2vec* (Mikolov et al., 2013); all other weights and biases are initialized randomly with uniform distribution.

For the structures of neural models, all the embeddings (word, POS and distances) have dimensions of 50. For basic local models, $\mathbf{h}_1$ and $\mathbf{h}_2$ are set to 200 and 100, and the local window size is set to 7. For convolutional models, a three-word-sized window for convolution is specified, and convolution output dimension (number of filters) is 100. When concatenating the convolution vector (after pooling) to $\mathbf{h}_1$, it will make the first hidden layer's dimension 300.

For the training of neural network, we set the initial learning rate to 0.1 and the momentum to 0.6. After each iteration, the parser is tested on the development set and if the accuracy decreases, the learning rate will be halved. The learning rate will also be halved if no decreases of the accuracy for three epochs. We train the neural models for 12 epochs and select the one that performs best on the development set. The regularization parameters λ_m and λ_s are set to 0.0001 and 0.001. For the perceptron training of the ensemble model, only one epoch is enough based on the results of the development set.

The runtime of the model is influenced by the hyper-parameter setting. According to our experiments, using dual-core on 3.0 GHz i7 CPU, the training costs 6 to 15 hours for different-order models and the testing is comparably efficient as recent neural graph-parsers. The calculation of the

[2]Tokens whose gold POS tags are one of {“ ” : , .} for PTB or *PU* for CTB.

[3]http://stp.lingfil.uu.se/~nivre/research/Penn2Malt.html

[4]http://nlp.cs.lth.se/software/treebank_converter

Method	UAS	LAS	CM
Basic (first-order)			
Unlabeled	91.53	–	42.82
Labeled	92.13	89.60	45.06
Labeled+pre-training	92.19	89.73	45.18
Convolutional (first-order)			
replace-add	92.26	89.83	44.76
replace-mul	92.02	89.61	44.24
concatenate-add	92.63	90.20	46.18
concatenate-mul	92.33	89.83	44.94
Higher-orders			
o2-nope	92.85	90.51	49.65
o2-adding	93.47	91.13	51.41
o2-perceptron	93.63	91.39	51.53
o3-nope	92.47	90.01	49.06
o3-adding	93.70	91.37	53.53
o3-perceptron	93.51	91.20	51.76

Table 1: Effects of the components, on *PTB-SD* development set.

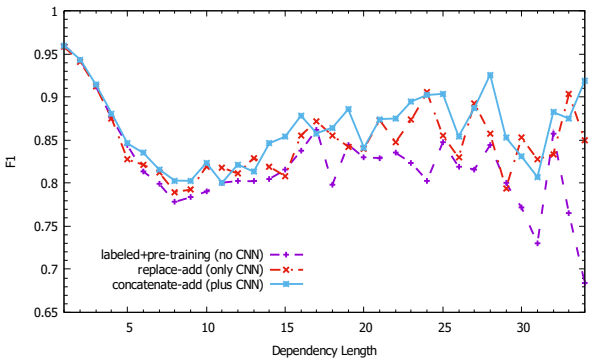


Figure 4: F1 measure of different dependency lengths, on *PTB-SD* development set.

convolution model approximately takes up 40% of all computations. The convolution operation indeed costs more, but the lexical parts $\mathbf{v}'_l$ of the convolution do not concern the factors and are computed only once for one sentence, which makes it less computationally expensive.

4.2 Pruning

For high-order parsing, the computation cost rises in proportion to the length of the sentence, and it will be too expensive to calculate scores for all the factors. Fortunately, many edges are quite unlikely to be valid and can be pruned away using low-order models. We follow the method of Koo and Collins (2010) and directly use the first-order probabilistic neural parser for pruning. We compute the marginal probability $m(h, m)$ for each edge and prune away the edges whose marginal probability is below $\epsilon \times \max_{h'} m(h', m)$. ϵ means the pruning threshold that is set to 0.0001 for second-order. For third-order parsing, considering the computational cost, we set it to 0.001.

4.3 Model Analysis

This section presents experiments to verify the effectiveness of the proposed methods and only the *PTB-SD* development set will be used in these experiments, which fall into three groups concerning basic models, convolutional models and ensemble ones, as shown in Table 1.

The first group focuses on the basic local models of first order. The first two, *Unlabeled* and *Labeled*, do not use pre-training vectors for initialization, while the third, *Labeled+pre-training*, utilizes them. The *Unlabeled* does not utilize the labels in training set and its model only gives one dependency score (we do not train a second stage labeling model, so the LAS of the unlabeled one is not available) and the *Labeled* directly predicts the scores for all labels. We can see that labeled parsing not only demonstrates the convenience of outputting dependency relations and labels for once, but also obtains better parsing performances. Also, we observe that pre-trained word vectors bring slight improvements. Pre-trained initialization and labeled parsing will be adopted for the next two groups and the rest experiments.

Next, we explore the effectiveness of the CNN enhancement. In the four entries of this group, *concatenate* or *replace* means whether to concatenate the sentence-level vector $\mathbf{v}_g$ to the first hidden layer $\mathbf{h}_1$ or just replace it (just throw away the representation from basic models), *add* or *mul* means to use which way for attaching distance information. Surprisingly, simple adding method surpasses the more complex multiplication-with-activation method, which might indicate that the direct activation operation may not be suitable for encoding distance information. With no surprises, the concatenating method works better because it combines both the local window-based and global sentence-level information. We also explore the influences of the convolution operations on dependencies of different lengths, as shown in Figure 4, the convolutional methods help the decisions of long-range dependencies generally. For the high-order parsing in the rest of this paper, we will all adopt the *concatenate-add* setting.

In the third group, we can see that high-order parsing brings significant performance improvement. For high-order parsing, three ensemble schemes are examined: no combination, adding

	PTB-Y&M			*PTB-SD*			*PTB-LTH*			*CTB*		
Methods	UAS	LAS	CM	UAS	LAS	CM	UAS	LAS	CM	UAS	LAS	CM
Graph-NN:proposed												
o3-adding	93.20	92.12	48.92	93.42	91.29	50.37	**93.14**	90.07	43.38	87.55	**86.19**	35.65
o3-perceptron	**93.31**	92.23	50.00	93.42	91.26	49.92	93.12	89.53	43.83	**87.65**	86.17	36.07
Graph-NN:others												
Pei et al. (2015)	93.29	92.13	–	–	–	–	–	–	–	–	–	–
Fonseca and Aluísio (2015)	–	–	–	–	–	–	91.6–	88.9–	–	–	–	–
Zhang and Zhao (2015)	–	–	–	–	–	–	92.52	–	41.10	86.01	–	31.88
Graph-Linear												
Koo and Collins (2010)	93.04	–	–	–	–	–	–	–	–	–	–	–
Martins et al. (2013)	93.07	–	–	92.82	–	–	–	–	–	–	–	–
Ma and Zhao (2015)	93.0–	–	48.8–	–	–	–	–	–	–	87.2–	–	37.0–
Transition-NN												
Chen and Manning (2014)	–	–	–	91.8–	89.6–	–	92.0–	**90.7–**	–	83.9–	82.4–	–
Dyer et al. (2015)	–	–	–	93.1–	90.9–	–	–	–	–	87.2–	85.7–	–
Weiss et al. (2015)	–	–	–	**93.99**	**92.05**	–	–	–	–	–	–	–
Zhou et al. (2015)	93.28	**92.35**	–	–	–	–	–	–	–	–	–	–

Table 2: Comparisons of results on the test sets.

and stacking another linear perceptron layer (with the suffixes of *-nope*, *-adding* and *-perceptron* respectively). The results show that model ensemble improves the accuracies quite a few. For third-order parsing, the no-combination method performs quite poorly compared to the others, which may be caused by the relative strict setting of the pruning threshold. Nevertheless, with model ensemble, the third-order models perform better than the second-order ones. Though the perceptron strategy does not work well for third-order parsing in this dataset, it is still more general than the simple adding method, since the latter can be seen as a special parameter setting of the former.

4.4 Results

We show the results of two of the best proposed parsers: third-order adding (*o3-adding*) and third-order perceptron (*o3-perceptron*) methods, and compare with the reported results of some previous work in Table 2. We compare with three categories of models: other *Graph*-based *NN* (neural network) models, traditional *Graph*-based *Linear* models and *Transition*-based *NN* models. For PTB, there have been several different dependency converters which lead to different sets of dependencies and we choose three of the most popular ones for more comprehensive comparisons. Since not all work report results on all of these dependencies, some of the entries might be not available.

From the comparison, we see that the proposed parser has output competitive performance for different dependency conversion conventions and treebanks. Compared with traditional graph-based linear models, neural models may benefit from better feature representations and more general non-linear transformations.

The results and comparisons in Table 2 demonstrate the proposed models can obtain comparable accuracies, which show the effectiveness of combining local and global features through window-based and convolutional neural networks.

5 Related Work

CNN has been explored in recent work of relation classification (Zeng et al., 2014; Chen et al., 2015), which resembles the task of deciding dependency relations in parsing. However, relation classification usually involves labeling for given arguments and seldom needs to consider the global structure. Parsing is more complex for it needs to predict structures and the use of CNN should be incorporated with the searching algorithms.

Neural network methods have been proved effective for graph-based parsing. Lei et al. (2014) explore a tensor scoring method, however, it needs to combine scores from linear models and we are not able to compare with it because of different datasets (they take datasets from CoNLL shared task). Zhang and Zhao (2015) also explore a probabilistic treatment, but its model may give mass to illegal trees or non-trees. Fonseca and Aluísio (2015) utilize CNN for scoring edges, though only explore first-order parsing. Its model is based on head selection for each modifier and might be difficult to be extended to high-order parsing. Recently, several neural re-ranking models, like Inside-Outside Recursive Neural Network

(Le and Zuidema, 2014) and Recursive CNN (Zhu et al., 2015), are utilized for capturing features with more contexts. However, re-ranking models depend on the underlying base parsers, which might already miss the correct trees. Generally, the re-ranking techniques play a role of additional enhancement for basic parsing models, and therefore they are not included in our comparisons.

The conditional log-likelihood probabilistic criterion utilized in this work is actually a (conditioned) Markov Random Field for tree structures, and it has been applied to parsing since long time ago. Johnson et al. (1999) utilize the Markov Random Fields for stochastic grammars and gradient based methods are adopted for parameter estimations, and Geman and Johnson (2002) extend this with dynamic programming algorithms for inference and marginal-probability calculation. Collins (2000) uses the same probabilistic treatment for re-ranking and the denominator only includes the candidate trees which can be seen as an approximation for the whole space of trees. Finkel et al. (2008) utilize it for feature-based parsing. The probabilistic training criterion for linear graph-based dependency models have been also explored in (Li et al., 2014; Ma and Zhao, 2015). However, these previous methods usually exploit log-linear models utilizing sparse features for input representations and linear models for score calculations, which are replaced by more sophisticated distributed representations and neural models, as shown in this work.

6 Conclusions

This work presents neural probabilistic graph-based models for dependency parsing, together with a convolutional part which could capture the sentence-level information. With distributed vectors for representations and complex non-linear neural network for calculations, the model can effectively capture more complex features when deciding the scores for sub-tree factors and experiments on standard treebanks show that the proposed techniques improve parsing accuracies.

References

Deng Cai and Hai Zhao. 2016. Neural word segmentation learning for Chinese. In *Proceedings of ACL*, Berlin, Germany, August.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750, Doha, Qatar, October.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of ACL*, pages 167–176, Beijing, China, July.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 25–70.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of ACL*, pages 302–312, Beijing, China, July.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of ACL*, pages 334–343, Beijing, China, July.

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

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *Proceedings of ACL*, pages 959–967, Columbus, Ohio, June.

Erick Fonseca and Sandra Aluísio. 2015. A deep architecture for non-projective dependency parsing. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 56–61, Denver, Colorado, June.

Stuart Geman and Mark Johnson. 2002. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 279–286, Philadelphia, Pennsylvania, USA, July.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin crfs: Training log-linear models with cost functions. In *Proceedings of NAACL*, pages 733–736, Los Angeles, California, June.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for english. In *16th Nordic Conference of Computational Linguistics*, pages 105–112. University of Tartu.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. 1999. Estimators for stochastic "unification-based" grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 535–541, College Park, Maryland, USA, June.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of ACL*, pages 1–11, Uppsala, Sweden, July.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Phong Le and Willem Zuidema. 2014. The inside-outside recursive neural network model for dependency parsing. In *Proceedings of EMNLP*, pages 729–739, Doha, Qatar, October.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of ACL*, pages 1381–1391, Baltimore, Maryland, June.

Zhenghua Li, Min Zhang, and Wenliang Chen. 2014. Ambiguity-aware ensemble training for semi-supervised dependency parsing. In *Proceedings of ACL*, pages 457–467, Baltimore, Maryland, June.

Xuezhe Ma and Eduard Hovy. 2015. Efficient inner-to-outer greedy algorithm for higher-order labeled dependency parsing. In *Proceedings of EMNLP*, pages 1322–1328, Lisbon, Portugal, September.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *Proceedings of COLING*, pages 785–796, Mumbai, India, December.

Xuezhe Ma and Hai Zhao. 2015. Probabilistic models for high-order projective dependency parsing. *arXiv preprint arXiv:1502.04174*.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of ACL*, pages 617–622, Sofia, Bulgaria, August.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of ACL*, pages 91–98, Ann Arbor, Michigan, June.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958, Columbus, Ohio, June.

Mark A Paskin. 2001. Cubic-time parsing and learning algorithms for grammatical bigram models. Technical report.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of ACL*, pages 313–322, Beijing, China, July.

Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*, pages 455–465, Sofia, Bulgaria, August.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of ENNLP*, pages 157–166, Honolulu, Hawaii, October.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180.

Rui Wang, Masao Utiyama, Isao Goto, Eiichro Sumita, Hai Zhao, and Bao-Liang Lu. 2013. Converting continuous-space language models into n-gram language models for statistical machine translation. In *Proceedings of EMNLP*, pages 845–850, Seattle, Washington, USA, October.

Rui Wang, Hai Zhao, Bao-Liang Lu, Masao Utiyama, and Eiichiro Sumita. 2014. Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of EMNLP*, pages 189–195, Doha, Qatar, October.

Peilu Wang, Yao Qian, Frank Soong, Lei He, and Hai Zhao. 2016. Learning distributed word representations for bidirectional lstm recurrent neural network. In *Proceedings of NAACL*, June.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL*, pages 323–333, Beijing, China, July.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344, Dublin, Ireland, August.

Yue Zhang and Stephen Clark. 2008. A tale of two parsers: Investigating and combining graph-based and transition-based dependency parsing. In *Proceedings of EMNLP*, pages 562–571, Honolulu, Hawaii, October.

Meishan Zhang and Yue Zhang. 2015. Combining discrete and continuous features for deterministic transition-based dependency parsing. In *Proceedings of EMNLP*, pages 1316–1321, Lisbon, Portugal, September.

Zhisong Zhang and Hai Zhao. 2015. High-order graph-based neural dependency parsing. In *Proceedings of the 29th Pacific Asia Conference on Language, Information, and Computation*, pages 114–123, Shanghai, China, October.

Hai Zhao, Wenliang Chen, and Chunyu Kit. 2009a. Semantic dependency parsing of NomBank and PropBank: An efficient integrated approach via a large-scale feature selection. In *Proceedings of EMNLP*, pages 30–39, Singapore, August.

Hai Zhao, Wenliang Chen, Chunyu Kity, and Guodong Zhou. 2009b. Multilingual dependency learning: A huge feature engineering method to semantic dependency parsing. In *Proceedings of CoNLL*, pages 55–60, Boulder, Colorado, June.

Hai Zhao, Yan Song, Chunyu Kit, and Guodong Zhou. 2009c. Cross language dependency parsing using a bilingual lexicon. In *Proceedings of ACL*, pages 55–63, Suntec, Singapore, August.

Hai Zhao, Xiaotian Zhang, and Chunyu Kit. 2013. Integrative semantic dependency parsing via efficient large-scale feature selection. *Journal of Artificial Intelligence Research*, 46:203–233.

Hai Zhao. 2009. Character-level dependencies in chinese: Usefulness and learning. In *Proceedings of EACL*, pages 879–887, Athens, Greece, March.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of ACL*, pages 1213–1222, Beijing, China, July.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of ACL*, pages 1159–1168, Beijing, China, July.

A Search-Based Dynamic Reranking Model for Dependency Parsing

Hao Zhou† **Yue Zhang**‡ **Shujian Huang**† **Junsheng Zhou***
Xin-Yu Dai† **Jiajun Chen**†
†State Key Laboratory for Novel Software Technology, Nanjing University, China
‡Singapore University of Technology and Design, Singapore
* Nanjing Normal University, China
zhouh@nlp.nju.edu.cn, yue_zhang@sutd.edu.sg
{huangshujian,daixinyu,chenjj}@nju.edu.cn, zhoujs@njnu.edu.cn

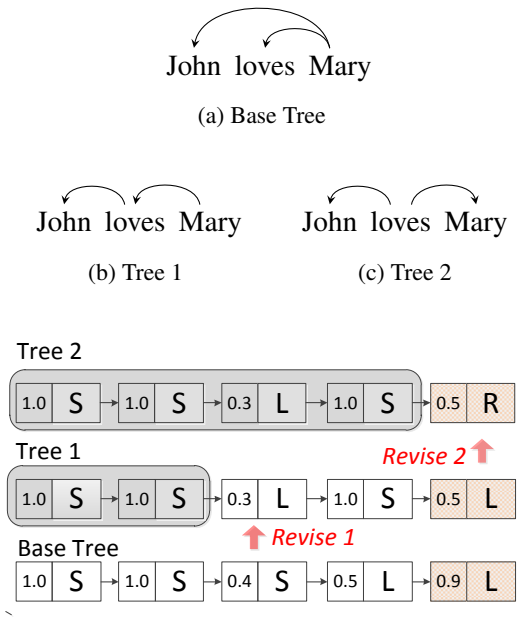
Abstract

We propose a novel reranking method to extend a deterministic neural dependency parser. Different to conventional k-best reranking, the proposed model integrates search and learning by utilizing a dynamic action revising process, using the reranking model to guide modification for the base outputs and to rerank the candidates. The dynamic reranking model achieves an absolute 1.78% accuracy improvement over the deterministic baseline parser on PTB, which is the highest improvement by neural rerankers in the literature.

1 Introduction

Neural network models have recently been exploited for dependency parsing. Chen and Manning (2014) built a seminal model by replacing the SVM classifier at the transition-based MaltParser (Nivre et al., 2007) with a feed-forward neural network, achieving significantly higher accuracies and faster speed. As a local and greedy neural baseline, it does not outperform the best discrete-feature parsers, but nevertheless demonstrates strong potentials for neural network models in transition-based dependency parsing.

Subsequent work aimed to improve the model of Chen and Manning (2014) in two main directions. First, global optimization learning and beam search inference have been exploited to reduce error propagation (Weiss et al., 2015; Zhou et al., 2015). Second, recurrent neural network models have been used to extend the range of neural features beyond a local window (Dyer et al., 2015; Ballesteros et al., 2015). These methods give accuracies that are competitive to the best results in the literature.



(d) 2-step action revising process for sentence "John loves Mary". Numbers before actions are the probabilities for that action.

Figure 1: Example action revising process. S, L, R stand for the SHIFT, LEFT, RIGHT actions, respectively (Section 2).

Another direction to extend a baseline parser is reranking (Collins and Koo, 2000; Charniak and Johnson, 2005; Huang, 2008). Recently, neural network models have been used to constituent (Socher et al., 2013; Le et al., 2013) and dependency (Le and Zuidema, 2014; Zhu et al., 2015) parsing reranking. Compared with rerankers that rely on discrete manual features, neural network rerankers can potentially capture more global information over whole parse trees.

Traditional rerankers are based on chart parsers, which can yield exact k-best lists and forests. For reranking, this is infeasible for the transition-based neural parser and neural reranker, which

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1393–1402,
Berlin, Germany, August 7-12, 2016.

have rather weak feature locality. In addition, k-best lists from the baseline parser are not necessarily the best candidates for a reranker. Our preliminary results show that reranking candidates can be constructed by modifying unconfident actions in the baseline parser output, and letting the baseline parser re-decode the sentence from the modified action. In particular, revising two incorrect actions of the baseline parser yields oracle with 97.79% UAS, which increases to 99.74% by revising five actions. Accordingly, we design a novel search-based dynamic reranking algorithm by revising baseline parser outputs.

For example, the sentence: "*John loves Mary*", the baseline parser generates a *base tree* (Figure 1a) using 5 shift-reduce actions (Figure 1d) of Section 2. The gold parse tree can be obtained by a 2-step action revising process:

$$\text{Base Tree} \xrightarrow{\text{Revise 1}} \text{Tree 1} \xrightarrow{\text{Revise 2}} \text{Tree 2}$$

As shown in Figure 1d, we first revise the least confident action S of the *base tree*, running the baseline parser again from the revised action to obtain *tree 1*. This corrects the *John* $\curvearrowleft$ *loves* dependency arc. Then we obtain the gold parsing tree (*tree 2*) by further revising the least confident action in *tree 1* on the second action sequence.

Rather than relying on the baseline model scores alone for deciding the action to revise (*static search*), we build a neural network model to guide which actions to revise, as well as to rerank the output trees (*dynamic search*). The resulting model integrates search and learning, yielding the minimum amount of candidates for the best accuracies. Given the extensively fast speed of the baseline parser, the reranker can be executed with high efficiency.

Our dynamic search reranker has two main advantages over the static one: the first is *training diversity*, the dynamic reranker searches over more different structurally diverse candidate trees, which allows the reranker to distinguish candidates more easily; the second is *reranking oracle*, with the guidance of the reranking model, the dynamic reranker has a better reranking oracle compared to the static reranker.

On WSJ, our dynamic reranker achieved 94.08% and 93.61% UAS on the development and test sets, respectively, at a speed of 16.1 sentences per second. It yields a 0.44% accuracy improvement (+1.78%) from the same number of candidates, compared to a static reranker (+1.34%), obtaining the largest accuracy improvement among related neural rerankers.

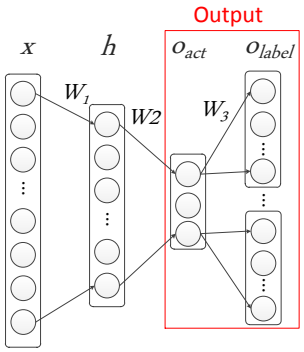


Figure 2: Hierarchical neural parsing model.

2 Baseline Dependency Parser

Transition-based dependency parsers scan an input sentence from left to right, performing a sequence of transition actions to predict its parse tree (Nivre, 2008). We employ the *arc-standard* system (Nivre et al., 2007), which maintains partially-constructed outputs using a *stack*, and orders the incoming words in the sentence in a *queue*. Parsing starts with an empty stack and a queue consisting of the whole input sentence. At each step, a *transition action* is taken to consume the input and construct the output.

Formally, a *parsing state* is denoted as $\langle j, S, L \rangle$, where S is a stack of subtrees $[\dots s_2, s_1, s_0]$, j is the head of the queue (i.e. $[\ q_0 = w_j, q_1 = w_{j+1} \cdots]$), and L is a set of dependency arcs that has been built. At each step, the parser chooses one of the following actions:

- SHIFT (S): move the front word w_j from the queue onto the stacks.
- LEFT-l (L): add an arc with label l between the top two trees on the stack ($s_1 \leftarrow s_0$), and remove s_1 from the stack.
- RIGHT-l (R): add an arc with label l between the top two trees on the stack ($s_1 \rightarrow s_0$), and remove s_0 from the stack.

Given the sentence "*John loves Mary*", the gold standard action sequence is S, S, L, S, R.

2.1 Model

Chen and Manning (2014) proposed a deterministic neural dependency parser, which rely on dense embeddings to predict the optimal actions at each step. We propose a variation of Chen and Manning

(2014), which splits the output layer into two hierarchical layers: the action layer and dependency label layer. The hierarchical parser determines a action in two steps, first deciding the *action type*, and then the *dependency label* (Figure 2).

At each step of deterministic parsing, the neural model extracts n atomic features from the parsing state. We adopt the feature templates of Chen and Manning (2014). Every atomic feature is represented by a feature embedding $e_i \in \mathbb{R}^d$, An *input layer* is used to concatenate the n feature embeddings into a vector $x = [e_1; e_2 \dots e_n]$, where $x \in \mathbb{R}^{d \cdot n}$. Then x is mapped to a d_h-dimensional *hidden layer* h by a mapping matrix $W_1 \in \mathbb{R}^{d_h \times d \cdot n}$ and a cube activation function for feature combination:

$$h = (W_1 x + b_1)^3 \tag{1}$$

Our method is different from Chen and Manning (2014) in the output layer. Given the hidden layer h, the action type output layer o_{act} and the label output layer $o_{label}(a_i)$ of the action type a_i are computed as

$$o_{act} = W_2 h \tag{2}$$

$$o_{label}(a_i) = W_3^i h \ , \tag{3}$$

Where $W_2 \in \mathbb{R}^{d_a \times d_h}$ is the mapping matrix from the hidden layer to the action layer, and d_a is the number of action types. $W_3^i \in \mathbb{R}^{d_{label} \times d_h}$ is the mapping matrix from the hidden layer to the corresponding label layer, d_{label} is the number of dependency labels.

The probability of a labeled action $y_{i,j}$ given its history $Acts$ and input x is computed as:

$$\begin{aligned} & p(y_{i,j} \mid x, Acts) \\ = \ & p(a_i \mid x, Acts) \times p(l_j \mid x, Acts, a_i) \end{aligned} \tag{4}$$

where

$$p(a_i \mid x, Acts) = \frac{e^{o^i_{act}}}{\sum_{k=1}^{d_a} e^{o^k_{act}}} \tag{5}$$

$$p(l_j \mid x, Acts, a_i) = \frac{e^{o^j_{label}(a_i)}}{\sum_{k=1}^{d_{label}} e^{o^k_{label}(a_i)}} \ , \tag{6}$$

Here a_i is the i_{th} action in the action layer, and l_j is the j_{th} label in the label layer for a_i.

In training, we use the cross-entropy loss to maximum the probability of training data A:

$$L(\theta) = -\sum_{y_{i,j} \in A} \log p(y_{i,j} \mid x, Acts) \tag{7}$$

Experiments show that our hierarchical neural parser is both faster and slightly accurate than the original neural parser.

3 Reranking Scorer

We adopt the recursive convolutional neural network (RCNN) of Zhu et al. (2015) for scoring full trees. Given a dependency subtree rooted at h, c_i ($0 < i \leq L$) is the i_{th} child of h. The dependency arc (h, c_i) is represented by:

$$z_i = tanh(W^{(h,c_i)} p_i) \ , \tag{8}$$

where

$$p_i = w_h \oplus x_{c_i} \oplus d^{(h,c_i)} \tag{9}$$

Here $p_i \in \mathbb{R}^n$ is the concatenation of head word embedding w_h, child phrase representation x_{c_i} and the distance embeddings $d^{(h,c_i)}$. $W^{(h,c_i)} \in \mathbb{R}^{m \times n}$ is a linear composition matrix, which depends on the POS tags of h and c_i. The subtree phrase representation x^h are computed using a max-pooling function on rows, over the matrix of arc representations Z^h.

$$Z^h = [z_1, z_2, \dots, z_L] \tag{10}$$

$$x^h_j = \max_i Z^h_{j,i}, 0 < j < m \tag{11}$$

The subtree with the head h is scored by:

$$score(h) = \sum_{i=1}^{L} v^{h,c_i} z_i \tag{12}$$

Here, v^{h,c_i} is the score vector, which is a vector of parameters that need to be trained. The score of the whole dependency tree y is computed as:

$$s_t(x, y, \Theta) = \sum_{w \in y} score(w), \tag{13}$$

where w is the node in tree y and Θ denotes the set of parameters in the network.

4 Search-based Dynamic Reranking for Dependency Parsing

Using the hierarchical parser of Section 2 as the baseline parser, we propose a search-based dynamic reranking model, which integrates search and learning by searching the reranking candidates dynamically, instead of limiting the scope to a fixed k-best list. The efficiency of the reranking model is guaranteed by 3 properties of the baseline parser, namely *revising efficiency*, *probability diversity* and *search efficiency*.

Revising Depth	UAS	LAS
0	92.28	91.15
1	95.76	94.42
2	97.79	96.63
3	98.77	97.55
4	99.39	98.15
5	99.74	98.47

Table 1: Oracle of the baseline parser after revising actions. Revising depth is the maximum number of revised actions for one sentence.

Action Type		Num	Average Probability
Gold	Shift	39194	99.38%
	Right	19477	98.90%
	Left	19556	99.61%
Incorrect	Shift	968	84.96%
	Right	746	85.88%
	Left	338	85.03%

Table 2: Average action probabilities.

4.1 Properties of the Baseline Parser

To demonstrate the above three properties, we give some preliminary results for the baseline. To parse the 1,695 sentences in Section 22 of WSJ, our baseline parser needs to perform 78,227 shift-reduce actions. During the process, if we correct every encountered incorrectly determined action and let the baseline parser re-decode the sentence from the point, we need to revise 2,052 actions, averaging 1.2 actions per sentence. In other words, the baseline parser can parse the 1,695 sentences correctly with 2,052 action being revised.

Note that the revise operation is required to change the action type (i.e. S, L). After revising the action type, the optimal dependency label will be chosen for parsing by the hierarchical baseline parser. We only modify the action type in the revising process. Thus the modified trees are always structurally different instead of only with different dependency labels compared to the original one, which guarantees structured diversity.

Revising Efficiency It can be seen from Table 1 that revising one incorrect action results in 3.5% accuracy improvement. We obtain a 99.74% UAS after a maximum 5 depth revising. Although we only revise the action type, the LAS goes up with the UAS. The property of *revising efficiency* suggests that high quality tree candidates can be found with a small number of changes.

Probability Diversity Actions with lower probabilities are more likely to be incorrect. We compute the average probabilities of gold and incorrect actions in parsing the section 22 of WSJ (Table 2), finding that most gold actions have very high probabilities. The average probabilities of the gold actions is much higher than that of the incorrectly predicted ones, indicating that revising actions with lower probabilities can lead to better trees.

Search Efficiency The fast speed of the baseline parser allows the reranker to search a large number of tree candidates efficiently. With the graph stack trick (Goldberg et al., 2013), the reranker only needs to perform partial parsing to obtain new trees. This enables a fast reranker in theory.

4.2 Search Strategy

Given an output sequence of actions by the baseline parser, we revise the action with the lowest *probability margin*, and start a new branch by taking a new action at this point. The probability margin of an action a is computed as: $p(a_{max}) - p(a)$, where a_{max} is the action taken by the baseline, which has the highest model probability. a is taken instead of a_{max} for this branch, and the baseline parser is executed deterministically until parsing finishes, thus yielding a new dependency tree. We require that the action type must change in the revision and the most probable dependency label among all for the revised action type will be used.

Multiple strategies can be used to search for the revised reranking process. For example, one intuitive strategy is best-first, which modifies the action with the lowest probability margin among all sequences of actions constructed so far. Starting from the original output of the baseline parser, modifying the action with the lowest probability margin results in a new tree. According to the best-first strategy, the action with the lowest probability margin in the two outputs will be revised next to yield the third output. The search repeats until k candidates are obtained, which are used as candidates for reranking.

The best-first strategy, however, does not consider the quality of the output, which is like a greedy process. A better candidate (with higher F1 score) is more likely to take us to the gold tree. With the best-first strategy, we revise one tree at each time. If the selected tree is not the optimal one, the revised tree will be less likely the gold one. Revising a worse output is less likely to generate the gold parse tree compared with revising a relatively better output. Our preliminary experi-

ments confirms this intuition. As a result, we take a beam search strategy, which uses a beam to hold b outputs to modify.

For each tree in beam search, most f actions with the lowest probability margin are modified, leading to $b \times f$ new trees. Here, b is the beam size, f is the revising factor. From these trees, the b best are put to the beam for the next step. Search starts with the beam containing only the original base parse, and repeats for l steps, where l is called the revising depth. The best tree will be selected from all the trees constructed. The search process for example in Figure 1 is illustrated in Figure 3, in which $b = 1$, $f = 3$ and $l = 2$.

At each iteration, the b best candidates can be decided by the baseline parser score alone, which is the product of the probability of each action. We call this the *static search* reranking. As mentioned in the introduction, the baseline model score might not be the optimal criteria to select candidates for reranking, since they may not reflect the best oracle or diversity. We introduce a *dynamic search* strategy instead, using the reranking model to calculate heuristic scores for guiding the search.

4.3 Search-Based Dynamic Reranking

Doppa et al. (2013) propose that structured-prediction by learning guide search should maintain two different scoring functions, a heuristic function for guiding search and a cost function for obtaining the best output. Following Doppa et al. (2013), we use the RCNN in Section 3 to yield two different scores, namely a heuristic score $s_t(x, y, \Theta_h)$ to guide the search of revising, and a cost score $s_t(x, y, \Theta_c)$ to select the best tree output.

Denote $b(i)$ as the beam at i-th step of search, k-best candidates in the beam of $i + 1$ step is:

$$b(i+1) = \arg\underset{c \in c(i)}{K}(s_t(x, c, \Theta_h) + s_b(x, c)), \quad (14)$$

where $c(i)$ denotes the set of newly constructed trees by revising trees in $b(i)$, $s_b(x, c)$ is the baseline model score and $\arg K$ leaves the k best candidate trees to the next beam. Finally, the output tree y_i of reranking is selected from all searched trees C in the revising process

$$y_i = \arg\max_{c \in C}(s_t(x, c, \Theta_c) + s_b(x, c)) \quad (15)$$

Interpolated Reranker In testing, we also adopt the popular mixture reranking strategy (Hayashi et al., 2013; Le and Mikolov, 2014), which obtains better reranking performance by a linear combination of the reranking score and the baseline model score.

$$y_i = \arg\max_{y \in \tau(x_i)} (\beta(s_t(x_i, y, \Theta_c) + s_t(x, y, \Theta_h)) + (1 - \beta)s_b(x_i, y)) \quad (16)$$

Here y_i is the final output tree for a sentence x_i; $\tau(x_i)$ returns all the trees candidates of the dynamic reranking; $\beta \in [0, 1]$ is a hyper-parameter.

Algorithm 1: Training Algorithm for the Search-Based Dynamic Reranking.

Input: Sentence **x**, Gold Trees **y**
Output: Θ_h, Θ_c
for *iter* ← *1* **to** *N* **do**
  D_h = [];
  D_k = [];
  foreach *(x, y)* ∈ **(x, y) do**
    bestHScoreT = null;
    bestCScoreT = null;
    bestUAST = null;
    initTree = BASELINEPARSE(x);
    b_1 = [*initTree*];
    b_2 = [];
    for $d \leftarrow 1$ **to** $depth$ **do**
      foreach $t \in b_1$ **do**
        revisedActs = SEEK (t);
        revisedTrees = REVISE (t, *revisedActs*);
        bestK = SORT (*revisedTrees*, Θ_h)
        b_2.ADD (*bestK*);
        bestHScoreT = MAXSCORE (*bestHScoreT*, *revisedTrees*, Θ_h);
        bestCScoreT = MAXSCORE (*bestCScoreT*, *revisedTrees*, Θ_c);
        bestUAST = MAXUAS (*bestUAST*, *revisedTrees*, y)
      $b_1 = b_2$;
      b_2 = [];
    D_h.ADD (x, *bestUAST*, *bestTScoreT*);
    D_c.ADD (x, y, *bestCScoreT*);
  UPDATE(D_h, Θ_h);
  UPDATE(D_c, Θ_c);

4.4 Training

As k-best neural rerankers (Socher et al., 2013; Zhu et al., 2015), we use the max-margin criterion to train our model in a stage-wise manner (Doppa et al., 2013). Given training data D_c = $(x_i, y_i, \hat{y}_i)_{i=1}^N$, where x_i is the sentence, $\hat{y}_i$ is the output tree with highest cost score and y_i is the corresponding gold tree, the final training objective is to minimize the loss function $J(\Theta_c)$, plus a

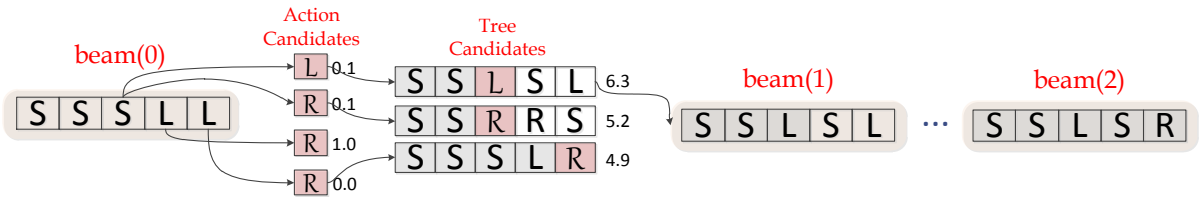


Figure 3: The beam search revising process of the example in Figure 1 with $b = 1$, $f = 3$ and $l = 2$

l_2-regularization:

$$J(\Theta_c) = \frac{1}{|D_c|} \sum_{(x_i, y_i, \hat{y}_i) \in D_c} r_i(\Theta_c) + \frac{\lambda}{2}||\Theta_c|| \quad (17)$$

$$r_i(\Theta_c) = \max(0, s_t(x_i, \hat{y}_i, \Theta_c) + \Delta(y_i, \hat{y}_i) - s_t(x_i, y_i, \Theta_c)) \quad (18)$$

Here, Θ_c is the model, $s_t(x_i, y_i, \Theta_c)$ is the cost reranking score for y_i.

$$\Delta(y_i, \hat{y}_i) = \sum_{d \in \hat{y}_i} \kappa \mathbf{1}\{d \notin y_i\} \quad (19)$$

$\Delta(y_i, \hat{y}_i)$ is the structured margin loss between y_i and $\hat{y}_i$, measured by counting the number of incorrect dependency arcs in the tree (Goodman, 1998; Zhu et al., 2015).

Given training data $D_h = (x_i, y'_i, \hat{y}'_i)_{i=1}^N$ for the heuristic score model, the training objective is to minimize the loss between the tree with the best UAS y'_i and the tree with the best heuristic reranking score $\hat{y}'_i$.

$$J(\Theta_h) = \frac{1}{|D_h|} \sum_{(x_i, y'_i, \hat{y}'_i) \in D_h} r_i(\Theta_h) + \frac{\lambda}{2}||\Theta_h|| \quad (20)$$

$$r_i(\Theta_h) = \max(0, s_t(x_i, \hat{y}'_i, \Theta_h)) - s_t(x_i, y'_i, \Theta_h) \quad (21)$$

The detailed training algorithm is given by Algorithm 1. AdaGrad (Duchi et al., 2011) updating with subgradient (Ratliff et al., 2007) and mini-batch is adopted for optimization.

5 Experiments

5.1 Set-up

Our experiments are performed using the English Penn Treebank (PTB; Marcus et al., (1993)). We follow the standard splits of PTB3, using sections 2-21 for training, section 22 for development and section 23 for final testing. Following prior work on reranking, we use Penn2Malt[1] to convert constituent trees to dependency trees. Ten-fold POS jackknifing is used in the training of the baseline parser. We use the POS-tagger of Collins (2002) to assign POS automatically. Because our reranking model is a dynamic reranking model, which generates training instances during search, we train 10 baseline parsing models on the 10-fold jackknifing data, and load the baseline parser model dynamically for reranking training .

We follow Chen and Manning (2014), using the set of pre-trained word embeddings with a dictionary size of 13,000[2] from Collobert et al. (2011). The word embeddings were trained on the entire English Wikipedia, which contains about 631 million words.

5.2 Hyper-parameters

There are two different networks in our system, namely a hierarchical feed-forward neural network for the baseline parsing and a recursive convolution network for dynamic reranking. The hyper-parameters of the hierarchical parser are set as described by Chen and Manning (2014), with the embedding size $d = 50$, the hidden layer size $d_h = 300$, the regularization parameter $\lambda = 10^{-8}$, the initial learning rate of Adagrad $\alpha = 0.01$ and the batch size $b = 100{,}000$. We set the hyper-parameters of the RCNN as follows: word embedding size $d^w_{rnn} = 25$, distance embedding size $d^d_{rnn} = 25$, initial learning rate of Adagrad $\alpha_{rnn} = 0.1$, regularization parameter $\lambda_{rnn} = 10^{-4}$, margin loss discount $\kappa = 0.1$ and revising factor $f = 8$.

5.3 The Hierarchical Neural Parser

Shown in Table 3, the proposed hierarchical base parser is 1.3 times faster, and obtains a slight accuracy improvement (Table 3) upon the parser of Chen and Manning (2014). The reason for the

[1] http://stp.lingfil.uu.se/ nivre/research/Penn2Malt.html
[2] http://ronan.collobert.com/senna/

Parser	dev		test		Speed
	UAS	LAS	UAS	LAS	
hiero	92.28	91.15	91.83	90.76	884.7
original	92.00	90.89	91.67	90.62	682.3

Table 3: Performance comparison between the hierarchical and original neural parsers. Speed: sentences per second.

Beam Size	1	2	4	8
UAS	93.38	93.45	93.81	93.51
Oracle	96.95	97.29	97.80	97.81
K	22.57	37.16	65.8	118.7

Table 4: Accuracies of the revising reranker with different beam sizes on the development set.

speed gain is that smaller output layer leads to less computation of mapping from the hidden layer to the output layer in neural networks (Morin and Bengio, 2005; Mnih and Hinton, 2009).

5.4 Development Tests

For the beam search dynamic reranking model, the selection of beam size b and revising depth l affect the accuracy and efficiency of the reranker. We tune the values on the development set.

Beam Size A proper beam size balances efficiency and accuracy in the search process. The reranking accuracies with different beam sizes are listed in Table 4. Here, the oracle is the best UAS among searched trees during reranking. K is the number of searched candidate trees in testing. The UAS and parsing oracle both go up with increasing the beam size. Reranking with beam size = 4 gives the best development performance. We set the final beam size as 4 in the next experiments.

Revising Depth As shown in Table 5, with revising depth increasing from 1 to 3, the reranker obtains better parsing oracle. The depth of 3 gives the best UAS 93.81% on the development set. The parsing oracle stops improving with deeper revised search. This may because in the fourth search step, the high quality trees begin to fall out the beam, resulting in worse output candidates, which make the revising step yield less oracle gains. We set the search depth as 3 in the next experiments.

Integrating Search and Learning Shown in Table 6, the dynamic and static rerankers both achieve significant accuracy improvements over the baseline parser. The dynamic reranker gives much better improvement, although the oracle of dynamic reranker is only 0.2% higher than the static one. This demostrates the benefit of diversity. The candidates are always the same for static search, but the dynamic reranker searches more diverse tree candidates in different iterations of training.

Revising Depth	1	2	3	4
UAS	93.22	93.50	93.81	93.53
Oracle	96.31	97.57	97.80	97.81
K	8.87	38.45	65.8	90.28

Table 5: Accuracies of the revised reranker with different revising depths on development set.

Search Type	UAS	+UAS	Oracle
Dynamic	93.81	+1.53	97.80
Static	93.29	+1.01	97.61

Table 6: Comparing dynamic and the static search.

To further explore the impact of training diversity to dynamic reranking, we also compare the dynamic search reranker of training and testing with different revising depth. In Table 7, *origin* is the results by training and testing with the same depth d. Results of *ts* is obtained by training with d = 3, and testing with a smaller d. For example, a reranker with training d = 3 and testing d = 2 achieves better performance than with training d = 2 and testing d = 2. The testing oracle of the former reranker is lower than the later, yet the former learns more from the training instance, obtaining better parsing accuracies. This again indicates that training diversity is very important besides the oracle accuracy.

Interpolated Reranker Finally, we mix the baseline model score and the reranking score by following Hayashi et al. (2013) and Zhu et al. (2015), and the mixture parameter β is optimized by searching with the step size of 0.005. With the mixture reranking trick, the dynamic reranker obtains an accuracy of 94.08% (Table 8), with an improvement of 0.28% on the development set.

5.5 Final Results

Comparison with Dependency Rerankers In Table 9, we compare the search-based dynamic rerankers with a list of dependency rerankers. The reranking models of Hayashi et al. (2013) and Hayashi et al. (2011) are forest reranking models. Le and Zuidema (2014) and Zhu et al. (2015) are neural k-best reranking models. Our dynamic

Depth		1	2	3
ordinary	UAS	93.22	93.50	93.81
	oracle	96.31	97.57	97.80
ts	UAS	93.59	93.79	93.81
	oracle	96.29	93.42	97.80

Table 7: Accuracies of the revised reranker with different revising depths on the development set.

Type	static	dynamic
w/o mixture	93.29	93.81
w/ mixture	93.53	94.08

Table 8: Effects of interpolated reranking.

reranking model achieves the highest accuracy improvement over the baseline parser on both the development and test sets. We obtain the best performance on the development set. Zhu et al. (2015) achieved higher accuracy on the test set, but they adopted a better baseline parser than ours, which could not be used in our dynamic reranker because it is not fast enough and will make our reranker slow in practice.

Comparing with Neural Dependency Parsers We also compare parsing accuracies and speeds with a number of neural network dependency parsers. Dyer et al. (2015) proposed a dependency parser with stack LSTM; Zhou et al. (2015) applied the beam search for structured dependency parsing. Both achieved significant accuracy improvements over the deterministic neural parser of Chen and Manning (2014). Our dynamic search reranker obtains a 93.61% UAS on the test set, which is higher than most of the neural parsers except Weiss et al. (2015), who employ a structured prediction model upon the neural greedy baseline, achieving very high parsing accuracy.

5.6 Results on Stanford dependencies

We also evaluate the proposed static and dynamic rerankers on Staford dependency treebank. The main results are consistent with CoNLL dependency treebank with the dynamic reranker achieving a 0.41% accuracy improvement upon the static reranker on test data. But the parsing accuracy on Stanford dependency is not the state-of-the-art. We speculate that there may be two reasons. First, the baseline parsing accuracy on Stanford dependencies is lower than CoNLL. Second, all the hyper-parameters are tuned on the CoNLL data.

Reranker		UAS	
		dev	test
Hayashi et al. (2011)		N/A	92.87 (+0.97)
Hayashi et al. (2013)		N/A	93.12 (+0.62)
Le and Zuidema (2014)		N/A	93.12 (+1.09)
(Zhu et al., 2015)	baseline	92.45	92.35
	reranking	93.50 (+1.05)	**93.83** (+1.48)
This work (CoNLL)	baseline	92.28	91.83
	dynamic	**94.08** (+**1.80**)	93.61 (+**1.78**)
	static	93.53 (+1.25)	93.17 (+1.34)

Table 9: Comparison of dependency rerankers.

6 Related Work

Neural Networks Reranking A line of work has been proposed to explore reranking using neural networks. Socher et al. (2013) first proposed a neural reranker using a recursive neural network for constituent parsing. Le and Zuidema (2014) extended the neural reranker to dependency parsing using a inside-outside recursive neural network (IORNN), which can process trees both bottom-up and top-down. Zhu et al. (2015) proposed a RCNN method, which solved the problem of modeling k-ary parsing tree in dependency parsing. The neural rerankers are capable of capturing global syntax features across the tree. In contrast, the most non-local neural parser with LSTM (Dyer et al., 2015) cannot exploit global features. Different to previous neural rerankers, our work in this paper contributes on integrating search and learning for reranking, instead of proposing a new neural model.

Forest Reranking Forest reranking (Huang, 2008; Hayashi et al., 2013) offers a different way to extend the coverage of reranking candidates, with computing the reranking score in the trees forests by decomposing non-local features with cube-pruning (Huang and Chiang, 2005). In contrast, the neural reranking score encodes the whole dependency tree, which cannot be decomposed for forest reranking efficiently and accurately.

HC-Search Doppa et al. (2013) proposed a structured prediction model with HC-Search strategy and imitation learning, which is closely related to our work in spirit. They used the complete space search (Doppa et al., 2012) for sequence labeling tasks, and the whole search process halts after a specific time bound. Different from them, we propose a dynamic parsing reranking model based on the action revising process, which is a multi-step process by revising the least confident

Type	System	UAS	Speed
Neural	Zhou et al. (2015)	93.28	14.3
	Dyer et al. (2015)‡	93.30	105
	Weiss et al. (2015)†	**93.99**	N/A
	Weiss et al. (2015) semi †	**94.26**	N/A
	Pei et al. (2015)	93.29	N/A
	Chen et al. (2015)	92.60	2.7
	Chen and Manning (2014)	92.00	1013
This work	dynamic	**93.61**	16.1

Table 10: Comparison with neural parsers. Speed: sentences per second. †: results are reported on Stanford dependencies. ‡: results are run by ourself using their codes.

System	UAS	
	dev	test
baseline	91.80	91.41
dynamic	93.44 (+1.64)	92.95 (+1.57)
static	93.09 (+1.29)	92.57 (+1.16)

Table 11: Dynamic reranking results on Stanford dependencies.

actions from the base output and the search stops in a given revising depth. The dynamic reranking model concentrates on extending the training diversity and testing oracle for parsing reranking, which is built on the transition-based parsing framework.

7 Conclusion

In this paper, we proposed a search-based dynamic reranking model using a hierarchical neural base parser and a recursive convolutional neural score model. The dynamic model is the first reranker integrating search and learning for dependency parsing. It achieves significant accuracy improvement (+1.78%) upon the baseline deterministic parser. With the dynamic search process, our reranker obtains a 0.44% accuracy improvement upon the static reranker. The code of this paper can be downloaded from `http://github.com/zhouh/dynamic-reranker`.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments. Xin-Yu Dai is the corresponding author of this paper. This work was supported by the Natural Science Foundation of China (61472183, 6130158, 61472191), the 863 program via 2015AA015406 and Singapore Ministratry of Education Tier 2 Grant T2MOE201301.

References

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180. Association for Computational Linguistics.

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

Xinchi Chen, Yaqian Zhou, Chenxi Zhu, Xipeng Qiu, and Xuanjing Huang. 2015. Transition-based dependency parsing using two heterogeneous gated recursive neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Michael Collins and Terry Koo. 2000. Discriminative reranking for natural language parsing. *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 175–182.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2012. Output space search for structured prediction. In *ICML*.

Janardhan Rao Doppa, Alan Fern, and Prasad Tadepalli. 2013. Hc-search: Learning heuristics and cost functions for structured prediction. In *AAAI*, volume 2, page 4.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*.

Yoav Goldberg, Kai Zhao, and Liang Huang. 2013. Efficient implementation of beam-search incremental parsers. In *ACL (2)*, pages 628–633.

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

Katsuhiko Hayashi, Taro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2011. Third-order variational reranking on packed-shared dependency forests. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1479–1488. Association for Computational Linguistics.

Katsuhiko Hayashi, Shuhei Kondo, and Yuji Matsumoto. 2013. Efficient stacked dependency parsing by forest reranking. *Transactions of the Association for Computational Linguistics*, 1:139–150.

Liang Huang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 53–64. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*.

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

Phong Le, Willem Zuidema, and Remko Scha, 2013. *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, chapter Learning from errors: Using vector-based compositional semantics for parse reranking, pages 11–19. Association for Computational Linguistics.

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

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

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

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

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

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proc. of ACL*.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2007. (online) subgradient methods for structured prediction.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *In Proceedings of the ACL conference*. Citeseer.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53th Annual Meeting of the Association for Computational Linguistics*.

Hao Zhou, Yue Zhang, Shujian Huang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1213–1222, Beijing, China, July. Association for Computational Linguistics.

Chenxi Zhu, Xipeng Qiu, Xinchi Chen, and Xuanjing Huang. 2015. A re-ranking model for dependency parser with recursive convolutional neural network. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Cross-Lingual Sentiment Classification with Bilingual Document Representation Learning

Xinjie Zhou, **Xianjun Wan** and **Jianguo Xiao**
Institute of Computer Science and Technology, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
{zhouxinjie,wanxiaojun,xiaojianguo}@pku.edu.cn

Abstract

Cross-lingual sentiment classification aims to adapt the sentiment resource in a resource-rich language to a resource-poor language. In this study, we propose a representation learning approach which simultaneously learns vector representations for the texts in both the source and the target languages. Different from previous research which only gets bilingual word embedding, our Bilingual Document Representation Learning model BiDRL directly learns document representations. Both semantic and sentiment correlations are utilized to map the bilingual texts into the same embedding space. The experiments are based on the multilingual multi-domain Amazon review dataset. We use English as the source language and use Japanese, German and French as the target languages. The experimental results show that BiDRL outperforms the state-of-the-art methods for all the target languages.

1 Introduction

Sentiment analysis for online user-generated contents has become a hot research topic during the last decades. Among all the sentiment analysis tasks, polarity classification is the most widely studied topic. It has been proved to be invaluable in many applications, such as opinion polling (Tang et al., 2012), customer feedback tracking (Gamon, 2004), election prediction (Tumasjan et al., 2010), stock market prediction (Bollen et al., 2011) and so on.

Most of the current sentiment classification systems are built on supervised machine learning algorithms which require manually labelled data. However, sentiment resources are usually unbalanced in different languages. Cross-lingual sentiment classification aims to leverage the resources in a resource-rich language (such as English) to classify the sentiment polarity of texts in a resource-poor language (such as Japanese). The biggest challenge for cross-lingual sentiment classification is the vocabulary gap between the source language and the target language. This problem is addressed with different strategies in different approaches. Wan (2009) use machine translation tools to translate the training data directly into the target language. Meng et al. (2012) and Lu et al. (2011) exploit parallel unlabeled data to bridge the language barrier. Prettenhofer and Stein (2010) use correspondence learning algorithm to learn a map between the source language and the target language. Recently, representation learning methods has been proposed to solve the cross-lingual classification problem (Xiao and Guo, 2013; Zhou et al., 2015). These methods aim to learn common feature representations for different languages. However, most of the current researches only focus on bilingual word embedding. In addition, these models only use the semantic correlations between aligned words or sentences in different languages while the sentiment correlations are ignored.

In this study, we propose a cross-lingual representation learning model BiDRL which simultaneously learns both the word and document representations in both languages. We propose a joint learning algorithm which exploits both monolingual and bilingual constraints. The monolingual constraints help to model words and documents in each individual language while the bilingual constraints help to build a consistent embedding space across languages.

For each individual language, we extend the paragraph vector model (Le and Mikolov, 2014)

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1403–1412,
Berlin, Germany, August 7-12, 2016.

to obtain word and document embeddings. The traditional paragraph vector model is fully unsupervised without using the valuable sentiment labels. We extend it into a semi-supervised manner by forcing the positive and negative documents to fall into different sides of a classification hyperplane. Learning task-specific embedding has been proved to be effective in previous research. To address the cross-language problem, different strategies are proposed to obtain a consistent embedding space across different languages. Both sentiment and semantic relatedness are exploited while previous studies only use the semantic connection between parallel sentences or documents.

The performance of BiDRL is evaluated on a multilingual multi-domain Amazon review dataset (Prettenhofer and Stein, 2010). By selecting English as the source language, a total of nine tasks are evaluated with different combinations of three different target languages and three different domains. The proposed method achieves the state-of-the-art performance on all the tasks.

The main contributions of this study are summarized as follows:

1) We propose a novel representation learning method BiDRL which directly learns bilingual document representations for cross-lingual sentiment classification. Different from previous studies which only obtain word embeddings, our model can learn vector representations for both words and documents in bilingual texts.

2) Our model leverages both the semantic and sentiment correlations between bilingual documents. Not only the parallel documents but also the documents with the same sentiment are required to get similar representations.

3) Our model achieves the state-of-the-art performances on nine benchmark cross-lingual sentiment classification tasks and it consistently outperforms the existing methods by a large margin.

2 Related Work

Sentiment analysis is the field of studying and analyzing people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions (Liu, 2012). Most of the previous sentiment analysis researches focus on customer reviews and classifying the sentiment polarity is the most widely studied task (Pang et al., 2002).

Cross-lingual sentiment classification is a popular topic in the sentiment analysis community which aims to solve the sentiment classification task from a cross-language view. It is of great importance for the area since it can exploit the existing labeled information in a source language to build a sentiment classification system in any other target language. It saves us from manually labeling data for all the languages in the world which is expensive and time-consuming. Cross-lingual sentiment classification has been extensively studied in the very recent years. Mihalcea et al. (2007) translate English subjectivity words and phrases into the target language to build a lexicon-based classifier. Banea et al. (2010) also use the machine translation service to obtain parallel corpus. It investigates several questions based on the parallel corpus including both the monolingual sentiment classification and cross-lingual sentiment classification. Wan (2009) translates both the training data (English to Chinese) and the test data (Chinese to English) to train different models in both the source and target languages. The co-training algorithm (Blum and Mitchell, 1998) is used to combine the bilingual models together and improve the performance. In addition to the translation-based methods, several studies utilize parallel corpus or existing resources to bridge the language barrier. Balamurali (2012) use WordNet senses as features for supervised sentiment classification. They use the linked WordNets of two languages to bridge the language gap. Lu et al. (2011) consider the multilingual scenario where small amount of labeled data is available in the target language. They attempted to jointly classify the sentiment for both source language and target language. Meng et al. (2012) propose a generative cross-lingual mixture model to leverage unlabeled bilingual parallel data. Prettenhofer and Stein (2010) use the structural correspondence learning algorithm to learn a map between the source language and the target language. Xiao and Guo (2014) treat the bilingual feature learning problem as a matrix completion task.

This work is also related to bilingual representation learning. Zou et al. (2013) propose to use word alignment as the constraints in bilingual word embedding. Each word in one language should be similar to the aligned words in another language. Gouws et al. (2015) propose a similar algorithm but only use sentence-level alignment. It tries to minimize a sampled L2-loss between the bag-of-words sentence vectors of the parallel cor-

pus. Xiao and Guo (2013) learn different representations for words in different languages. Part of the word vector is shared among different languages and the rest is language-dependent. Klementiev et al. (2012) treat the task as a multi-task learning problem where each task corresponds to a single word, and task relatedness is derived from co-occurrence statistics in bilingual parallel data. Hermann and Blunsom (2015) propose the bilingual CVM model which directly minimizes the representation of a pair of parallel documents. The document representation is calculated with a composition function based on words. Chandar A P et al. (2014) and Zhou et al. (2015) use the autoencoder to model the connections between bilingual sentences. It aims to minimize the reconstruction error between the bag-of-words representations of two parallel sentences. Luong et al. (2015) propose the bilingual skip-gram model which leverages the word alignment between parallel sentences. Pham et al. (2015) extend the paragraph vector model to force bilingual sentences to share the same sentence vector.

This study differs with the existing works in the following three aspects, 1) we exploit both the semantic and sentiment correlations of the bilingual texts. Existing bilingual embedding algorithms only use the semantic connection between parallel sentences or documents. 2) Our algorithm learns both the word and document representations. Most of the previous studies simply compute the average of the word vectors in a document. 3) Sentiment labels are used in our embedding algorithm by introducing a classification hyperplane. It not only helps to achieve better embedding performance in each individual language but also helps to bridge the language barrier.

3 Framework

Firstly we introduce several notations used in BiDRL. Let S and S_u denote the documents from the training dataset and the documents from the unlabeled dataset in the source language respectively. For each document $d \in S$, it has a sentiment label $y \in \{1, -1\}$. We denote the sentiment label set of all the documents in S as Y. Let T and T_u denote the documents from the test dataset and the documents from the unlabeled dataset in the target language. The documents in the training and test datasets in the source and target languages are translated into the other language using the online machine translation service - Google Translate[1]. We denote them as T_s (the translation of S) and S_t (the translation of T). We wish to learn a D-dimensional vector representation for all the documents in the dataset.

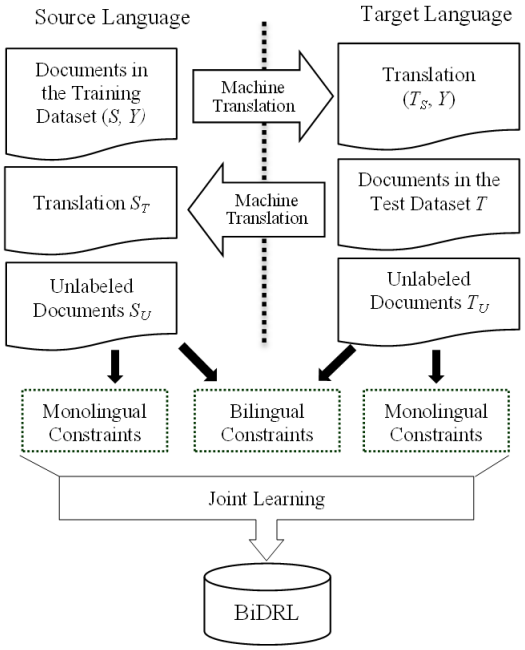


Figure 1: Framework of BiDRL

The general framework of BiDRL is shown in Figure 1. After we obtain the data in the source and target languages, we propose both the monolingual and bilingual constraints to learn the model. The monolingual constraints help to model words and documents in each individual language. The bilingual constraints help to build a consistent embedding space across different languages. The joint learning framework is semi-supervised which uses the sentiment labels Y of the training documents.

3.1 Monolingual Constraints

In this subsection, we describe the representation learning algorithm for the source and target languages. We start from the paragraph vector model (Le and Mikolov, 2014) which has been proved to be one of the state-of-the-art methods for document modeling. In the paragraph vector framework, both documents and words are mapped to unique vectors. Each document is treated as a unique token which is the context of all the words in the document. Therefore, each word in the doc-

[1] http://translate.google.com/

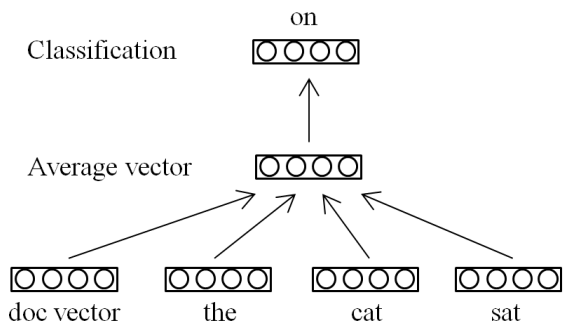


Figure 2: Paragraph vector

ument can be predicted by the context tokens and the document as shown in Figure 2. The idea leads to the following function,

$$\arg\max \sum_{d \in D} \sum_{(w,C) \in d} \log p(w \mid C, d) \quad (1)$$

where D is the document set, w and C are the word and its context in a document d. The only difference between paragraph vector algorithm and the well-known `word2vec` algorithm (Mikolov et al., 2013) is the additional document vector.

The conditional probability of predicting a word from its context is modeled via softmax which is very expensive to compute. It is usually approximately solved via negative sampling. A log-bilinear model is used instead to predict whether two words are in the same context. For a word and context pair (w, C) in a document d, the objective function becomes,

$$\begin{aligned} L_1 = &- \log \sigma(v_w^T \cdot \frac{1}{k+1} \cdot (v_d + \sum_{c \in C} v_c)) - \\ &\sum_{i=1}^{n} E_{w' \in p_n(w)} (\log \sigma(-v_w'^T \cdot \frac{1}{k+1} \cdot (v_d + \sum_{c \in C} v_c))) \end{aligned} \quad (2)$$

where $\sigma(\cdot)$ is the sigmoid function, c is a context word in C, k is the window size, v_w and v_c are the vectors for words and context words, v_d is the vector for the document d, w' is the negative sample from the noise distribution $P_n(w)$. Mikolov et al. (2013) set $P_n(w)$ as the $\frac{3}{4}$ power of the unigram distribution which outperforms the unigram and the uniform distribution significantly.

The paragraph vector model captures the semantic relations between words and documents. In addition, we hope that the vector representation of a document can directly reflect its sentiment. The document vectors associated with different sentiments should fall into different positions in the embedding space. We introduce a logistic regression classifier into the embedding algorithm. For each labeled document d with the label y, we use it to classify the sentiment based on the cross entropy loss function,

$$\begin{aligned} L_2 = &-y \log \sigma((x^T v_d + b) \\ &- (1 - y) \log \sigma(-x^T v_d - b)) \end{aligned} \quad (3)$$

where x is the weight vector for the features and b is the bias, v_d is the document vector.

Combining the above two loss functions, we obtain the monolingual embedding algorithm. For the source language, we get the word and document representations via

$$\arg\min L_s = \sum_{d \in S \cup S_u} \sum_{(w,C) \in d} L_1 + \sum_{d \in S} L_2 \quad (4)$$

The embedding for the target language is obtained similarly,

$$\arg\min L_t = \sum_{d \in T \cup T_u} \sum_{(w,C) \in d} L_1 + \sum_{d \in T_s} L_2 \quad (5)$$

where the labeled dataset T_s is translated from the source language.

3.2 Bilingual Constraints

The key problem of bilingual representation learning is to obtain a consistent embedding space across the source and the target languages. We propose three strategies for BiDRL to bridge the language gap.

The first strategy is to share the classification hyperplane in Equation 3. The logistics regression parameter x and b are the same in the source and the target languages. By sharing the same classification parameter, bilingual documents with the same sentiment will fall into similar areas in the embedding space. Therefore, introducing the logistic regression classifier in our embedding algorithm not only helps to obtain task-specific embedding but also helps to narrow the language barrier.

The second strategy is to minimize the difference between a pair of parallel documents, i.e., the

original documents and the translated documents. Such word or document based regularizer is widely used in previous works (Gouws et al., 2015; Zou et al., 2013). We simply measure the differences of two documents via the Euclidean Distance. The parallel documents refer to the documents in S and T and their corresponding translations (S_t and T_s). It leads to the following loss function,

$$L_3 = \sum_{(d_s,d_t)\in(S,T_s)} \|v_{d_s} - v_{d_t}\|^2 + \sum_{(d_s,d_t)\in(S_t,T)} \|v_{d_s} - v_{d_t}\|^2 \quad (6)$$

where (d_s, d_t) is a pair of parallel documents and v_* is their vector representations.

Our third strategy aims to generate similar representations for texts with the same sentiment. The traditional paragraph vector model focuses on modeling the semantic relationship between words and documents while our method aims to preserve the sentiment relationship as well. For each document $d \in S$ in the source language, we find K least similar documents with the same sentiment. The document similarity is measured by cosine similarity using TF-IDF features. We hope that these K documents should have similar representation with document d despite of their textual difference. We denote the K documents as Q_s and their parallel documents in the target language as Q_t. It leads to the following loss function,

$$L_4 = \sum_{d\in S}\Big(\sum_{d_s\in Q_s} \|v_{d_s} - v_d\|^2 + \sum_{d_t\in Q_t} \|v_{d_t} - v_d\|^2\Big) \quad (7)$$

where v_* denotes the vector representation of a document.

Combining the monolingual embedding algorithm and the cross-lingual correlations, we have the overall objective function as follows,

$$\arg\min L = L_s + L_t + L_3 + L_4 \quad (8)$$

which can be solved using stochastic gradient descent (SGD).

After learning BiDRL, we represent each document in the training and test dataset by the concatenation of its vector representation in both the source and the target languages. In particular, for each training document $d \in S$, we represent it as $[v_d\ v_{d'}]$ where d' is its corresponding translation and v_* is the learned document representation in BiDRL. Similarly, for each test document $d \in T$, we represent it as $[v_{d'}\ v_d]$. Afterwards, a logistic regression classifier is trained using the concatenated feature vectors of S. The polarity of the reviews in T can be predicted by applying the classifier on the concatenated feature vectors of T.

4 Experiments

4.1 Dataset

We use the multilingual multi-domain Amazon review dataset[2] created by (Prettenhofer and Stein, 2010). It contains three different domains book, DVD and music. Each domain has reviews in four different languages English, German, French and Japanese. In our experiments, we use English as the source language and the rest three as target languages. Therefore, we have a total of nine tasks with different combinations of three domains and three target languages. For each task, the training and test datasets have 1000 positive reviews and 1000 negative reviews. There are also several thousand of unlabeled reviews but the quantity of them varies significantly for different tasks. Following (Prettenhofer and Stein, 2010), when there are more than 50000 unlabeled reviews we randomly selected 50000 of them, otherwise we use all the unlabeled reviews. The detailed statistics of the dataset are shown in Table 1.

We translated the 2000 training reviews and 2000 test reviews into the other languages using Google Translate. Prettenhofer and Stein (2010) has already provided the translation of the test data. We only need to translate the English training data into the three target languages. All the review texts are tokenized and converted into lowercase. We use Mecab[3] to segment the Japanese reviews.

4.2 Implementation

In the bilingual representation learning algorithm, we set the vector size as 200 and the context windows as 10. The learning rate is set to 0.025 fol-

[2] https://www.uni-weimar.de/medien/webis/corpora/corpus-webis-cls-10/

[3] http://taku910.github.io/mecab/

Target Language	Domain	MT-BOW	MT-PV	CL-SCL	BSE	CR-RL	Bi-PV	BiDRL
German	book	79.68	79.90	79.50	80.27	79.89	79.51	84.14
	DVD	77.92	80.09	76.92	77.16	77.14	78.60	84.05
	music	77.22	80.71	77.79	77.98	77.27	82.45	84.67
French	book	80.76	80.14	78.49	-	78.25	84.25	84.39
	DVD	78.83	81.49	78.80	-	74.83	79.60	83.60
	music	75.78	81.92	77.92	-	78.71	80.09	82.52
Japanese	book	70.22	67.45	73.09	70.75	71.11	71.75	73.15
	DVD	71.30	68.86	71.07	74.96	73.12	75.40	76.78
	music	72.02	74.53	75.11	77.06	74.38	75.45	78.77
Average Accuracy		75.97	77.23	76.52	74.26	76.08	78.57	81.34

Table 2: Cross-lingual sentiment classification accuracy for the nine tasks. For all the methods, we get ten different runs of the algorithm and calculate the mean accuracy.

Target Language	Domain	$\|S_U\|$	$\|T_U\|$
German	book	50000	50000
	DVD	30000	50000
	music	25000	50000
French	book	50000	32000
	DVD	30000	9000
	music	25000	16000
Japanese	book	50000	50000
	DVD	30000	50000
	music	25000	50000

Table 1: The amount of unlabeled reviews used in the experiments. There are also 1000 positive and 1000 negative reviews both for training and test in each task, i.e. $|S| = |T| = 2000$.

lowing `word2vec` and it declines with the training procedure. K is empirically chosen as 10. The algorithm runs 10 iterations on the dataset.

4.3 Baseline

We introduce several state-of-the-art methods used for comparison in our experiment as follows.

MT-BOW: It learns a classifier in the source language using bag-of-words features and the test data is translated into the source language via a Google Translate. We directly use the experimental results reported in (Prettenhofer and Stein, 2010).

MT-PV: We translate the training data into the target language and also translate the test data into the source language. In both the source and target languages, we use the paragraph vector model to learn the vector representation of the documents. Therefore, each document can be represented by the concatenation of the vector in two languages. A logistic regression classifier is trained using the concatenated feature vectors similarly to BiDRL. MT-PV can be regarded as a simplified version of BiDRL without the L_2, L_3 and L_4 regularizers.

CL-SCL: It is the cross-lingual structural correspondence learning algorithm proposed by (Prettenhofer and Stein, 2010). It learns a map between the bag-of-words representations in the source and the target languages. It also leverages Google Translate to obtain the word translation oracle.

BSE: It is the bilingual embedding method of (Tang et al., 2012). It aims to learn two different mapping matrices for the source and target languages. The two matrices map the bag-of-words representations in the source and the target languages into the same feature space. Tang et al. (2012) only report their results on 6 of the 9 tasks.

CR-RL: It is the bilingual word representation learning method of (Xiao and Guo, 2013). It learns different representations for words in different languages. Part of the word vector is shared among different languages and the rest is language-dependent. The document representation is calculated by taking average over all words in the document.

Bi-PV: Pham et al. (2015) extended the paragraph vector model into bilingual setting by sharing the document representation of a pair of parallel documents. Their method requires large amounts of parallel data and does not need the machine translation service during test phase. In our setting, there are not enough parallel data to train the model and it will lead to an unfair comparison without using the machine translated text. We implement a variant of their method which learns the

vector representation for the training and test data using both the original and the translated texts. Each pair of parallel documents shares the same document representation.

We also implement the method of (Zhou et al., 2015) which is originally designed for the English-Chinese cross-lingual sentiment classification task. We find that it is not very adaptable in our case because the negation pattern and sentiment words are hard to choose for our target languages. The results of our replication do not achieve comparable results with the rest methods and are not listed here to avoid misleading the readers.

4.4 Results and Analysis

Table 2 shows the experimental results for all the baselines and our method. For all the nine tasks, our bilingual document representation learning method achieves the state-of-the-art performance. The two most simple approaches MT-BOW implemented by (Prettenhofer and Stein, 2010) and MT-PV implemented by us are both strong baselines. They achieve comparable results with the more complex baselines on many tasks. MT-PV performs better than MT-BOW on most tasks which proves that the representation learning method is more useful than the traditional bag-of-words features.

The three word-based representation learning methods CL-SCL, BSE and CR-RL achieve similar results with the simple model MT-BOW and only outperform it on some tasks. However, the document representation learning methods MT-PV, Bi-PV and BiDRL performs much better. It shows that capturing the compositionality of words is important for sentiment classification. The isolated word representations are not enough to model the whole document. The Bi-PV model outperforms MT-PV on most tasks and shows that the authors idea of learning a single representation for a pair of parallel documents is more useful than learning them separately.

For all the baselines and our method, the performance of the English-Japanese tasks is lower than that of the English-German and English-French tasks. It is reasonable because the English language is much closer to German and French than Japanese. The machine translation tool also performs better when translating between the Western languages.

Our BiDRL model outperforms all the existing methods on all the tasks. The accuracy is over 80% on all the six tasks for the two European target languages. The mean accuracy of the nine tasks shows a significant gap between BiRDL and the existing models. It achieves an improvement about 3% compared to the previous state-of-the-art methods.

4.5 Parameter Sensitivity Study

In this subsection, we investigate the influence of the vector size of our representation learning algorithm. We conduct the experiments by changing the vector size from 50 to 400. For each parameter setting, we run the algorithm for ten times and get the mean accuracy.

The results of MT-PV and BiDRL on all the nine tasks are shown in Figure 3. For almost all the tasks, we can observe that our model BiDRL steadily outperforms the strong baseline MT-PV. It proves the efficacy of our bilingual embedding constraints.

For most of the nine tasks including DE-DVD, DE-MUSIC, FR-DVD, FR-MUSIC and JP-MUSIC, the performance of BiDRL increases with the growth of the vector size at the beginning and remains stable afterwards. For the rest tasks, our model responds less sensitively to the change of the vector size and the prediction accuracy keeps steady. However, the results of MT-PV show no regular patterns with the change of the vector size which makes it hard to choose a satisfying parameter value.

The parameter K is empirically chosen as 10 because we find that its value has little influence to our model when it is chosen between 10 and 50. Selecting a small K will help to accelerate the training procedure.

4.6 Analysis of the Sentiment Information

The traditional paragraph vector model only models the semantic relatedness between texts via the word co-occurrence statistics. In this study, we propose to learn the bilingual representation utilizing the sentiment information. Firstly, we introduce a classification hyperplane to separate the embedding of texts with different polarities, i.e the loss function L_2. Secondly, we consider the texts with the same sentiment but has largely different textual expressions. They are forced the have similar representations, i.e. L_4. Table 3 shows the

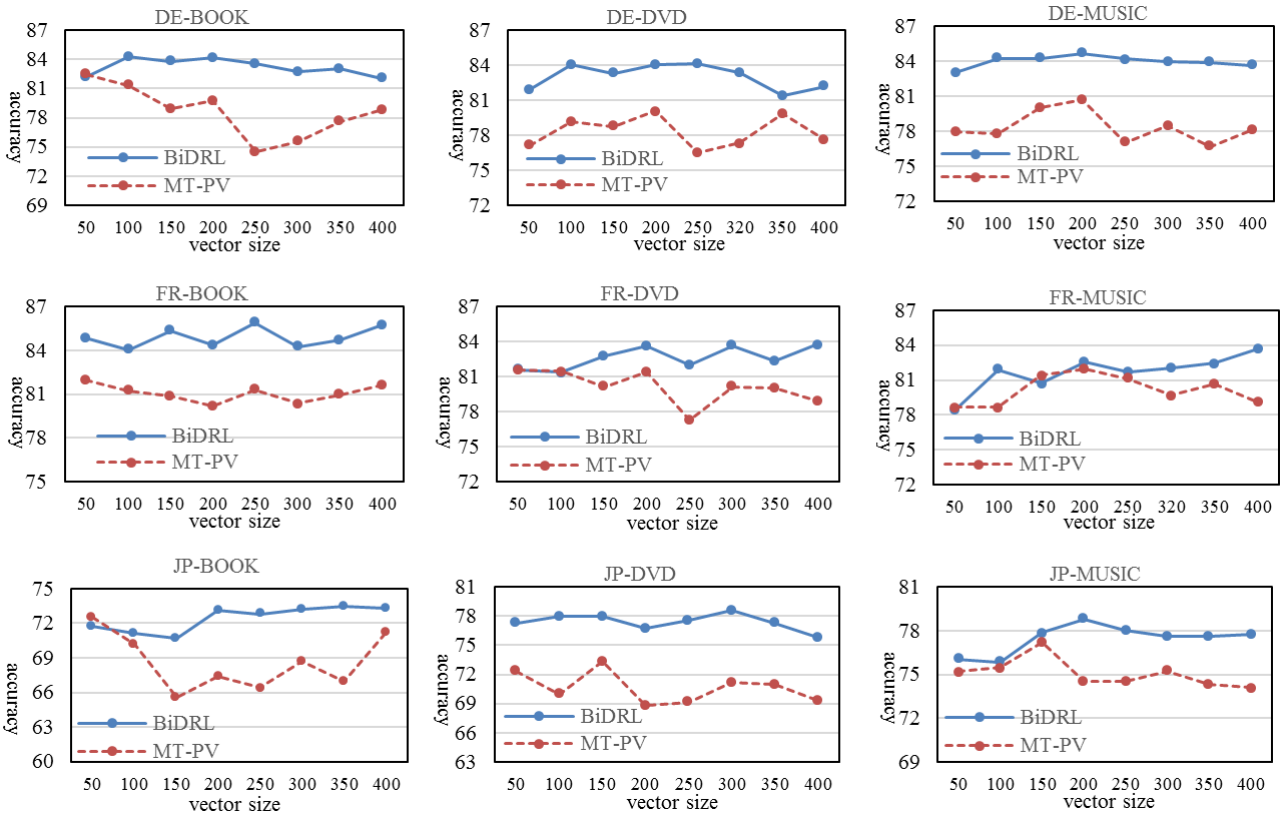


Figure 3: Influence of vector size for the nine cross-lingual sentiment classification tasks

results of our model without using these sentiment information.

Model	MT-PV	BiDRL-L_2	BiDRL-L_4	BiDRL
Accuracy	77.23	79.51	80.43	81.34

Table 3: Influence of the sentiment information. We only show the mean accuracy of the nine tasks due to space limit.

MT-PV can be regarded as BiDRL without all the sentiment information. It achieves lower results than the other three methods. We can also observe that removing L_2 or L_4 both decreases the accuracy. It proves that the sentiment information helps BiDRL to achieve better results.

5 Conclusion and Future Work

In this study, we propose a bilingual document representation learning method for cross-lingual sentiment classification. Different from previous studies which only get bilingual word embeddings, we directly learn the vector representation for documents in different languages. We propose three strategies to achieve a consistent embedding space for the source and target languages. Both sentiment and semantic correlations are exploited in our algorithm while previous works only use the semantic relatedness between parallel documents. Our model is evaluated on a benchmarking dataset which contains three different target languages and three different domains. Several state-of-the-art methods including several bilingual representation learning models are used for comparison. Our algorithm outperforms all the baseline methods on all the nine tasks in the experiment.

Our future work will focus on extending the bilingual document representation model into the multilingual scenario. We will try to learn a single embedding space for a source language and multiple target languages simultaneously. In addition, we will also explore the possibility of using more complex neural network models such as convolutional neural network and recurrent neural network to build bilingual document representation system.

Acknowledgments

The work was supported by National Natural Science Foundation of China (61331011), National Hi-Tech Research and Development Pro-

gram (863 Program) of China (2015AA015403, 2014AA015102) and IBM Global Faculty Award Program. We thank the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

AR Balamurali. 2012. Cross-lingual sentiment analysis for indian languages using linked wordnets. In *Proceedings of 2012 International Conference on Computational Linguistics (COLING)*, pages 73–82.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2010. Multilingual subjectivity: Are more languages better? In *Proceedings of the 23rd international conference on computational linguistics*, pages 28–36. Association for Computational Linguistics.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C. Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Advances in Neural Information Processing Systems*, pages 1853–1861.

Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th international conference on Computational Linguistics*, page 841. Association for Computational Linguistics.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 748–756.

Karl Moritz Hermann and Phil Blunsom. 2015. Multilingual models for compositional distributed semantics. In *Proceedings of 52rd Annual Meeting of the Association for Computational Linguistic*, pages 58–68.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proceedings of COLING-2012*, pages 1459–1474.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.

Bin Lu, Chenhao Tan, Claire Cardie, and Benjamin K. Tsou. 2011. Joint bilingual sentiment classification with unlabeled parallel corpora. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 320–330. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159.

Xinfan Meng, Furu Wei, Xiaohua Liu, Ming Zhou, Ge Xu, and Houfeng Wang. 2012. Cross-lingual mixture model for sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 572–581. Association for Computational Linguistics.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. 2007. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of ACL-2007*, pages 976–983.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Hieu Pham, Minh-Thang Luong, and Christopher D. Manning. 2015. Learning distributed representations for multilingual text sequences. In *Proceedings of NAACL-HLT*, pages 88–94.

Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1118–1127. Association for Computational Linguistics.

Jie Tang, Yuan Zhang, Jimeng Sun, Jinghai Rao, Wenjing Yu, Yiran Chen, and Alvis Cheuk M. Fong. 2012. Quantitative study of individual emotional states in social networks. *Affective Computing, IEEE Transactions on*, 3(2):132–144.

Andranik Tumasjan, Timm Oliver Sprenger, Philipp G. Sandner, and Isabell M. Welpe. 2010. Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10:178–185.

Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 235–243. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2013. Semi-supervised representation learning for cross-lingual text classification. In *Proceedings of EMNLP-2013*, pages 1465–1475. Association for Computational Linguistics.

Min Xiao and Yuhong Guo. 2014. Semi-supervised matrix completion for cross-lingual text classification. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1607–1614.

Huiwei Zhou, Long Chen, Fulin Shi, and Degen Huang. 2015. Learning bilingual sentiment word embeddings for cross-language sentiment classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 433–440. Association for Computational Linguistics.

Will Y. Zou, Richard Socher, Daniel M. Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *EMNLP*, pages 1393–1398.

Segment-Level Sequence Modeling using Gated Recursive Semi-Markov Conditional Random Fields

Jingwei Zhuo[1,2] *, **Yong Cao**[2], **Jun Zhu**[1] †, **Bo Zhang**[1], **Zaiqing Nie**[2]
[1]Dept. of Comp. Sci. & Tech., State Key Lab of Intell. Tech. & Sys., TNList Lab,
Tsinghua University, Beijing, 100084, China
[2]Microsoft Research, Beijing, 100084, China
{zjw15@mails, dcszj@mail, dcszb@mail}.tsinghua.edu.cn; {yongc, znie}@microsoft.com

Abstract

Most of the sequence tagging tasks in natural language processing require to recognize segments with certain syntactic role or semantic meaning in a sentence. They are usually tackled with Conditional Random Fields (CRFs), which do indirect word-level modeling over word-level features and thus cannot make full use of segment-level information. Semi-Markov Conditional Random Fields (Semi-CRFs) model segments directly but extracting segment-level features for Semi-CRFs is still a very challenging problem. This paper presents Gated Recursive Semi-CRFs (grSemi-CRFs), which model segments directly and automatically learn segment-level features through a gated recursive convolutional neural network. Our experiments on text chunking and named entity recognition (NER) demonstrate that grSemi-CRFs generally outperform other neural models.

1 Introduction

Most of the sequence tagging tasks in natural language processing (NLP) are segment-level tasks, such as text chunking and named entity recognition (NER), which require to recognize segments (i.e., a set of continuous words) with certain syntactic role or semantic meaning in a sentence. These tasks are usually tackled with Conditional Random Fields (CRFs) (Lafferty et al., 2001), which do word-level modeling as putting each word a tag, by using some predefined tagging schemes, e.g., the "IOB" scheme (Ramshaw and Marcus, 1995). Such tagging schemes are lossy transformations of original segment tags: They do indicate the boundary of adjacent segments but lose the length information of segments to some extent. Besides, CRFs can only employ word-level features, which are either hand-crafted or extracted with deep neural networks, such as window-based neural networks (Collobert et al., 2011) and bidirectional Long Short-Term Memory networks (BI-LSTMs) (Huang et al., 2015). Therefore, CRFs cannot make full use of segment-level information, such as inner properties of segments, which cannot be fully encoded in word-level features.

Semi-Markov Conditional Random Fields (Semi-CRFs) (Sarawagi and Cohen, 2004) are proposed to model segments directly and thus readily utilize segment-level features that encode useful segment information. Existing work has shown that Semi-CRFs outperform CRFs on segment-level tagging tasks such as sequence segmentation (Andrew, 2006), NER (Sarawagi and Cohen, 2004; Okanohara et al., 2006), web data extraction (Zhu et al., 2007) and opinion extraction (Yang and Cardie, 2012). However, Semi-CRFs need many more features compared to CRFs as they need to model segments with different lengths. As manually designing the features is tedious and often incomplete, *how to automatically extract good features* becomes a very important problem for Semi-CRFs. A naive solution that builds multiple feature extractors, each of which extracts features for segments with a specific length, is apparently time-consuming. Moreover, some of these separate extractors may underfit as the segments with specific length may be very rare in the training data. By far, Semi-CRFs are lacking of an automatic segment-level feature extractor.

In this paper, we fill the research void by

* This work was done when J.W.Z was on an internship with Microsoft Research.

† J.Z is the corresponding author.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1413–1423,
Berlin, Germany, August 7-12, 2016.

proposing Gated Recursive Semi-Markov Conditional Random Fields (grSemi-CRFs), which can automatically learn features for segment-level sequence tagging tasks. Unlike previous approaches which usually use a neural-based feature extractor with a CRF layer, a grSemi-CRF consists of a gated recursive convolutional neural network (grConv) (Cho et al., 2014) with a Semi-CRF layer. The grConv is a variant of recursive neural networks. It builds a pyramid-like structure to extract segment-level features in a hierarchical way. This feature hierarchy well matches the intuition that long segments are combinations of their short sub-segments. This idea was first explored in Cho et al. (2014) to build an encoder in neural machine translation and then extended to solve other problems, such as sentence-level classification (Zhao et al., 2015) and Chinese word segmentation (Chen et al., 2015).

The advantages of grSemi-CRFs are two folds. First, thanks to the pyramid architecture of grConvs, grSemi-CRFs can extract all the segment-level features using one single feature extractor, and there is no underfitting problem as all parameters of the feature extractor are shared globally. Besides, unlike recurrent neural network (RNN) models, the training and inference of grSemi-CRFs are very fast as there is no time dependency and all the computations can be done in parallel. Second, thanks to the semi-Markov structure of Semi-CRFs, grSemi-CRFs can model segments in sentences directly without the need to introduce extra tagging schemes, which solves the problem that segment length information cannot be fully encoded in tags. Besides, grSemi-CRFs can also utilize segment-level features which can flexibly encode segment-level information such as inner properties of segments, compared to word-level features as used in CRFs. By combining grConvs with Semi-CRFs, we propose a new way to automatically extract segment-level features for Semi-CRFs.

Our major contributions can be summarized as:

(1) We propose grSemi-CRFs, which solve both the automatic feature extraction problem for Semi-CRFs and the indirect word-level modeling problem in CRFs. As a result, grSemi-CRFs can do segment-level modeling directly and make full use of segment-level features;

(2) We evaluate grSemi-CRFs on two segment-level sequence tagging tasks, text chunking and NER. Experimental results show the effectiveness of our model.

2 Preliminary

In sequence tagging tasks, given a word sequence, the goal is to assign each word (e.g., in POS Tagging) or each segment (e.g., in text chunking and NER) a tag. By leveraging a tagging scheme like "IOB", all the tasks can be regarded as word-level tagging. More formally, let $\mathcal{X}$ denote the set of words and $\mathcal{Y}$ denote the set of tags. A word sentence with length T can be denoted by $\mathbf{x} = (x_1, ..., x_T)$ and its corresponding tags can be denoted as $\mathbf{y} = (y_1, ..., y_T)$. A CRF (Lafferty et al., 2001) defines a conditional distribution

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{t=1}^{T} F(y_t, \mathbf{x}) + A(y_{t-1}, y_t)\right), \quad (1)$$

where $F(y_t, \mathbf{x})$ is the tag score (or potential) for tag y_t at position t, $A(y_{t-1}, y_t)$ is the transition score between y_{t-1} and y_t to measure the tag dependencies of adjacent words and $Z(\mathbf{x}) = \sum_{\mathbf{y}'} \exp\left(\sum_{t=1}^{T} F(y'_t, \mathbf{x}) + A(y_{t-1}, y'_t)\right)$ is the normalization factor. For the common log-linear models, $F(y_t, \mathbf{x})$ can be computed by

$$F(y_t, \mathbf{x}) = \mathbf{v}_{y_t}^{\mathrm{T}} \mathbf{f}(y_t, \mathbf{x}) + b_{y_t}, \quad (2)$$

where $\mathrm{V} = (\mathbf{v}_1, ..., \mathbf{v}_{|\mathcal{Y}|})^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{Y}| \times D}$, $\mathbf{b}_{\mathrm{V}} = (b_1, ..., b_{|\mathcal{Y}|})^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{Y}|}$, $f(y_t, \mathbf{x}) \in \mathbb{R}^D$ are the word-level features for y_t over the sentence $\mathbf{x}$ and D is the number of features. $f(y_t, \mathbf{x})$ can be manually designed or automatically extracted using neural networks, such as window-based neural networks (Collobert et al., 2011).

If we consider segment-level tagging directly[1], we get a segmentation of the previous tag sentence. With a little abuse of notation, we denote a segmentation by $\mathbf{s} = (s_1, ..., s_{|\mathbf{s}|})$ in which the jth segment $s_j = \langle h_j, d_j, y_j \rangle$ consists of a start position h_j, a length $d_j < L$ where L is a predefined upperbound and a tag y_j. Conceptually, s_j means a tag y_j is given to words $(x_{h_j}, ..., x_{h_j + d_j - 1})$. A Semi-CRF (Sarawagi and Cohen, 2004) defines a conditional distribution

$$p(\mathbf{s}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\left(\sum_{j=1}^{|\mathbf{s}|} F(s_j, \mathbf{x}) + A(y_{j-1}, y_j)\right), \quad (3)$$

[1] Word-level tagging can be regarded as segment-level tagging over length-1 segments.

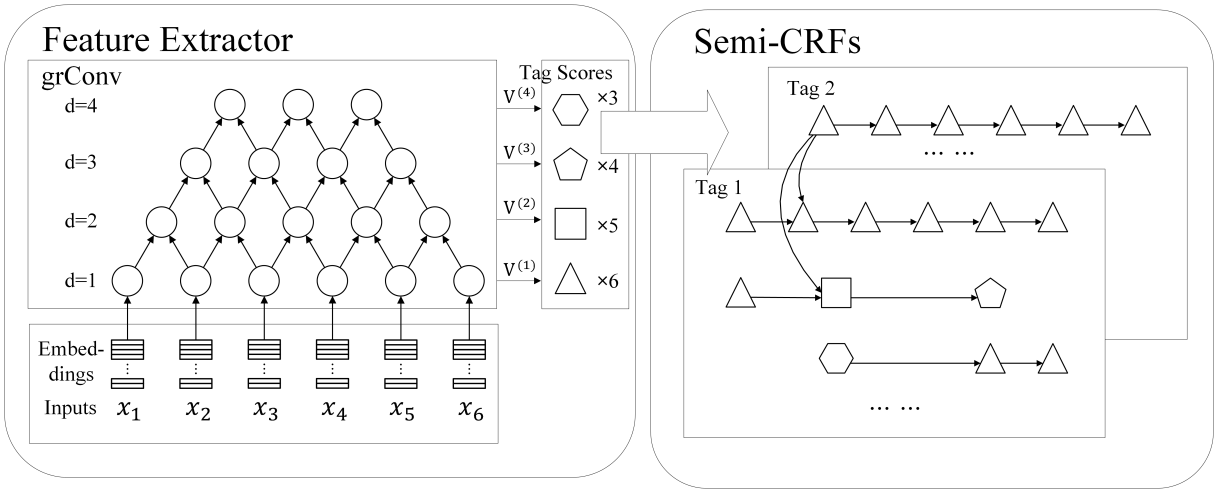


Figure 1: An overview of grSemi-CRFs. For simplicity, we set the segment length upperbound $L = 4$, and the sentence length $T = 6$. The left side is the feature extractor, in which each node denotes a vector of segment-level features (e.g., $\mathbf{z}_k^{(d)}$ for the kth node in the dth layer). Embeddings of word-level input features are used as length-1 segment-level features, and the length-d feature is extracted from two adjacent length-$(d-1)$ features. The right side is the Semi-CRF. Tag score vectors are computed as linear transformations of segment-level features and the number of them equals the number of nodes in the same layer. For clarity, we use triangle, square, pentagon and hexagon to denote the tag score vectors for length-$1, 2, 3, 4$ segments and directed links to denote the tag transformations of adjacent segments.

where $F(s_j, \mathbf{x})$ is the potential or tag score for segment s_j, $A(y_{j-1}, y_j)$ is the transition score to measure tag dependencies of adjacent segments and $Z(\mathbf{x}) = \sum_{\mathbf{s}'} \exp\left(\sum_{j=1}^{|\mathbf{s}'|} F(s'_j, \mathbf{x}) + A_{y'_{j-1}, y'_j}\right)$ is the normalization factor. For the common log-linear models, $F(s_j, \mathbf{x})$ can be computed by

$$F(s_j, \mathbf{x}) = \mathbf{v}_{y_j}^{\mathrm{T}} \mathbf{f}(s_j, \mathbf{x}) + b_{y_j}, \tag{4}$$

where $\mathrm{V} = (\mathbf{v}_1, ..., \mathbf{v}_{|\mathcal{Y}|})^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{Y}| \times D}$, $\mathbf{b}_{\mathrm{V}} = (b_1, ..., b_{|\mathcal{Y}|})^{\mathrm{T}} \in \mathbb{R}^{|\mathcal{Y}|}$ and $f(s_j, \mathbf{x}) \in \mathbb{R}^D$ are the segment-level features for s_j over the sentence $\mathbf{x}$.

As Eq. (1) and Eq. (3) show, CRFs can be regarded as a special case of Semi-CRFs when $L = 1$. CRFs need features for only length-1 segments (i.e., words), while Semi-CRFs need features for length-ℓ segments ($1 \leq \ell \leq L$). Therefore, to model the same sentence, Semi-CRFs generally need many more features than CRFs, especially when L is large. Besides, unlike word-level features used in CRFs, the sources of segment-level features are often quite limited. In existing work, the sources of $f(s_j, \mathbf{x})$ can be roughly divided into two parts: (1) Concatenations of word-level features (Sarawagi and Cohen, 2004; Okanohara et al., 2006); and (2) Hand-crafted segment-level features, including task-insensitive features, like the length of segments, and task-specific features, like the verb phrase patterns in opinion extraction (Yang and Cardie, 2012). As manually designing features is time-consuming and often hard to capture rich statistics underlying the data, how to automatically extract features for Semi-CRFs remains a challenge.

3 Gated Recursive Semi-Markov CRFs

In this section, we present Gated Recursive Semi-CRFs (grSemi-CRFs), which inherit the advantages of Semi-CRFs in segment-level modeling, and also solve the feature extraction problem of Semi-CRFs by introducing a gated recursive convolutional neural network (grConv) as the feature extractor. Instead of building multiple feature extractors at different scales of segment lengths, grSemi-CRFs can extract features with any length by using a single grConv, and learn the parameters effectively via sharing statistics.

3.1 Architecture

The architecture of grSemi-CRFs is illustrated in Figure 1. A grSemi-CRF can be divided into two parts, a feature extractor (i.e., grConv) and a Semi-CRF. Below, we explain each part in turn.

As is shown, the feature extractor is a pyramid-like directed acyclic graph (DAG), in which nodes

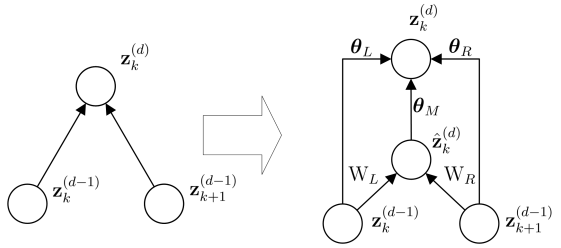


Figure 2: The the building block of the feature extractor (i.e., grConv), in which parameters are shared among the pyramid structure. We omit the dependency of $\boldsymbol{\theta}_L, \boldsymbol{\theta}_R, \boldsymbol{\theta}_M$ on $\mathrm{G}_L, \mathrm{G}_R$.

are stacked layer by layer and information is propagated from adjacent nodes in the same layer to their co-descendants in the higher layer through directed links. Recall that L denotes the upper-bound of segment length (i.e., the height of the grConv), we regard the bottom level as the 1st level and the top level as the Lth level. Then, for a length-T sentence, the dth level will have $T-d+1$ nodes, which correspond to features for $T-d+1$ length-d segments. The kth node in the dth layer corresponds to the segment-level latent features $\mathbf{z}_k^{(d)} \in \mathbb{R}^D$, which denote the meaning of the segment, e.g., the syntactic role (i.e., for text chunking) or semantic meaning (i.e., for NER).

Like CRFs, grSemi-CRFs allow word-level categorical inputs (i.e., $\mathbf{x}_k$) which are transformed into continuous vectors (i.e., embeddings) according to look-up tables and then used as length-1 segment-level features (i.e., $\mathbf{z}_k^{(1)}$). To be clear, we call these inputs as input features and those extracted segment-level features (i.e., $\mathbf{z}_k^{(d)}$) as segment-level latent features. Besides, grSemi-CRFs also allow segment-level input features (e.g., gazetteers) directly as shown in Eq. (12). We will discuss more details in section 4.3.

The building block of the feature extractor is shown in Figure 2, where an intermediate node $\hat{\mathbf{z}}_{(d)} \in \mathbb{R}^D$ is introduced to represent the interactions of two length-$(d-1)$ segments. To capture such complex interactions, $\hat{\mathbf{z}}_k^{(d)}$ is computed through a non-linear transformation, i.e.,

$$\hat{\mathbf{z}}_k^{(d)} = g(\boldsymbol{\alpha}_k^{(d)}) = g(\mathrm{W}_L \mathbf{z}_k^{(d-1)} + \mathrm{W}_R \mathbf{z}_{k+1}^{(d-1)} + \mathbf{b}_{\mathrm{W}}), \quad (5)$$

where $\mathrm{W}_L, \mathrm{W}_R \in \mathbb{R}^{D \times D}$ and $\mathbf{b}_{\mathrm{W}} \in \mathbb{R}^D$ are shared globally, and $g(\cdot)$ is a non-linear activation function[2].

[2]We use a modified version of the sigmoid function, i.e., $g(x) = 4\left(\frac{1}{1+e^{-x}} - \frac{1}{2}\right)$.

Then, the length-d segment-level latent features $\mathbf{z}_k^{(d)}$ can be computed as

$$\mathbf{z}_k^{(d)} = \theta_L \mathbf{z}_k^{(d-1)} + \theta_R \mathbf{z}_{k+1}^{(d-1)} + \theta_M \hat{\mathbf{z}}_k^{(d)}, \quad (6)$$

where θ_L, θ_M and $\theta_R \in \mathbb{R}$ are the gating coefficients which satisfy the condition $\theta_L, \theta_R, \theta_M \geq 0$ and $\theta_L + \theta_R + \theta_M = 1$. Here, we make a little modification of grConvs by making the gating coefficients as vectors instead of scalars, i.e.,

$$\mathbf{z}_k^{(d)} = \boldsymbol{\theta}_L \circ \mathbf{z}_k^{(d-1)} + \boldsymbol{\theta}_R \circ \mathbf{z}_{k+1}^{(d-1)} + \boldsymbol{\theta}_M \circ \hat{\mathbf{z}}_k^{(d)}, \quad (7)$$

where $\circ$ denotes the element-wise product and $\boldsymbol{\theta}_L$, $\boldsymbol{\theta}_R$ and $\boldsymbol{\theta}_M \in \mathbb{R}^D$ are vectorial gating coefficients[3] which satisfy the condition that $\theta_{L,i}, \theta_{R,i}, \theta_{M,i} \geq 0$ and $\theta_{L,i} + \theta_{R,i} + \theta_{M,i} = 1$ for $1 \leq i \leq D$. There are two reasons for this modification: (1) Theoretically, the element-wise combination makes a detailed modeling as each feature in $\mathbf{z}_k^{(d)}$ may have its own combining; and (2) Experimentally, this setting makes our grSemi-CRF[4] more flexible, which increases its generalizability and leads to better performance in experiments as shown in Table 4.

We can regard Eq. (7) as a soft gate function to control the propagation flows. Besides, all the parameters (i.e., $\mathrm{W}_L, \mathrm{W}_R, \mathbf{b}_{\mathrm{W}}, \mathrm{G}_L, \mathrm{G}_R, \mathbf{b}_{\mathrm{G}}$) are shared globally and recursively applied to the input sentence in a bottom-up manner. All of these account for the name *gated recursive convolutional neural networks* (grConvs).

Eq. (5) and Eq. (7) build the information propagation criteria in a grConv. The basic assumption behind Eq. (5) and Eq. (7) is that the meaning of one segment can be represented as a linear combination of three parts: (1) the meaning of its prefix segment, (2) the meaning of its suffix segment and (3) the joint meaning of both (i.e., the complex interaction). This process matches our intuition about the hierarchical structure in the composition of a sentence. For example, the meaning of *the United States* depends on the suffix segment *United States*, whose meaning is not only from its prefix *United* or suffix *States*, but the interaction of both.

The vectorial gating coefficients $\boldsymbol{\theta}_L$, $\boldsymbol{\theta}_R$ and $\boldsymbol{\theta}_M$

[3]We omit the dependency of $\boldsymbol{\theta}_L, \boldsymbol{\theta}_R$ and $\boldsymbol{\theta}_M$ on d and k for notation simplicity.

[4]Unless otherwise stated, we regard "grSemi-CRF" as grSemi-CRF with vectorial gating coefficients in default.

are computed adaptively, i.e.,

$$\begin{pmatrix}\boldsymbol{\theta}_L\\ \boldsymbol{\theta}_R\\ \boldsymbol{\theta}_M\end{pmatrix} = \begin{pmatrix}1/\mathbf{Z}\\ 1/\mathbf{Z}\\ 1/\mathbf{Z}\end{pmatrix} \circ \exp\left(\mathrm{G}_L\mathbf{z}_k^{(d-1)} + \mathrm{G}_R\mathbf{z}_{k+1}^{(d-1)} + \mathbf{b}_\mathrm{G}\right), \tag{8}$$

where $\mathrm{G}_L, \mathrm{G}_R \in \mathbb{R}^{3D\times D}$ and $\mathbf{b}_\mathrm{G} \in \mathbb{R}^{3D}$ are shared globally. $\mathbf{Z} \in \mathbb{R}^d$ is normalization coefficients and the ith element of $\mathbf{Z}$ is computed via

$$Z_i = \sum_{j=1}^{3}\left[\exp\left(\mathrm{G}_L\mathbf{z}_k^{(d-1)} + \mathrm{G}_R\mathbf{z}_{k+1}^{(d-1)} + \mathbf{b}_\mathrm{G}\right)\right]_{D\times(j-1)+i}. \tag{9}$$

After the forward propagation of the feature extractor is over, the tag scores (i.e., the potential functions for Semi-CRFs) are computed through a linear transformation. For segment $s_j = \langle h_j, d_j, y_j\rangle$, its latent feature is $f(s_j, \mathbf{x}) = \mathbf{z}_{h_j}^{(d_j)}$ and corresponding potential/tag score is

$$F(\mathbf{s}_j;\mathbf{x}) = f(\langle h_j, d_j, y_j\rangle;\mathbf{x}) = \left[\mathrm{V}_0^{(d_j)}\mathbf{z}_{h_j}^{(d_j)} + \mathbf{b}_\mathrm{V}^{(d_j)}\right]_{y_j}, \tag{10}$$

where $\mathrm{V}_0^{(d_j)} \in \mathbb{R}^{|\mathcal{Y}|\times D}$ and $\mathbf{b}^{(d_j)} \in \mathbb{R}^{|\mathcal{Y}|}$ are parameters for length-d_j segments. To encode contextual information, we can assume that the tag of a segment depends not only on itself but also its neighbouring segments with the same length, i.e.,

$$F(\mathbf{s}_j;\mathbf{x}) = \left[\sum_{i=-H}^{H}\mathrm{V}_i^{(d_j)}\mathbf{z}_{h_j+i}^{(d_j)} + \mathbf{b}_\mathrm{V}^{(d_j)}\right]_{y_j}, \tag{11}$$

where $\mathrm{V}_{-H}^{(d_j)}, ..., \mathrm{V}_0^{(d_j)}, ..., \mathrm{V}_H^{(d_j)} \in \mathbb{R}^{|\mathcal{Y}|\times D}$ and H is the window width for neighbouring segments. Apart from the automatically extracted segment-level latent features $\mathbf{z}_k^{(d)}$, grSemi-CRFs also allow segment-level input features (e.g., gazetteers), i.e.,

$$F(\mathbf{s}_j;\mathbf{x}) = \left[\sum_{i=-H}^{H}\mathrm{V}_i^{(d_j)}\mathbf{z}_{h_j+i}^{(d_j)} + \mathbf{b}_\mathrm{V}^{(d_j)} + \mathrm{U}^{(d_j)}\mathbf{c}_{h_j}^{(d_j)}\right]_{y_j}, \tag{12}$$

where $\mathrm{U}^{(d_j)} \in \mathbb{R}^{|\mathcal{Y}|\times D'}$ and $\mathbf{c}_{h_j}^{(d_j)} \in \mathbb{R}^{D'}$ is a vector of segment-level input features.

Then, we can use Eq. (3) for inference by using a Semi-CRF version of Viterbi algorithms (Sarawagi and Cohen, 2004).

3.2 Learning of Parameters

To learn grSemi-CRFs, we maximize the log likelihood $\mathcal{L} = \log p(\mathbf{s}|\mathbf{x})$ over all the parameters. Here, for notation simplity, we consider the simpliest case, i.e., using Eq. (10) to compute tag scores. More details can be found in the supplementary note.

Gradients of Semi-CRF-based parameters (i.e., A and V_0) and tag scores $F(s_j, \mathbf{x})$ can be computed based on the marginal probability of neighbouring segments via a Semi-CRF version of forward-backward algorithms (Sarawagi and Cohen, 2004). As for the grConv-based parameters, we can compute their gradients by back propagation. For example, gradients for W_L and G_L are[5]

$$\frac{\partial\mathcal{L}}{\partial\mathrm{W}_L} = \sum_{d=1}^{L}\sum_{k=1}^{T-d+1}\frac{\partial\mathcal{L}}{\partial\mathbf{z}_k^{(d)}}\frac{\partial\mathbf{z}_k^{(d)}}{\partial\mathrm{W}_L}, \frac{\partial\mathcal{L}}{\partial\mathrm{G}_L} = \sum_{d=1}^{L}\sum_{k=1}^{T-d+1}\frac{\partial\mathcal{L}}{\partial\mathbf{z}_k^{(d)}}\frac{\partial\mathbf{z}_k^{(d)}}{\partial\mathrm{G}_L}, \tag{13}$$

where $\frac{\partial\mathbf{z}_k^{(d)}}{\partial\mathrm{W}_L}$ and $\frac{\partial\mathbf{z}_k^{(d)}}{\partial\mathrm{G}_L}$ can be derived from Eq. (5), Eq. (7) and Eq. (8). For $\frac{\partial\mathcal{L}}{\partial\mathbf{z}_k^{(d)}}$, thanks to the recursive structure, it can be computed as

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\mathbf{z}_k^{(d)}} =& \frac{\partial\mathbf{z}_k^{(d+1)}}{\partial\mathbf{z}_k^{(d)}}\frac{\partial\mathcal{L}}{\partial\mathbf{z}_k^{(d+1)}} + \frac{\partial\mathbf{z}_{k-1}^{(d+1)}}{\partial\mathbf{z}_k^{(d)}}\frac{\partial\mathcal{L}}{\partial\mathbf{z}_{k-1}^{(d+1)}}\\ &+ \mathrm{V}_0^{(d)\mathrm{T}}\frac{\partial\mathcal{L}}{\partial F(\mathbf{s}_k^{(d)},\mathbf{x})},\end{aligned} \tag{14}$$

where $\mathbf{s}_k^{(d)} = \langle k, d, \mathcal{Y}\rangle$ is a length-$|\mathcal{Y}|$ vector which denotes segments with all possible tags for $\mathbf{z}_k^{(d)}$, $\frac{\partial\mathcal{L}}{\partial F(\mathbf{s}_k^{(d)},\mathbf{x})}$ is the gradient for $F(\mathbf{s}_k^{(d)},\mathbf{x})$ and

$$\frac{\partial\mathbf{z}_k^{(d+1)}}{\partial\mathbf{z}_k^{(d)}} = \mathrm{diag}(\boldsymbol{\theta}_L) + \mathrm{diag}(\boldsymbol{\theta}_M \circ g'(\boldsymbol{\alpha}_k^{(d+1)}))\mathrm{W}_L, \tag{15}$$

where $\mathrm{diag}(\boldsymbol{\theta}_L)$ denotes the diagonal matrix spanned by vector $\boldsymbol{\theta}_L$, and $\frac{\partial\mathbf{z}_{k-1}^{(d+1)}}{\partial\mathbf{z}_k^{(d)}}$ has a similar form. As Eq. (14) shows, for each node in the feature extractor of grSemi-CRFs, its gradient consists of two parts: (1) the gradients back propagated from high layer nodes (i.e., longer segments); and (2) the supervising signals from Semi-CRFs. In other words, the supervision in the objective function is added to each node in grSemi-CRFs. This is a nice property compared to other neural-based feature extractors used in CRFs, in which only the nodes of several layers on the top receive supervision. Besides, the term $\mathrm{diag}(\boldsymbol{\theta}_L)$ in Eq. (15) prevents $\frac{\partial\mathbf{z}_k^{(d+1)}}{\partial\mathbf{z}_k^{(d)}}$ from being too small when $g'(\boldsymbol{\alpha}_k^{(d)})$ and W_L are small, which acts as the linear unit recurrent connection in the memory block of LSTM (Hochreiter and Schmidhuber, 1997; Zhao et al., 2015). All of these help in avoiding gradient vanishing problems in training grSemi-CRFs.

[5]Gradients for $\mathrm{W}_R, \mathbf{b}_\mathrm{W}, \mathrm{G}_R, \mathbf{b}_\mathrm{G}$ can be computed in similar ways.

4 Experiments

We evaluate grSemi-CRFs on two segment-level sequence tagging NLP tasks: text chunking and named entity recognition (NER).

4.1 Datasets

For text chunking, we use the CONLL 2000 text chunking shared dataset[6] (Tjong Kim Sang and Buchholz, 2000), in which the objective is to divide the whole sentence into different segments according to their syntactic roles, such as noun *phrases* ("NP"), *verb phrases* ("VP") and *adjective phrases* ("ADJP"). We call it a "segment-rich" tasks as the number of phrases are much higher than that of non-phrases which is tagged with *others* ("O"). We evaluate performance over all the chunks instead of only noun pharse (NP) chunks.

For NER, we use the CONLL 2003 named entity recognition shared dataset[7] (Tjong Kim Sang and De Meulder, 2003), in which segments are tagged with one of four entity types: *person* ("PER"), *location* ("LOC"), *organization* ("ORG") and *miscellaneous*("MISC"), or *others* ("O") which is used to denote non-entities. We call it a "segment-sparse" task as entities are rare while non-entities are common.

4.2 Input Features

For each word, we use multiple input features, including the word itself, its length-3 prefix and length-4 suffix, its capitalization pattern, its POS tag, the length-4,8,12,20 prefixs of its Brown clusters (Brown et al., 1992) and gazetteers[8]. All of them are used as word-level input features except gazetteers, which are used as segment-level features directly. All the embeddings for word-level inputs are randomly initialized except word embeddings, which can be initialized randomly or by pretraining over unlabeled data, which is external information compared to the dataset. Besides word embeddings, Brown clusters and gazetteers are also based on external information, as summarized below:

- **Word embeddings**. We use Senna embeddings[9] (Collobert et al., 2011), which are 50-dimensional and have been commonly used in sequence tagging tasks (Collobert et al., 2011; Turian et al., 2010; Huang et al., 2015);
- **Brown clusters**. We train two types of Brown clusters using the implementation from Liang (2005): (1) We follow the setups of Ratinov and Roth (2009), Turian et al. (2010) and Collobert et al. (2011) to generate 1000 Brown clusters on Reuters RCV1 dataset (Lewis et al., 2004); (2) We generate 1000 Brown clusters on New York Times (NYT) corpus (Sandhaus, 2008);
- **Gazetteers**. We build our gazetteers based on the gazetteers used in Senna (Collobert et al., 2011) and Wikipedia entries, mainly the locations and organizations. We also denoise our gazetteers by removing overlapped entities and using BBN Pronoun Coreference and Entity Type Corpus (Weischedel and Brunstein, 2005) as filters[10].

4.3 Implementation Details

To learn grSemi-CRFs, we employ Adagrad (Duchi et al., 2011), an adaptive stochastic gradient descent method which has been proved successful in similar tasks (Chen et al., 2015; Zhao et al., 2015). To avoid overfitting, we use the dropout strategy (Srivastava et al., 2014) and apply it on the first layer (i.e., $\mathbf{z}_k^{(0)}$). We also use the strategy of ensemble classifiers, which is proved an effective way to improve generalization performance (Collobert et al., 2011). All results are obtained by decoding over an average Semi-CRF after 10 training runs with randomly initialized parameters.

For the CONLL 2003 dataset, we use the F_1 scores on the development set to help choose the best-performed model in each run. For the CONLL 2000 dataset, as there is no development set provided, we use cross validation as Turian et al. (2010) to choose hyperparameters. After that, we retrain model according to the hyperparameters and choose the final model in each run.

Our hyperparameter settings for these two tasks are shown in Table 1. The segment length is set according to the maximum segment length in training set. We set the minibatch size to 10, which means that we process 10 sentences in a batch. The window width defines the parameter H in Eq. (12) when producing tag score vectors.

[6] Available at: http://www.cnts.ua.ac.be/conll2000/chunking/
[7] Available at: http://www.cnts.ua.ac.be/conll2003/ner/
[8] Among them, POS tags are provided in the dataset.
[9] Available at http://ronan.collobert.com/senna/
[10] We apply gazetteers on BBN corpus, collect lists of false positive entities and clean our gazetteers according to these lists.

Hyperparameters	CONLL 2000	CONLL 2003
Segment length	15	10
Dropout	0.3	0.3
Learning rate	0.3	0.3
Epochs	15	20
Minibatches	10	10
Window width	2	2

Table 1: Hyperparameter settings for our model.

4.4 Results and Analysis

Table 2 shows the results of our grSemi-CRFs and other models[11]. We divide other models into two categories, i.e., neural models and non-neural models, according to whether neural networks are used as automatic feature extractors. For neural models, Senna (Collobert et al., 2011) consists of a window-based neural network for feature extraction and a CRF for word-level modeling while BI-LSTM-CRF (Huang et al., 2015) uses a bidirectional Long Short-Term Memory network for feature extraction and a CRF for word-level modeling.

For non-neural models, JESS-CM (Suzuki and Isozaki, 2008) is a semi-supervised model which combines Hidden Markov Models (HMMs) with CRFs and uses 1 billion unlabelled words in training. Lin and Wu (2009) cluster 20 million phrases over corpus with around 700 billion tokens, and use the resulting clusters as features in CRFs. Passos et al. (2014) propose a novel word embedding method which incorporates gazetteers as supervising signals in pretraining and builds a log-linear CRF over them. Ratinov and Roth (2009) use CRFs based on many non-local features and 30 gazetteers extracted from Wikipedia and other websites with more than 1.5 million entities.

As Table 2 shows, grSemi-CRFs outperform other neural models, in both text chunking and named entity recognition (NER) tasks. BI-LSTM-CRFs use many more input features than ours, which accounts for the phenomenon that the performance of our grSemi-CRFs is rather mediocre (i.e., 93.92% versus 94.13% and 84.66% versus 84.26%) without external information. However, once using Senna embeddings, our grSemi-CRFs perform much better than BI-LSTM-CRFs.

For non-neural models, one similarity of them is that they use a lot of hand-crafted features, and many of them are even task-specific. Unlike them, grSemi-CRFs use much fewer input features and most of them are task-insensitive[13]. However, grSemi-CRFs achieve almost the same performance, sometimes even better. For text chunking, grSemi-CRF outperforms all reported supervised models, except JESS-CM (Suzuki and Isozaki, 2008), a semi-supervised model using giga-word scale unlabeled data in training[14]. However, the performance of our grSemi-CRF (95.01%) is very close to that of JESS-CM (95.15%). For NER, the performance of grSemi-CRFs are also very closed to state-of-the-art results (90.87% versus 90.90%).

[11]Because of the space limit, we only compare our model with other models which follow similar settings and achieve high performance.

Input Features	CONLL 2000	CONLL 2003
None	93.92	84.66
Brown(NYT)	94.18	86.57
Brown(RCV1)	94.05	**88.22**
Emb	**94.73**	88.12
Gaz	–	87.94
Emb + Brown(NYT)	**95.01**	88.86
Emb + Brown(RCV1)	94.87	89.44
Emb + Gaz	–	**89.88**
Brown(NYT) + Gaz	–	88.69
Brown(RCV1) + Gaz	–	89.82
All(NYT)	–	90.00
All(RCV1)	–	**90.87**

Table 3: Results of grSemi-CRF with external information, measured in F_1 score. None = no external information, Emb = Senna embeddings, Brown = Brown clusters, Gaz = gazetteers and All = Emb + Brown + Gaz. NYT and RCV1 in the parenthesis denote the corpus used to generate Brown clusters. "–" means no results. Notice that gazetteers are only applied to NER.

4.4.1 Impact of External Information

As Table 3 shows, external information improve the performance of grSemi-CRFs for both tasks. Compared to text chunking, we can find out that external information plays an extremely important role in NER, which coincides with the general idea that NER is a knowledge-intensive task (Ratinov and Roth, 2009). Another interesting thing is that, Brown clusters generated from NYT corpus performs better on the CONLL 2000 task while those generated from Reuters RCV1 dataset performs better on the CONLL 2003 task. The reason is

[13]E.g.: for NER, JESS-CM uses 79 different features; Lin and Wu (2009) use 48 baseline and phrase cluster features; while we only use 11. Besides, grSemi-CRFs use almost the same features for chunking and NER (except gazetteers).

[14]Being semi-supervised, JESS-CM can learn from interactions between labelled and unlabelled data during training but the training is slow compared to supervised models.

	Models	CONLL 2000	CONLL 2003
Ours	grSemi-CRF (Random embeddings)	93.92	84.66
	grSemi-CRF (Senna embeddings)	**95.01**	**89.44** (**90.87**)
Neural Models	Senna (Random embeddings)	90.33	81.47
	Senna (Senna embeddings)	94.32	88.67 (89.59)
	BI-LSTM-CRF (Random)	94.13	84.26
	BI-LSTM-CRF (Senna embeddings)	94.46	88.83 (90.10)
Non-Neural Models	JESS-CM (Suzuki and Isozaki, 2008), 15M	94.67	89.36
	JESS-CM (Suzuki and Isozaki, 2008), 1B	**95.15**	89.92
	Ratinov and Roth (2009)[12]	–	90.57
	Lin and Wu (2009)	–	**90.90**
	Passos et al. (2014)	–	**90.90**

Table 2: Experimental results over the CONLL-2000 and CONLL-2003 shared datasets, measured in F_1 score. Numbers in parentheses are the F_1 score when using gazetteers. JESS-CM (Suzuki and Isozaki, 2008) is a semi-supervised model, in which 15M or 1B denotes the number of unlabeled words it uses for training.

Gating Coefficients	CONLL 2000	CONLL 2003
Scalars	94.47	89.27(90.54)
Vectors	**95.01**	**89.44**(**90.87**)

Table 4: F_1 scores of grSemi-CRF with scalar or vectorial gating coefficients. Numbers in parentheses are the F_1 score when using gazetteers.

that the CONLL 2000 dataset is the subset of Wall Street Journal (WSJ) part of the Penn Treebank II Corpus (Marcus et al., 1993) while the CONLL 2003 dataset is a subset of Reuters RCV1 dataset. Maybe the writing styles between NYT and WSJ are more similar than those between RCV1 and WSJ.

4.4.2 Impact of Vectorial Gating Coefficients

As Table 4 shows, a grSemi-CRF using vectorial gating coefficients (i.e., Eq. (7)) performs better than that using scalar gating coefficients (i.e., Eq. (6)), which provides evidences for the theoretical intuition that vectorial gating coefficients can make a detailed modeling of the combinations of segment-level latent features and thus performs better than scalar gating coefficients.

4.4.3 Visualization of Learnt Segment-Level Features

To demonstrate the quality of learnt segment-level features, we use an indirect way as widely adopted in previous work, e.g., Collobert et al. (2011). More specifically, we show 10 nearest neighbours for some selected queried segments according to Euclidean metric of corresponding features[15]. To fully demonstrate the power of grSemi-CRF in learning segment-level features, we use the Emb+Brown(RCV1) model in Table 3, which uses no gazetteers. We train the model on the CONLL 2003 training set and find nearest neighbours in the CONLL 2003 test set. We make no restrictions on segments, i.e., all possible segments with different lengths in the CONLL 2003 test set are candidates.

As Table 5 shown, most of the nearest segments are meaningful and semantically related. For example, the nearest segments for "Filippo Inzaghi" are not only tagged with *person*, but also names of famous football players as "Filippo Inzaghi".

There also exist some imperfect results. E.g., for "Central African Republic", nearest segments, which contain the same queried segment, are semantically related but not syntactically similar. The major reason may be that the CONLL 2003 dataset is a small corpus (if compared to the vast unlabelled data used to train Senna embeddings), which restricts the range for candidate segments and the quality of learnt segment-level features. Another reason is that labels in the CONLL 2003 dataset mainly encodes semantic information (e.g., named entities) instead of syntactic information (e.g., chunks).

Besides, as we make no restriction on the formulation of candidate segments, sometimes only a part of the whole phrase will be retrieved, e.g., "FC Hansa", which is the prefix of "FC Hansa Rostock". Exploring better way of utilizing unla-

[15]Using the cosine similarity generates similar results. However, as Collobert et al. (2011) use Euclidean metric, we follow their settings.

Queried Segments	Filippo Inzaghi	AC Milan	Central African Republic	Asian Cup
Nearest Neighbour Results	Pierluigi Casiraghi	FC Hansa	From Central African Republic	Scottish Cup
	Fabrizio Ravanelli	SC Freiburg	Southeast Asian Nations	European Cup
	Bogdan Stelea	FC Cologne	In Central African Republic	African Cup
	Francesco Totti	Aston Villa	The Central African Republic	World Cup
	Predrag Mijatovic	Red Cross	South African Breweries	UEFA Cup
	Fausto Pizzi	Yasuto Honda	Of Southeast Asian Nations	Europoean Cup
	Pierre Laigle	NAC Breda	New South Wales	Asian Games
	Pavel Nedved	La Plagne	Central African Republic .	Europa Cup
	Anghel Iordanescu	Sporting Gijon	Papua New Guinea	National League
	Zeljko Petrovic	NEC Nijmegen	Central Africa	F.A. Cup

Table 5: Visualization of segment-level features learnt on the CONLL 2003 dataset. For each column the queried segment is followed by its 10 nearest neighbors (measured by the cosine similarity of their feature vectors). Corresponding tags for these four queried segments are (from left to right): *person*, *organization*, *location* and *miscellaneous*.

belled data to improve learning segment-level features is part of the future work.

5 Discussions and Related Work

Cho et al. (2014) first propose grConvs to learn fix-length representations of the whole source sentence in neural machine translation. Zhao et al. (2015) use grConvs to learn hierarchical representations (i.e., multiple fix-length representations) of the whole sentence for sentence-level classification problem. Both of them focus on sentence-level classification problems while grSemi-CRFs are solving segment-level classification (sequence tagging) problems, which is fine-grained. Chen et al. (2015) propose Gated Recursive Neural Networks (GRNNs), a variant of grConvs, to solve Chinese word segmentation problem. GRNNs still do word-level modeling by using CRFs while grSemi-CRFs do segment-level modeling directly by using semi-CRFs and makes full use of the recursive structure of grConvs.

We believe that, the recursive neural network (e.g., grConv) is a natural feature extractor for Semi-CRFs, as it extracts features for every possible segments by one propagation over one trained model, which is fast-computing and efficient. In this sense, grSemi-CRFs provide a promising direction to explore.

6 Conclusions

In this paper, we propose Gated Recursive Semi-Markov Conditional Random Fields (grSemi-CRFs) for segment-level sequence tagging tasks. Unlike word-level models such as CRFs, grSemi-CRFs model segments directly without the need of using extra tagging schemes and also readily utilize segment-level features, both hand-crafted and automatically extracted by a grConv. Experimental evaluations demonstrate the effectiveness of grSemi-CRFs on both text chunking and NER tasks.

In future work, we are interested in exploring better ways of utilizing vast unlabelled data to improve grSemi-CRFs, e.g., to learn phrase embeddings from unlabelled data or designing a semi-supervised version of grSemi-CRFs.

Acknowledgments

J.W.Z, J.Z and B.Z are supported by the National Basic Research Program (973 Program) of China (No. 2013CB329403), National NSF of China (Nos. 61322308, 61332007), the Youngth Top-notch Talent Support Program, Tsinghua TNList Lab Big Data Initiative, and Tsinghua Initiative Scientific Research Program (No. 20141080934).

References

Galen Andrew. 2006. A hybrid markov/semi-markov conditional random field for sequence segmentation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 465–472.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, and Xuanjing Huang. 2015. Gated recursive neural network for chinese word segmentation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, page 103.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

Percy Liang. 2005. *Semi-Supervised Learning for Natural Language*. Ph.D. thesis, Massachusetts Institute of Technology.

Dekang Lin and Xiaoyun Wu. 2009. Phrase clustering for discriminative learning. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 1030–1038.

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

Daisuke Okanohara, Yusuke Miyao, Yoshimasa Tsuruoka, and Jun'ichi Tsujii. 2006. Improving the scalability of semi-markov conditional random fields for named entity recognition. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 465–472.

Alexandre Passos, Vineet Kumar, and Andrew McCallum. 2014. Lexicon infused phrase embeddings for named entity resolution. *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 78–86.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third Workshop on Very Large Corpora*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, pages 147–155.

Evan Sandhaus. 2008. The new york times annotated corpus. *Linguistic Data Consortium, Philadelphia*, 6.

Sunita Sarawagi and William W Cohen. 2004. Semimarkov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, pages 1185–1192.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Jun Suzuki and Hideki Isozaki. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 665–673.

Erik F. Tjong Kim Sang and Sabine Buchholz. 2000. Introduction to the conll-2000 shared task: Chunking. In *Proceedings of the Forth Conference on Computational Natural Language Learning*, pages 127–132.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Computational Natural Language Learning*, pages 142–147.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

Ralph Weischedel and Ada Brunstein. 2005. Bbn pronoun coreference and entity type corpus. *Linguistic Data Consortium, Philadelphia*, 112.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345. Association for Computational Linguistics.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 4069–4076.

Jun Zhu, Zaiqing Nie, Ji-Rong Wen, Bo Zhang, and Hsiao-Wuen Hon. 2007. Webpage understanding: an integrated approach. In *Proceedings of SIGKDD Conference on Knowledge Discovery and Data Mining*.

Identifying Causal Relations Using Parallel Wikipedia Articles

Christopher Hidey
Department of Computer Science
Columbia University
New York, NY 10027
chidey@cs.columbia.edu

Kathleen McKeown
Department of Computer Science
Columbia University
New York, NY 10027
kathy@cs.columbia.edu

Abstract

The automatic detection of causal relationships in text is important for natural language understanding. This task has proven to be difficult, however, due to the need for world knowledge and inference. We focus on a sub-task of this problem where an open class set of linguistic markers can provide clues towards understanding causality. Unlike the explicit markers, a closed class, these markers vary significantly in their linguistic forms. We leverage parallel Wikipedia corpora to identify new markers that are variations on known causal phrases, creating a training set via distant supervision. We also train a causal classifier using features from the open class markers and semantic features providing contextual information. The results show that our features provide an 11.05 point absolute increase over the baseline on the task of identifying causality in text.

1 Introduction

The automatic detection of causal relationships in text is an important but difficult problem. The identification of causality is useful for the understanding and description of events. Causal inference may also aid upstream applications such as question answering and text summarization. Knowledge of causal relationships can improve performance in question answering for "why" questions. Summarization of event descriptions can be improved by selecting causally motivated sentences. However, causality is frequently expressed implicitly, which requires world knowledge and inference. Even when causality is explicit, there is a wide variety in how it is expressed.

Causality is one type of relation in the Penn Discourse Tree Bank (PDTB) (Prasad et al, 2008). In general, discourse relations indicate how two text spans are logically connected. In PDTB theory, these discourse relations can be marked explicitly or conveyed implicitly. In the PDTB, there are 102 known explicit discourse markers such as "and", "but", "after", "in contrast", or "in addition". Of these, 28 explicitly mark causal relations (e.g., "because", "as a result", "consequently").

In addition to explicit markers, PDTB researchers recognize the existence of an open class of markers, which they call *AltLex*. There is a tremendous amount of variation in how AltLexes are expressed and so the set of AltLexes is arguably infinite in size. In the PDTB, non-causal AltLexes include "That compares with" and "In any event." Causal AltLexes include "This may help explain why" and "This activity produced."

Discourse relations with explicit discourse markers can be identified with high precision (Pitler and Nenkova, 2009) but they are also relatively rare. Implicit relations are much more common but very difficult to identify. AltLexes fall in the middle; their linguistic variety makes them difficult to identify but their presence improves the identification of causality.

One issue with causality identification is the lack of data. Unsupervised identification on open domain data yields low precision (Do et al, 2011) and while supervised methods on the PDTB have improved (Ji and Eisenstein, 2015), creating enough labeled data is difficult. Here, we present a distant supervision method for causality identification that uses parallel data to identify new causal connectives given a seed set. We train a classifier on this data and self-train to obtain new data. Our novel approach uses AltLexes that were automatically identified using semi-supervised learning over a parallel corpus. Since we do not know

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1424–1433,
Berlin, Germany, August 7-12, 2016.

a priori what these phrases are, we used a monolingual parallel corpus to identify new phrases that are aligned with known causal connectives. As large corpora of this type are rare, we used Simple and English Wikipedia to create one.

Section 2 discusses prior research in causality and discourse. Section 4 describes how we created a new corpus from Wikipedia for causality and extracted a subset of relations with AltLexes. In section 5, we recount the semantic and marker features and how they were incorporated into a classifier for causality. We show that these features improve causal inference by an 11.05 point increase in F-measure over a naive baseline in 6. Finally, we discuss the results and future work in 7.

2 Related Work

Recent work on causality involved a combination of supervised discourse classification with unsupervised metrics such as PMI (Do et al, 2011). They used a minimally supervised approach using integer linear programming to infer causality. Other work focused on specific causal constructions events paired by verb/verb and verb/noun (Riaz and Girju, 2013) (Riaz and Girju, 2014). Their work considered semantic properties of nouns and verbs as well as text-only features.

There has also been significant research into discourse semantics over the past few years. One theory of discourse structure is represented in the PDTB (Prasad et al, 2008). The PDTB represents discourse relationships as connectives between two arguments. Early work with the PDTB (Pitler and Nenkova, 2009) showed that discourse classes with explicit discourse connectives can be identified with high accuracy using a combination of the connective and syntactic features. Further work (Pitler et al, 2009) resulted in the identification of implicit discourse relations using word pair features; this approach extended earlier work using word pairs to identify rhetorical relations (Marcu, 2001) (Blair-Goldensohn et al, 2007). These word pairs were created from text by taking the cross product of words from the Gigaword corpus for explicit causal and contrast relations. Others built on this work by aggregating word pairs for every explicit discourse connective (Biran and McKeown, 2013). They then used the cosine similarity between a prospective relation and these word pairs as a feature. Recently, the first end-to-end discourse parser was completed (Lin et al, 2012). This parser jointly infers both argument spans and relations. The current state-of-the-art discourse relation classifier is a constituent parse recursive neural network with coreference (Ji and Eisenstein, 2015).

Our work is similar to previous work to identify discourse connectives using unsupervised methods (Laali, 2014). In their research, they used the EuroParl parallel corpus to find discourse connectives in French using known English connectives and filtering connectives using patterns. Unlike this effort, we created our own parallel corpus and we determined new English connectives.

Compared to previous work on causality, we focus specifically on causality and the AltLex. The work by Do and Riaz used minimally supervised (Do et al, 2011) or unsupervised (Riaz and Girju, 2013) approaches and a slightly different definition of causality, similar to co-ocurrence. The work of Riaz and Girju (2013) is most similar to our own. We also examine causality as expressed by the author of the text. However, they focus on intra-sentence constructions between noun or verb phrases directly whereas we attempt to examine how the AltLex connectives express causality in context. Lastly, Riaz and Girju used FrameNet and WordNet to identify training instances for causal verb-verb and verb-noun pairs (Riaz and Girju, 2014) whereas we use them as features for an annotated training set. Overall our contributions are a new dataset created using a distant supervision approach and new features for causality identification. One major advantage is that our method requires very little prior knowledge about the data and requires only a small seed set of known connectives.

3 Linguistic Background

One disadvantage of the PDTB is that the marked AltLexes are limited only to discourse relations across sentences. We know that there are additional phrases that indicate causality within sentences but these phrases are neither found in the set of Explicit connectives nor AltLexes. Thus we expand our definition of AltLex to include these markers when they occur within a sentence. Although some phrases or words could be identified by consulting a thesaurus or the Penn Paraphrase Database (Ganitkevitch et al, 2013), we still need the context of the phrase to identify causality.

We hypothesize that there is significant linguis-

tic variety in causal AltLexes. In the set of known explicit connectives there are adjectives ("subsequent"), adverbs ("consequently"), and prepositions and prepositional phrases ("as a result"). We consider that these parts of speech and syntactic classes can be found in AltLexes as well. In addition, verbs and nouns often indicate causality but are not considered explicit connectives.

Some obvious cases of AltLexes are the verbal forms of connectives such as "cause" and "result". In addition to these verbs, there exist other verbs that can occur in causal contexts but are ambiguous. Consider that "make" and "force" can replace "cause" in this context:

> The explosion **made** people evacuate the building.
>
> The explosion **forced** people to evacuate the building.
>
> The explosion **caused** people to evacuate the building.

However, the words can not be substituted in the following sentence:

> The baker **made** a cake.
>
> *The baker **caused** a cake.
>
> *The baker **forced** a cake.

Furthermore, verbs such as "given" may replace additional causal markers:

> It's not surprising he is tired **since** he did not get any sleep.
>
> It's not surprising he is tired **given that** he did not get any sleep.

There are also some phrases with the same structure as partial prepositional phrases like "as a result" or "as a result of", where the pattern is preposition and noun phrase followed by an optional preposition. Some examples of these phrases include "on the basis of," "with the goal of," and "with the idea of."

We may also see phrases that are only causal when ending in a preposition such as "thanks to" or "owing to." "Lead" may only be causal as a part of "lead to" and the same for "develop" versus "develop from." In addition, prepositions can affect the direction of the causality. Comparing "resulting in" versus "resulting from", the preposition determines that the latter is of the "reason" class and the former is of the "result" class.

Ultimately, we want to be able to detect these phrases automatically and determine whether they are a large/small and open/closed class of markers.

4 Data

In order to discover new causal connectives, we can leverage existing information about known causal connectives. It should be the case that if a phrase is a causal AltLex, it will occur in some context as a replacement for at least one known explicit connective. Thus, given a large dataset, we would expect to find some pairs of sentences where the words are very similar except for the connective. This approach requires a parallel corpus to identify new AltLexes. As large English paraphrase corpora are rare, we draw from previous work identifying paraphrase pairs in Wikipedia (Hwang et al, 2015).

The dataset we used was created from the English and Simple Wikipedias from September 11, 2015. We used the software WikiExtractor to convert the XML into plain text. All articles with the same title were paired and any extra articles were ignored. Each article was lemmatized, parsed (both constituent and dependency), and named-entity tagged using the Stanford CoreNLP suite (Manning et al, 2014). We wish to identify paraphrase pairs where one element is in English Wikipedia and one is in Simple Wikipedia. Furthermore, we do not limit these elements to be single sentences because an AltLex can occur within a sentence or across sentences.

Previous work (Hwang et al, 2015) created a score for similarity (WikNet) between English Wikipedia and Simple Wikipedia. Many similarity scores are of the following form comparing sentences W and W':

$$s(W, W') = \frac{1}{Z} \sum_{w \in W} \max_{w' \in W'} \sigma(w, w') idf(w) \quad (1)$$

where $\sigma(w, w')$ is a score[1] between 2 words and Z is a normalizer ensuring the score is between 0 and 1. For their work, they created a score where $\sigma(w, w') = \sigma_{wk}(w, w') + \sigma_{wk}(h, h')\sigma_r(r, r')$. σ_{wk} is a distance function derived from Wiktionary by creating a graph based on words appearing in a definition. h and h' are the governors of w and w' in a dependency parse and r and r' are the relation. Similar sentences should have similar structure and the governors of two words in different sentences should also be similar. σ_r is 0.5 if h and h' have the same relation and 0 otherwise.

For this work, we also include partial matches, as we only need the connective and the immediate

[1]The score is not a metric, as it is not symmetric.

Method	Max F1
WikNet	0.4850
WikNet, $\lambda = 0.75$	0.5981
Doc2Vec	0.6226
Combined	0.6263

Table 1: Paraphrase Results

surrounding context on both sides. If one sentence contains an additional clause, it does not affect whether it contains a connective. Thus, one disadvantage to this score is that when determining whether a sentence is a partial match to a longer sentence or a shorter sentence, the longer sentence will often be higher as there is no penalty for unmatched words between the two elements. We experimented with penalizing content words that do not match any element in the other sentence. The modified score, where W and W' are nouns, verbs, adjectives, or adverbs, is then:

$$s(W, W') = \frac{1}{Z} \sum_{w \in W} \max_{w' \in W'} \sigma(w, w') idf(w) - \lambda(|W' - W| + |W - W'|) \quad (2)$$

We also compared results with a model trained using doc2vec (Le and Mikolov, 2014) on each sentence and sentence pair and identifying paraphrases with their cosine similarity.

As these methods are unsupervised, only a small amount of annotated data is needed to tune the similarity thresholds. Two graduate computer science students annotated a total of 45 Simple/English article pairs. There are 3,891 total sentences in the English articles and 794 total sentences in the Simple Wikipedia articles. Inter-annotator agreement (IAA) was 0.9626, computed on five of the article pairs using Cohen's Kappa. We tune the threshold for each possible score: for doc2vec the cosine similarity and for WikNet the scoring function. We also tune the lambda penalty for WikNet. F1 scores were calculated via grid search over these parameters and the best settings are a combined score using doc2vec and penalized WikNet with $\lambda = 0.75$ where a pair is considered to be a paraphrase if either threshold is greater than 0.69 or 0.65 respectively.

Using the combined score we obtain 187,590 paraphrase pairs. After combining and deduping this dataset with the publicly available dataset released by (Hwang et al, 2015), we obtain 265,627 pairs, about 6 times as large as the PDTB.

In order to use this dataset for training a model to distinguish between causal and non-causal instances, we use the paired data to identify pairs where an explicit connective appears in at least one element of the pair. The explicit connective can appear in a Simple Wikipedia sentence or an English Wikipedia sentence. We then use patterns to find new phrases that align with these connectives in the matching sentence.

Class	Type	Subtype
Temporal		
Contingency	Cause	reason
		result
	Pragmatic cause	
	Condition	
	Pragmatic condition	
Comparison		
Expansion		

Table 2: PDTB Discourse Classes

To identify a set of seed words that unambiguously identify causal and non-causal phrases we examine the PDTB. As seen in Table 2, causal relations fall under the Contingency class and Cause type. We consider connectives from the PDTB that either only or never appear as that type. The connective "because" is the only connective to be almost always a "reason" connective, whereas there are 11 unambiguous connectives for "result", including "accordingly", "as a consequence", "as a result", and "thus". There were many markers that were unambiguously not causal (e.g. "but", "though", "still", "in addition").

In order to label paraphrase data, we use constraints to identify possible AltLexes.[2] We used Moses (Koehn et al, 2007) to train an alignment model on the created paraphrase dataset. Then for every paraphrase pair we identify any connectives that match with any potential AltLexes. Based on our linguistic analysis, we require these phrases to contain at least one content word, which we identify based on part of speech. We also draw on previous work (Pitler and Nenkova, 2009) that used the left and right sibling of a phrase. Therefore, we use the following rules to label new AltLexes:

1. Must be less than 7 words.
2. Must contain at least one content word:
   (a) A non-proper noun
   (b) A non-modal and non-auxiliary verb
   (c) An adjective or adverb
3. Left sibling of the connective must be a noun phrase, verb phrase, or sentence.
4. Right sibling of the connective must be a noun phrase, verb phrase, or sentence.

[2]We do not attempt to label arguments at this point.

5. May not contain a modal or auxilary verb.

Because connectives identify causality between events or agents, we require that each potential connective link 2 events/agents. We define an event or agent as a noun, verb, or an entire sentence. This means that we require the left sibling of the first word in a phrase and the right sibling of the last word in a phrase to be an event, where a sibling is the node at the same level in the constituent parse. We also require the left and right sibling rule for the explicit connectives, but we allow additional non-content words (for example, we would mark "because of" as a connective rather than "because." We then mark the AltLex as causal or not causal.

Given that the paraphrases and word alignments are noisy, we use the syntactic rules to decrease the amount of noise in the data by more precisely determining phrase boundaries. These rules are the same features used by Pitler and Nenkova (2009) for the early work on the PDTB on explicit connectives. These features were successful on the Wall Street Journal and they are applicable for other corpora as well. Also, they are highly indicative of discourse/non-discourse usage so we believe that we are improving on noisy alignments without losing valuable data. In the future, however, we would certainly like to move away from encoding these constraints using a rule-based method and use a machine learning approach to automatically induce rules.

This method yields 72,135 non-causal and 9,190 causal training examples. Although these examples are noisy, the dataset is larger than the PDTB and was derived automatically. There are 35,136 argument pairs in the PDTB marked with one of the 3 relations that implies a discourse connective (Implicit, Explicit, and AltLex), and of these 6,289 are causal. Of the 6,289 causal pairs, 2,099 are explicit and 273 contain an AltLex.

5 Methods

Given training data labeled by this distant supervision technique, we can now treat this problem as a supervised learning problem and create a classifier to identify causality.

We consider two classes of features: features derived from the parallel corpus data and lexical semantic features. The parallel corpus features are created based on where AltLexes are used as paraphrases for causal indicators and in what context. The lexical semantic features use FrameNet, WordNet, and VerbNet to derive features from all the text in the sentence pair. These lexical resources exploit different perspectives on the data in complementary ways.

The parallel corpus features encourage the classifier to select examples with AltLexes that are likely to be causal whereas the lexical semantic features allow the classifier to consider context for disambiguation. In addition to the dataset, the parallel corpus and lexical semantic features are the main contributions of this effort.

5.1 Parallel Corpus Features

We create a subclass of features from the parallel corpus: a KL-divergence score to encourage the identification of phrases that replace causal connectives. Consider the following datapoints and assume that they are aligned in the parallel corpus:

I was late **because of** traffic.

I was late **due to** traffic.

We want both of these examples to have a high score for causality because they are interchangeable causal phrases. Similarly, we want non-causal phrases that are often aligned to have a high score for non-causality.

We define several distributions in order to determine whether an AltLex is likely to replace a known causal or non-causal connective. We consider all aligned phrases, not just ones containing a causal or non-causal connective to attempt to reduce noisy matches. The idea is that non-connective paraphrases will occur often and in other contexts.

The following conditional Bernoulli distributions are calculated for every aligned phrase in the dataset, where w is the phrase, s is the sentence it occurs in, c is "causal" and nc is "not causal":

$$p_1 = p(w_1 \in s_1 | rel(s_1) \in \{c\}, w_1 \notin s_2) \quad (3)$$

$$p_2 = p(w_1 \in s_1 | rel(s_1) \in \{nc\}, w_1 \notin s_2) \quad (4)$$

We compare these two distributions to other distributions with the same word and in a different context (where o represents "other"):

$$q_1 = p(w_1 \in s_1 | rel(s_1) \in \{nc, o\}, w_1 \notin s_2) \quad (5)$$

$$q_2 = p(w_1 \in s_1 | rel(s_1) \in \{c, o\}, w_1 \notin s_2) \quad (6)$$

We then calculate $D_{KL}(p_1||q_1)$ and $D_{KL}(p_2||q_2)$. In order to use KL-divergence as a feature, we multiply the score by $(-1)^{p<q}$ and add a feature for **causal** and one for **non-causal**.

5.2 Lexical Semantic Features

As events are composed of predicates and arguments and these are usually formed by nouns and verbs, we consider using lexical semantic resources that have defined hierarchies for nouns and verbs. We thus use the lexical resources FrameNet, WordNet, and VerbNet as complementary resources from which to derive features. We hypothesize that these semantic features provide context not present in the text; from these we are able to infer causal and anti-causal properties.

FrameNet is a resource for frame semantics, defining how objects and relations interact, and provides an annotated corpus of English sentences. WordNet provides a hierarchy of word senses and we show that the top-level class of verbs is useful for indicating causality. VerbNet provides a more fine-grained approach to verb categorization that complements the views provided by FrameNet and WordNet.

In **FrameNet**, a semantic frame is a conceptual construction describing events or relations and their participants (Ruppenhofer et al, 2010). Frame semantics abstracts away from specific utterances and ordering of words in order to represent events at a higher level. There are over 1,200 semantic frames in FrameNet and some of these can be used as evidence or counter-evidence for causality (Riaz and Girju, 2013). In Riaz's work, they identified 18 frames as causal (e.g. "Purpose", "Internal cause", "Reason", "Trigger").

We use these same frames to create a lexical score based on the FrameNet 1.5 corpus. This corpus contains 170,000 sentences manually annotated with frames. We used a part-of-speech tagged version of the FrameNet corpus and for each word and tag, we count how often it occurs in the span of one of the given frames. We only considered nouns, verbs, adjectives, and adverbs. We then calculate $p_w(c|t)$ and c_{wct}, the probability that a word w is causal given its tag t and its count, respectively. The lexical score of a word i is calculated by using the assigned part-of-speech tag and is given by $CS_i = p_{w_i}(c|t_i) \log c_{w_i c t_i}$. The total score of a sequence of words is then $\sum_{i=0}^{n} CS_i$.

We also took this further and determined what frames are likely to be *anti-causal*. We started with a small set of seed words derived directly from 11 discourse classes (types and subtypes from Table 2), such as "Compare", "Contrast", "Explain", "Concede", and "List". We expanded this list using WordNet synonyms for the seed words. We then extracted every frame associated with their stems in the stemmed FrameNet corpus. These derived frames were manually examined to develop a list of 48 anti-causal frames, including "Statement", "Occasion", "Relative time", "Evidence", and "Explaining the facts".

We create an anti-causal score using the FrameNet corpus just as we did for the causal score. The total anti-causal score of a sequence of words is $\sum_{i=0}^{n} ACS_i$ where $ACS_i = p_{w_i}(a|t_i) \log c_{w_i a t_i}$ for anti-causal probabilities and counts. We split each example into three parts: the text before the AltLex, the AltLex, and the text after. Each section is given a causal score and an anti-causal score. Overall, there are six features derived using FrameNet: causal score and anti-causal score for each part of the example.

In **WordNet**, words are grouped into "synsets," which represent all synonyms of a particular word sense. Each word sense in the WordNet hierarchy has a top-level category based on part of speech (Miller, 1995). Every word sense tagged as noun, verb, adjective, or adverb is categorized. Some examples of categories are "change", "stative", or "communication". We only include the top level because of the polysemous nature of WordNet synsets. We theorize that words having to do with change or state should be causal indicators and words for communication or emotion may be anti-causal indicators.

Similar to the FrameNet features, we split the example into three sections. However, we also consider the dependency parse of the data. We believe that causal relations are between events and agents which are represented by nouns and verbs. Events can also be represented by predicates and their arguments, which is captured by the dependency parse. As the root of a dependency parse is often a verb and sometimes a noun or adjective, we consider the category of the root of a dependency parse and its arguments.

We include a categorical feature indicating the top-level category of the root of each of the three sections, including the AltLex. For both sides of the AltLex, we include the top-level category of all arguments as well. If a noun has no category, we mark it using its named-entity tag. If there is still no tag, we mark the category as "none."

VerbNet VerbNet is a resource devoted to storing information for verbs (Kipper et al, 2000).

In contrast to WordNet, VerbNet provides a more fine-grained description of events while focusing less on polysemy. Some examples of VerbNet classes are "force", "indicate", and "wish". In VerbNet, there are 273 verb classes, and we include their presence as a categorical feature. Similar to WordNet, we use VerbNet categories for three sections of the sentence: the text pre-AltLex, the AltLex, and the text post-AltLex. Unlike WordNet, we only mark the verbs in the AltLex, root, or arguments.

Interaction Finally, we consider interactions between the WordNet and VerbNet features. As previous work (Marcu, 2001) (Biran and McKeown, 2013) used word pairs successfully, we hypothesize that pairs of higher-level categories will improve classification without being penalized as heavily by the sparsity of dealing with individual words. Thus we include interaction features between every categorical feature for the pre-AltLex text and every feature for the post-AltLex text.

In all, we include the following features (L refers to the AltLex, B refers to the text *before* the AltLex and A refers to the text *after* the AltLex):

1. FrameNet causal score for L, B, and A.
2. FrameNet anti-causal score for L, B, and A.
3. WordNet top-level of L.
4. WordNet top-level of the root of B and A.
5. WordNet top-level for arguments of B and A.
6. VerbNet category for verb at the root of L.
7. VerbNet top-level category for any verb in the root of B and A.
8. VerbNet top-level category for any verbs in the arguments of B and A.
9. Categorical interaction features between the features from B and the features from A.

6 Results

We evaluated our methods on two manually annotated test sets. We used one of these test sets for development only. For this set, one graduate computer science student and two students from the English department annotated a set of Wikipedia articles by marking any phrases they considered to indicate a causal relationship and marking the phrase as "reason" or "result." Wikipedia articles from the following categories were chosen as we believe they are more likely to contain causal relationships: science, medicine, disasters, history, television, and film. For each article in this category, both the English and Simple Wikipedia articles were annotated. A total of 12 article pairs were annotated. IAA was computed to be 0.31 on two article pairs using Kripendorff's alpha.

IAA was very low and we also noticed that annotators seemed to miss sentences containing causal connectives. It is easy for an annotator to overlook a causal relation when reading through a large quantity of text. Thus, we created a new task that required labeling a connective as causal or not when provided with the sentence containing the connective. For testing, we used CrowdFlower to annotate the output of the system using this method. We created a balanced test set by annotating 600 examples, where the system labeled 300 as causal and 300 as non-causal. Contributors were limited to the highest level of quality and from English-speaking countries. We required 7 annotators for each data point. The IAA was computed on the qualification task that all annotators were required to complete. There were 15 questions on this task and 410 annotators. On this simplified task, the IAA improved to 0.69.

We also considered evaluating the results on the PDTB but encountered several issues. As the PDTB only has a limited set of explicit intra-sentence connectives marked, this would not show the full strength of our method. Many causal connectives that we discovered are not annotated in the PDTB. Alternatively, we considered evaluating on the AltLexes in the PDTB but these examples are only limited to inter-sentence cases, whereas the vast majority of our automatically annotated training data was for the intra-sentence case. Thus we concluded that any evaluation on the PDTB would require additional annotation. Our goal in this work was to identify new ways in which causality is expressed, unlike the PDTB where annotators were given a list of connectives and asked to determine discourse relations.

We tested our hypothesis by training a binary[3] classifier on our data using the full set of features we just described. We used a linear Support Vector Machine (SVM) classifier (Vapnik, 1998) trained using stochastic gradient descent (SGD) through the sci-kit learn package. (Pedregosa et al, 2011)[4] We used elasticnet to encourage sparsity and tuned the regularization constant α through grid search.

We use two baselines. The first baseline is the

[3]We combine "reason" and "result" into one "causal" class and plan to work on distinguishing between non-causal, reason, and result in the future.

[4]We also considered a logistic regression classifier.

	Accuracy	True Precision	True Recall	True F-measure
Most Common Class	63.50	60.32	82.96	69.85
$CONN$	62.21	78.47	35.64	49.02
LS	67.68	61.98	58.51	60.19
KLD	58.03	**91.17**	19.55	32.20
$LS \cup KLD$	**73.95**	80.63	64.35	71.57
$LS \cup LS_{inter}$	72.99	78.54	64.66	70.93
$KLD \cup LS \cup LS_{inter}$	70.09	76.95	58.99	66.78
$LS \cup KLD \cup CONN$	71.86	70.28	**77.60**	**73.76**
$Bootstrapping_1$	79.26	**77.97**	82.64	80.24
$Bootstrapping_2$	**79.58**	77.29	**84.85**	**80.90**

Table 3: Experimental Results

most common class of each AltLex according to its class in the initial training set. For example, "caused by" is almost always a causal AltLex. A second baseline uses the AltLex itself as a categorical feature and is shown as $CONN$ in Table 3. For comparison, this is the same baseline used in (Pitler and Nenkova, 2009) on the explicit discourse relations in the PDTB. We compare these two baselines to ablated versions of our system. We evaluate on the KLD (KLD) and semantic (LS and LS_{inter}) features described in sections 5 and 5.1. LS consists of features 1-8, all the FrameNet, VerbNet, and WordNet features. LS_{inter} includes only the interaction between categorical features from WordNet and VerbNet.

We calculate accuracy and true precision, recall, and F-measure for the causal class. As seen in Table 3, the best system ($LS \cup KLD \cup CONN$) outperforms the baselines.[5] The lexical semantic features by themselves (LS) are similar to those used by (Riaz and Girju, 2014) although on a different task and with the WordNet and VerbNet features included. Note that the addition of the Altlex words and KL divergence ($LS \cup KLD \cup CONN$) yields an absolute increase in f-measure of 13.57 points over lexical semantic features alone.

6.1 Bootstrapping

Our method for labeling AltLexes lends itself naturally to a bootstrapping approach. As we are using explicit connectives to identify new AltLexes, we can also use these new AltLexes to identify additional ones. We then consider any paraphrase pairs where at least one of the phrases contains one of our newly discovered AltLexes. We also use our classifier to automatically label these new data points and remove any phrases where the classifier did not agree on both elements in the pair. The set of features used were the $KLD \cup LS \cup LS_{inter}$ features as these performed best on the development set. We use early stopping on the development data to identify the point when adding additional data is not worthwhile. The bootstrapping method converges quickly. After 2 iterations we see a decrease in the F-measure of the development data.

The increase in performance on the test data is significant. In Table 3, $Bootstrapping_n$ refers to results after n rounds of bootstrapping. Bootstrapping yields improvement over the supervised method with an absolute gain of 7.14 points.

6.2 Discussion

Of note is that the systems without connectives (combinations of LS, LS_{inter}, and KLD) perform well on the development set without using any lexical features. Using this system enables the discovery of new AltLexes during bootstrapping, as we cannot rely on having a closed class of connectives but need a way of classifying connectives not seen in the initial training set.

Also important is that the Altlex by itself ($CONN$) performs poorly. In comparison, in the task of identifying discourse relations in the PDTB these features yield an 75.33 F-score and 85.85% accuracy in distinguishing between discourse and non-discourse usage (Pitler and Nenkova, 2009) and an accuracy of 93.67% when distinguishing between discourse classes. Although this is a different data set, this shows that identifying causality when there is an open class of connectives is much more difficult. We believe the connective by itself performs poorly because of the wide

[5]These results are statistically significant by a binomial test with $p < 7 * 10^{-6}$.

	True Precision	True Recall	True F-measure
$FrameNet$	67.88	53.14	59.61
$WordNet$	76.92	9.52	16.94
$VerbNet$	38.70	3.80	6.92

Table 4: Semantic Feature Ablation

linguistic variation in these alternative lexicalizations. Many connectives appear only once or not at all in the training set, so the additional features are required to improve performance.

In addition, the "most common class" baseline is a strong baseline. The strength of this performance provides some indication of the quality of the training data, as the majority of the time the connective is very indicative of its class in the held-out test data. However, the the overall accuracy is still much lower than if we use informative features.

The KLD and LS feature sets appear to be complementary. The KLD feature sets have higher precision on a smaller section of the data, whereas the LS system has higher recall overall. These lexical semantic features likely have higher recall because these resources are designed to represent *classes* of words rather than individual words. Some connectives occur very rarely, so it is necessary to generalize the key aspects of the connectives and class-based resources provide this capability.

In order to determine the contribution of each lexical resource, we perform additional feature ablation for each of FrameNet, WordNet, and VerbNet. As seen in Table 4, the lexical semantic resources each contribute uniquely to the classifier. The FrameNet features provide most of the performance of the classifier. The WordNet and VerbNet features, though not strong individually, supply complementary information and improve the overall performance of the LS system (see Table 3) compared to just using FrameNet alone.

Finally, the model ($LS \cup KLD \cup CONN$) correctly identifies some causal relations that neither baseline identifies, such as:

> Language is reduced to simple phrases or even single words, eventually **leading to** complete loss of speech.
>
> Kulap quickly accelerated north, **prompting** the PAGASA to issue their final advisory on the system.

These examples do not contain standard causal connectives and occur infrequently in the data, so the lexical semantic features help to identify them.

After two rounds of bootstrapping, the system is able to recover additional examples that were not found previously, such as:

> When he finally changed back, Buffy stabbed him **in order to** once again save the world.

This connective occurs rarely or not at all in the initial training data and is only recovered because of the improvements in the model.

7 Conclusion

We have shown a method for identifying and classifying phrases that indicate causality. Our method for automatically building a training set for causality is a new contribution. We have shown statistically significant improvement over the naive baseline using semantic and parallel corpus features. The text in the AltLex alone is not sufficient to accurately identify causality. We show that our features are informative by themselves and perform well even on rarely occurring examples.

Ultimately, the focus of this work is to improve detection of causal relations. Thus, we did not evaluate some intermediate steps, such as the quality of the automatically annotated corpus. Our use of distant supervision demonstrates that we can use a large amount of possibly noisy data to develop an accurate classifer. To evaluate on the intermediate step would have required an additional annotation process. In the future, we may improve this step using a machine learning approach.

Although we have focused exclusively on Wikipedia, these methods could be adapted to other domains and languages. Causality is not easily expressed in English using a fixed set of phrases, so we would expect these methods to apply to formal and informal text ranging from news and journals to social media. Linguistic expressions of causality in other languages is another avenue for future research, and it would be interesting to note if other languages have the same variety of expression.

References

Or Biran and Kathleen McKeown. 2013. *Aggregated Word Pair Features for Implicit Discourse Relation Disambiguation*. Proceedings of ACL.

Sasha Blair-Goldensohn, Kathleen McKeown, and Owen Rambow. 2007. *Building and Refining Rhetorical-Semantic Relation Models*. Proceedings of NAACL-HLT.

Nathanael Chambers and Dan Jurafsky. 2008. *Unsupervised Learning of Narrative Event Chains*. Stanford University. Stanford, CA 94305.

Quang Xuan Do, Yee Seng Chan, and Dan Roth. 2011. *Minimally Supervised Event Causality Identification*. Transactions of ACL, 3:329344.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. *Aligning Sentences from Standard Wikipedia to Simple Wikipedia*. Proceedings of NAACL-HLT.

Yangfeng Ji and Jacob Eisenstein. 2015. *One Vector is Not Enough: Entity-Augmented Distributed Semantics for Discourse Relations*. Proceedings of EMNLP.

Karin Kipper, Hoa Trang Dan, and Martha Palmer. 2000. *Class-Based Construction of a Verb Lexicon*. American Association for Artifical Intelligence.

Majid Laali and Leila Kosseim. 2014. *Inducing Discourse Connectives from Parallel Texts*. Proceedings of COLING: Technical Papers.

Quoc Le and Tomas Mikolov. 2014. *Distributed Representations of Sentences and Documents*. Proceedings of ICML.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2012. *A PDTB-Styled End-to-End Discourse Parser*. Department of Computer Science, National University of Singapore.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. *The Stanford CoreNLP Natural Language Processing Toolkit*. Proceedings of ACL: System Demonstrations.

Daniel Marcu and Abdessamad Echihabi 2001. *An Unsupervised Approach to Recognizing Discourse Relations*. Proceedings of ACL.

George A. Miller. 1995. *WordNet: A Lexical Database for English*. Communications of the ACM.

The PDTB Research Group 2008. *The PDTB 2.0. Annotation Manual*. Technical Report IRCS-08-01. Institute for Research in Cognitive Science, University of Pennsylvania.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. *Automatic sense prediction for implicit discourse relations in text* Proceedings of ACL.

Emily Pitler and Ani Nenkova. 2009. *Using Syntax to Disambiguate Explicit Discourse Connectives in Text*. Proceedings of ACL-IJCNLP Short Papers.

Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind Joshi and Bonnie Webber. 2008. *The Penn Discourse Treebank 2.0*. Proceedings of LREC.

Rashmi Prasad, Aravind Joshi, and Bonnie Webber. 2010. *Realization of Discourse Relations by Other Means: Alternative Lexicalizations*. Proceedings of COLING.

Kira Radinsky and Eric Horvitz. 2013. *Mining the Web to Predict Future Events*. Proceedings of WSDM.

Mehwish Riaz and Roxana Girju. 2013. *Toward a Better Understanding of Causality between Verbal Events: Extraction and Analysis of the Causal Power of Verb-Verb Associations*. Proceedings of SIGDIAL.

Mehwish Riaz and Roxana Girju. 2014. *Recognizing Causality in Verb-Noun Pairs via Noun and Verb Semantics*. Proceedings of the EACL 2014 Workshop on Computational Approaches to Causality in Language.

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2010. *FrameNet II: Extended Theory and Practice*. University of California, Berkeley.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. Wiley-Interscience.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. *PPDB: The Paraphrase Database*. Proceedings of NAACL-HLT.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. 2011. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research Volume 12.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondej Bojar, Alexandra Constantin, and Evan Herbst. 2007. *Moses: open source toolkit for statistical machine translation*. Proceedings of ACL: Interactive Poster and Demonstration Sessions.

Compositional Learning of Embeddings for Relation Paths in Knowledge Bases and Text

Kristina Toutanova
Microsoft Research
Redmond, WA, USA

Xi Victoria Lin*
Computer Science & Engineering
University of Washington

Wen-tau Yih
Microsoft Research
Redmond, WA, USA

Hoifung Poon
Microsoft Research
Redmond, WA, USA

Chris Quirk
Microsoft Research
Redmond, WA, USA

Abstract

Modeling relation paths has offered significant gains in embedding models for knowledge base (KB) completion. However, enumerating paths between two entities is very expensive, and existing approaches typically resort to approximation with a sampled subset. This problem is particularly acute when text is jointly modeled with KB relations and used to provide direct evidence for facts mentioned in it. In this paper, we propose the first exact dynamic programming algorithm which enables efficient incorporation of all relation paths of bounded length, while modeling both relation types and intermediate nodes in the compositional path representations. We conduct a theoretical analysis of the efficiency gain from the approach. Experiments on two datasets show that it addresses representational limitations in prior approaches and improves accuracy in KB completion.

1 Introduction

Intelligent applications benefit from structured knowledge about the entities and relations in their domains. For example, large-scale knowledge bases (KB), such as Freebase (Bollacker et al., 2008) or DBPedia (Auer et al., 2007), have proven to be important resources for supporting open-domain question answering (Berant et al., 2013; Sun et al., 2015; Yih et al., 2015). In biomedicine, KBs such as the Pathway Interaction Database (NCI-PID) (Schaefer et al., 2009) are crucial for understanding complex diseases such as cancer and for advancing precision medicine. While these knowledge bases are often carefully curated, they are far from complete. In non-static domains, new facts become true or are discovered at a fast pace, making the manual expansion of knowledge bases impractical. Extracting relations from a text corpus (Mintz et al., 2009; Surdeanu et al., 2012; Poon et al., 2015) or inferring facts from the relationships among known entities (Lao and Cohen, 2010) are thus important approaches for populating existing knowledge bases.

Originally proposed as an alternative statistical relational learning method, the *knowledge base embedding* approach has gained a significant amount of attention, due to its simple prediction time computation and strong empirical performance (Nickel et al., 2011; Chang et al., 2014). In this framework, entities and relations in a knowledge base are represented in a continuous space, such as vectors and matrices. Whether two entities have a previously unknown relationship can be predicted by simple functions of their corresponding vectors or matrices. Early work in this direction focuses on exploring various kinds of learning objectives and frameworks, but the model is learned solely from known *direct* relationships between two entities (e.g., `father(barack, sasha)`) (Nickel et al., 2011; Socher et al., 2013; Bordes et al., 2013; Chang et al., 2014; Yang et al., 2015). In contrast, using *multi-step* relation paths (e.g., `husband(barack, `*`michelle`*`) ∧ mother(`*`michelle`*`, sasha)`) to train KB embeddings has been proposed very recently (Guu et al., 2015; Garcia-Duran et al., 2015; Lin et al., 2015; Neelakantan et al., 2015).

While using relation paths improves model performance, it also poses a critical technical challenge. As the number of possible relation paths between pairs of entities grows exponentially with path length, the training complexity increases sharply. Consequently, existing methods need

*This research was conducted during the author's internship at Microsoft Research.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1434–1444,
Berlin, Germany, August 7-12, 2016.

to make approximations by sampling or pruning. The problem is worsened when the input is augmented with unlabeled text, which has been shown to improve performance (Lao et al., 2012; Gardner et al., 2013; Riedel et al., 2013; Gardner et al., 2014; Toutanova and Chen, 2015). Moreover, none of the prior methods distinguish relation paths that differ in the intermediate nodes they pass through (e.g., `michelle` in our example); all represent paths as a sequence of relation types.

In this work, we aim to develop a KB completion model that can incorporate relation paths efficiently. We start from analyzing the procedures in existing approaches, focusing on their time and space complexity. Based on the observation that compositional representations of relation paths are in fact decomposable, we propose a novel dynamic programming method that enables efficient modeling of *all* possible relation paths, while also representing *both* relation types and nodes on the paths.

We evaluated our approach on two datasets. The first is from the domain of gene regulatory networks. Apart from its obvious significance in biomedicine, it offers an excellent testbed for learning joint embedding of KBs and text, as it features existing knowledge bases such as NCI-PID and an even larger body of text that grows rapidly (over one million new articles per year). By modeling intermediate nodes on relation paths, we improve the model by 3 points in mean average precision compared to previous work, while also providing a more efficient algorithm. The second dataset is based on a network derived from WordNet and previously used in work on knowledge base completion. On that dataset we demonstrate the ability of the model to effectively handle longer relation paths composed of a larger set of knowledge base relation types, with smaller positive impact of modeling intermediate nodes.

2 Preliminaries

In this section, we first give a brief overview of the knowledge base and text representation used in this work. We then describe the task of knowledge base completion more formally and introduce our basic model setting and training objective.

Knowledge Base A knowledge base (KB) is represented as a collection of subject-predicate-object triples (s, r, t), where s and t are the subject and object entities from a set $\mathcal{E}$, and r is the predicate from a set $\mathcal{R}$ that denotes a relationship between s and t. For example, in a KB of movie facts, we may find a triple (`Han_Solo`, `character`, `Star_Wars`), indicating that "*Han Solo is a character in the Star Wars movie.*"

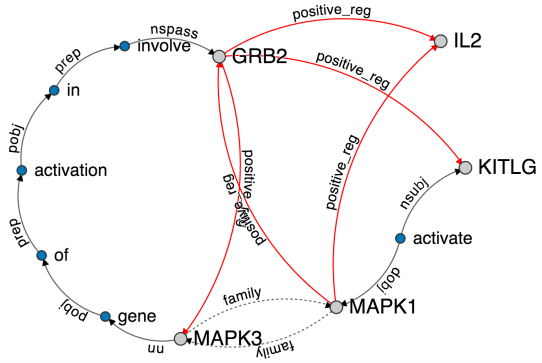


Figure 1: A snapshot depicting the knowledge graph of a gene regulatory network, augmented with gene family and dependency path relations.

Let $(s, \pi, t) = (s, r_1, e_1, r_2, e_2 \ldots, e_{n-1}, r_n, t)$ be a path in $\mathcal{G}$ with s/t as the start/end entities, $r_1, \ldots, r_n$ as the relation edges and $e_1, \ldots, e_{n-1}$ as the intermediate entities.

In the domain of gene regulations, entities are genes and directed edges represent regulations. At the abstract level, there are two key relations, positive_reg and negative_reg, which signify that the subject gene increases or decreases the activity of the object gene, respectively. Figure 1 shows a snapshot of a gene regulatory network. Genes such as `GRB2` and `MAPK3` are denoted by the light grey nodes. Regulations such as positive_reg are denoted by long edges pointing from subject to object. In addition, some genes stem from a common evolutionary origin and share similar functions. They form a "gene family", such as the `MAPK` family that includes `MAPK1` and `MAPK3`.

To jointly embed text, we use sentences containing co-occurring gene pairs, such as "*GRB2 was involved in the activation of gene MAPK3...*", and augment the above knowledge graph with dependency paths between the gene mentions, following the general approach of Riedel et al. (2013).

Task and Scoring Models The KB completion task is to predict the existence of links (s, r, t) that are not seen in the training knowledge base. More specifically, we focus on ranking object and subject entities for given queries $(s, r, *)$ and $(*, r, t)$.

The basic KB embedding models learn latent vector/matrix representations for entities and relations, and score each possible triple using learned parameters θ and a scoring function $f(s, r, t|\theta)$.

Below, we describe two basic model variations which we build on in the rest of the paper.

BILINEAR The BILINEAR model learns a square matrix $W_r \in \mathbb{R}^{d\times d}$ for each relation $r \in \mathcal{R}$ and a vector $x_e \in \mathbb{R}^d$ for each entity $e \in \mathcal{E}$. The scoring function of a relation triple (s, r, t) is defined as:

$$f(s, r, t|\theta) = x_s^\top W_r x_t. \quad (1)$$

For a knowledge graph $\mathcal{G}$ with $|\mathcal{E}| = N_e$ and $|\mathcal{R}| = N_r$, the parameter size of the BILINEAR model is $\mathrm{O}(d^2)$. The large number of parameters make it prone to overfitting, which motivates its diagonal approximation, BILINEAR-DIAG.

BILINEAR-DIAG The BILINEAR-DIAG model restricts the relation representations to the class of diagonal matrices. [1] In this case, the parameter size is reduced to $\mathrm{O}(d)$. Although we present model variants in terms of the BILINEAR representation, all experiments in this work are based on the BILINEAR-DIAG special case.

All models are trained using the same loss function, which maximizes the probability of correct subject/object fillers of a given set of triples with relations or relation paths. The probability is defined by a log-linear model normalized over a set of negative samples: $P(t|s, \pi; \theta) = \frac{e^{f(s,\pi,t|\theta)}}{\sum_{t'\in Neg(s,\pi,*)\cup\{t\}} e^{f(s,\pi,t'|\theta)}}$. The probability of subject entities is defined analogously.

Define the loss for a given training triple $L(s, \pi, t|\theta) = -\log P(t|s, \pi; \theta) - \log P(s|t, \pi; \theta)$. The overall loss function is the sum of losses over all triples, with an L_2 regularization term.

$$L(\theta) = \sum_i L(s_i, \pi_i, t_i; \theta) + \lambda\|\theta\|^2 \quad (2)$$

3 Relation-path-aware Models

We first review two existing methods using vector space relation paths modeling for KB completion in §3.1. We then introduce our new algorithm that can efficiently take into account all relation paths between two nodes as features and simultaneously model intermediate nodes on relation paths in §3.2. We present a detailed theoretical comparison of the efficiency of these three types of methods in §3.3.

3.1 Prior Approaches

The two approaches we consider here are: using relation paths to generate new auxiliary triples for training (Guu et al., 2015) and using relation paths as features for scoring (Lin et al., 2015).

Both approaches take into account embeddings of relation paths between entities, and both of them used vector space compositions to combine the embeddings of individual relation links r_i into an embedding of the path π. The intermediate nodes e_i are neglected. The natural composition function of a BILINEAR model is matrix multiplication (Guu et al., 2015). For this model, the embedding of a length-n path $\Phi_\pi \in \mathbb{R}^{d\times d}$ is defined as the matrix product of the sequence of relation matrices for the relations in π.

$$\Phi_\pi = W_{r_1} \dots W_{r_n}. \quad (3)$$

For the BILINEAR-DIAG model, all the matrices are diagonal and the computation reduces to coordinate-wise product of vectors in $\mathbb{R}^d$.

3.1.1 Relation Paths as a Compositional Regularizer

In Guu et al. (2015), information from relation paths was used to generate additional auxiliary terms in training, which serve to provide a compositional regularizer for the learned node and relation embeddings. A more limited version of the same method was simultaneously proposed in Garcia-Duran et al. (2015).

The method works as follows: starting from each node in the knowledge base, it samples m random walks of length 2 to a maximum length L, resulting in a list of samples $\{[s_i, \pi_i, t_i]\}$. s_i and t_i are the start and end nodes of the random walk, respectively, and π_i consists of a sequence of intermediate edges and nodes. Each of these samples is used to define a new triple used in the training loss function (eq. 2).

The score of each triple under a BILINEAR composition model is defined as $f(s_i, \pi_i, t_i|\theta) = x_{s_i}^t \Phi_{\pi_i} x_{t_i}$, where Φ_{π_i} is the product of matrices for relation link types in the path (eq. 3).

3.1.2 PRUNED-PATHS: Relation Paths as Compositional Features

Instead of using relation paths to augment the set of training triples, Lin et al. (2015) proposed to

[1] The downside of this approximation is that it enforces symmetry in every relation, i.e., $f(s, r, t) = (x_s \circ x_t)^\top w_r = (x_t \circ x_s)^\top w_r = f(t, r, s)$. However, empirically it has been shown to outperform the Bilinear model when the number of training examples is small (Yang et al., 2015).

use paths (s, π, t) to define the scoring function $f(s, r, t|\theta, \Pi_{s,t})$. Here $\Pi_{s,t}$ denotes the sum of the embeddings of a set of paths π between the two nodes in the graph, weighted by path-constrained random walk probabilities. Their implementation built on the TransE (Bordes et al., 2013) embedding model; in comparison, we formulate a similar model using the BILINEAR model.

We refer to such an approach as the PRUNED-PATHS model, for it discards paths with weights below a certain threshold. We define the model under a BILINEAR composition as follows: Let $\{\pi_1, \pi_2, \ldots, \pi_K\}$ denote a fixed set of path types that can be used as a source of features for the model. We abuse notation slightly to refer to a sequence of types of relation links $r_1, r_2, \ldots, r_n$ in a path in $\mathcal{G}$ as π. We denote by $P(t|s, \pi)$ the path-constrained random walk probability of reaching node t starting from node s and following sequences of relation types as specified in π. We define the weighted path representation $F(s, t)$ for a node pair as the weighted sum of the representations of paths π in the set $\{\pi_1, \pi_2, \ldots, \pi_K\}$ that have non-zero path-constrained random walk probabilities. The representation is defined as: $F(s, t) = \sum_\pi w_{|\pi|} P(t|s, \pi)\Phi(\pi)$, where $\Phi(\pi)$ is the path relation type representation from (eq. 3). The weights $w_{|\pi|}$ provide a shared parameter for paths of each length, so that the model may learn to trust the contribution of paths of different lengths differentially. The dependence on θ and the set of paths $\Pi_{s,t}$ between the two nodes in the graph was dropped for simplicity. Figure 2 illustrates the weighted path representation between nodes GRB2 and MAPK3 from Figure 1. [2]

Using the above definition, the score of a candidate triple $f(s, r, t|\theta, \Pi_{s,t})$ is defined as:

$$f(s, r, t) = x_s^\top W_r x_t + \text{vec}(F(s, t))^\top \text{vec}(W_r) \quad (4)$$

The first term of the scoring function is the same as that of the BILINEAR model, and the second term takes into account the similarity of the weighted path representations for (s, t) and the predicted relation r. Here we use element-wise product of the two matrices as the similarity metric. Training and scoring under this model requires explicit constructions of paths, and computing and storing the

[2]Unlike this illustration, the representations of dependency paths are not decomposed into representations of individual edges in our implementation.

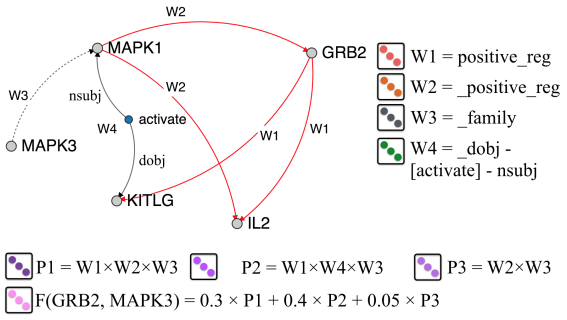


Figure 2: An illustration of the weighted sum of path representations for paths connecting GRB2 and MAPK3 from Figure 1.The prefix "_" of a relation type indicates an inverse relation. The path-constrained random walk probabilities for $P1$, $P2$ and $P3$ are 0.3, 0.4 and 0.05 respectively, and 0.0 for the rest. The path length weights are omitted.

random walk probabilities $P(t|s, \pi)$, which makes it expensive to scale. In the next section, we introduce an algorithm that can efficiently take into account all paths connecting two nodes, and naturally extend it to model the impact of intermediate nodes on the informativeness of the paths.

3.2 ALL-PATHS: A Novel Representation and Algorithm

We now introduce a novel algorithm for efficiently computing and learning the scoring function from (eq. 4), while summing over the set of all paths π up to a certain length L, and additionally modeling the impact of intermediate nodes on the representations $\Phi(\pi)$ of paths. Depending on the characteristics of the knowledge graph and the dimensionality of the learned embedding representations, this method in addition to being more exact, can be faster and take less memory than a method that explicitly generates and prunes full relation paths.

We first define a new representation function for paths of the form $(s, \pi, t) = (s, r_1, e_1, r_2, e_2 \ldots, e_{n-1}, r_n, t)$, which has as a special case the relation-type based function used in prior work, but can additionally model the impact of intermediate nodes e_j on the path representation.

We introduce new parameters w_{e_i} which can impact the representations of paths passing through nodes e_i. w_{e_i} is a scalar weight in our implementation, but vectors of dimensionality d could also be implemented using the same general approach. The embedding of the sample path π under a BILINEAR model is defined as: $\Phi_\pi = W_{r_1} \tanh(w_{e_1}) \cdots W_{r_n} \tanh(w_{e_n})$.

Here the weight of each node is first transformed into the range $[-1, 1]$ using a non-linear $\tanh$ function. To derive our exact algorithm, we

use quantities $F_l(s,t)$ denoting the weighted sum of path representations for all paths of length l between nodes s and t. The weighted sum of path representations $F(s,t)$ can be written as:

$$\sum_{l=1...L} w_l F_l(s,t), \tag{5}$$

where $F_l(s,t) = \sum_{\pi \in \mathcal{P}_l(s,t)} p(s|t,\pi)\Phi_\pi$ and $P_l(s,t)$ denotes all paths of length l between s,t.

Computing $F(s,t)$ in the naive way by enumerating all possible (s,t) paths is impractical since the number of possible paths can be very large, especially in graphs containing text. Therefore prior work selected a subset of paths through sampling and pruning (Neelakantan et al., 2015; Lin et al., 2015). However, the properties of the BILINEAR composition function for path representation enable us to incrementally build the sums of all path representations exactly, using dynamic programming. Algorithm 1 shows how to compute the necessary quantities $F_l(s,t)$ for all entity pairs in $\mathcal{G}$. [3]

Algorithm 1 Compute the sum of path representations (up to length L) for every entity pair (s,t).

Input : $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, $\{W_r | r \in \mathcal{R}\}$, list of all entity pairs $EP = \{(s,t)\}$.
Output : $F_L(s,t)$ for every $(s,t) \in EP$.
Initialize:
for $(s,t) \in EP$ **do**
  if $\mathcal{R}(s,t) \neq \emptyset$ **then**
    $F_1(s,t) \leftarrow \sum_{r \in \mathcal{R}(s,t)} \tanh(w_t) p(t|s,r) W_{r(s,t)}$
  else
    $F_1(s,t) \leftarrow \mathbf{0}$
  end
end
for $l = 2, \ldots, L$ **do**
  for $(s,t) \in EP$ **do**
    $F_l(s,t) \leftarrow \mathbf{0}$
    for $e \in \nu(s)$ **do**
      $F_l(s,t) \leftarrow F_l(s,t) + F_1(s,e) F_{l-1}(e,t)$
    end
  end
end

The key to this solution is that the representations of the longer paths are composed of those of the shorter paths and that the random walk probabilities of relation paths are products of transition probabilities over individual relation edges. The sub-components of paths frequently overlap. For example, all paths π depicted in Figure 2 share the same tail edge family. The algorithms from prior work perform a sum over product representations of paths, but it is more efficient to regroup these into a product of sums. This regrouping allows taking into account exponentially many possible paths without explicitly enumerating them and individually computing their path-constrained random walk probabilities.

After all quantities $F_l(s,t)$ are computed, their weighted sum $F(s,t)$ can be computed using $O(dL)$ operations for each entity pair. These can directly be used to compute the scores of all positive and negative triples for training, using (eq. 4). As we can see, no explicit enumeration or storage of multi-step relation paths between pairs of nodes is needed. To compute gradients of the loss function with respect to individual model parameters, we use a similar algorithm to compute the error for each intermediate edge (e_i, r', e_j) for each group of length l paths between s and t. The algorithm is a variant of forward-backward, corresponding to the forward Algorithm 1.

Notice that directly adding node representations for the paths in the PRUNED-PATHS approach would be infeasible since it would lead to an exposition in the possible path types and therefore the memory and running time requirements of the model. Thus the use of an adaptation of our dynamic programming algorithm to the task of summing over node sequences for a given path type would become necessary to augment the parametric family of the PRUNED-PATHS approach with node representations.

3.3 Efficiency Analysis

We perform a worst-case analysis of the time and memory required for training and evaluation for each of the introduced methods. For the models taking into account relation paths, we assume that all paths up to a certain length L will be modeled[4]. For the basic model text is not taken into account in training, since modeling text and KB relations uniformly in this model did not help performance as seen in the Experiments section. We only take text into account as a source of relation path features or auxiliary path facts.

Notation Let N_e denote the number of nodes in the knowledge graph, E_{kb} the number of knowl-

[3] We further speed up the computation by pre-computing, for each node s, the set of notes t reachable via paths of length l. Then we iterate over non-zero entries only, instead of all pairs in the loops.

[4] If only a subset of paths is used, the complexity of the methods other than ALL-PATHS will be reduced. However, in order to compare the methods in a similar setting, we analyze performance in the case of modeling all multi-step paths.

edge graph links/triples, E_{txt} the number of textual links/triples, a the average number of outgoing links for a node in the graph given the outgoing relation type, N_r the number of distinct relations in the graph, η the number of negative triples for each training triple, and d the dimensionality of entity embeddings and (diagional) matrix representations of relations.

BILINEAR-DIAG In this basic model, the training time is $\mathrm{O}\big(2d(\eta+1)E_{kb}\big)$ and memory is $\mathrm{O}\big(dN_e + dN_r\big)$. The memory required is for storing embeddings of entities and relations, and the time is for scoring $2(\eta+1)$ triples for every fact in the training KB.

Guu et al. (2015) This method generates training path triples $\{(x, \pi, y)\}$ for all entity pairs connected by paths π up to length L. The number of such triples[5] is $\mathcal{T} = E_{kb} + E_{txt} + \sum_{l=2\ldots L} {N_r}^l \times N_e \times a^l$. The memory required is the memory to store all triples and the set of relation paths, which is $\mathrm{O}\big(\mathcal{T} + \sum_{l=2\ldots L} l{N_r}^l\big)$. The time required includes the time to compute the scores and gradients for all triples and their negative examples (for subject and object position), as well as to compute the compositional path representations of all path types. The time to compute compositional path representations is $\mathrm{O}\big(d\sum_{l=2\ldots L} l{N_r}^l\big)$ and the time spent per triple is $2d(\eta+1)$ as in the basic model. Therefore the overall time per iteration is $\mathrm{O}\big(2d(\eta+1)\mathcal{T}\big)$+$\mathrm{O}\big(d\sum_{l=2\ldots L} l{N_r}^l\big)$. The test time and memory requirements of this method are the same as these of BILINEAR-DIAG, which is a substantial advantage over other methods, if evaluation-time efficiency is important.

PRUNED-PATHS This method computes and stores the values of the random walk probabilities for all pairs of nodes and relation paths, for which these probabilities are non-zero. This can be done in time $\mathrm{O}\big(\mathcal{T}\big)$ where Triples is the same quantity used in the analysis of Guu et al. (2015). The memory requirements of this method are the same as these of (Guu et al., 2015), up to a constant to store random-walk probabilities for paths.

The time requirements are different, however. At training time, we compute scores and update gradients for triples corresponding to direct knowledge base edges, whose number is E_{kb}. For each considered triple, however, we need to compute the sum of representations of path features that are active for the triple. We estimate the average number of active paths per node pair as $\frac{\mathcal{T}}{{N_e}^2}$. Therefore the overall time for this method per training iteration is $\mathrm{O}\big(2d(\eta+1)E_{kb}\frac{\mathcal{T}}{{N_e}^2}\big)$+$\mathrm{O}\big(d\sum_{l=2\ldots L} l{N_r}^l\big)$.

We should note that whether this method or the one of Guu et al. (2015) will be faster in training depends on whether the average number of paths per node pair multiplied by E_{kb} is bigger or smaller than the total number of triples $\mathcal{T}$. Unlike the method of Guu et al. (2015), the evaluation-time memory requirements of this approach are the same as its training memory requirements, or they could be reduced slightly to match the evaluation-time memory requirements of ALL-PATHS, if these are lower as determined by the specific problem instance.

ALL-PATHS This method does not explicitly construct or store fully constructed paths (s, π, t). Instead, memory and time is determined by the dynamic program in Algorithm 1, as well as the forward-backward algorithm for computation of gradients. The memory required to store path representation sums $F_l(s,t)$ is $\mathrm{O}\big(dL{N_e}^2\big)$ in the worst case. Denote $E = E_{kb} + E_{txt}$. The time to compute these sums is $\mathrm{O}\big(dE(1+\sum_{l=2\ldots L}(l-1)N_e)\big)$. After this computation, the time to compute the scores of training positive and negative triples is $\mathrm{O}\big(d2(\eta+1)E_{kb}L\big)$. The time to increment gradients using each triple considered in training is $\mathrm{O}\big(dEL^2\big)$. The evaluation time memory is reduced relative to training time memory by a factor of L and the evaluation time per triple can also be reduced by a factor of L using precomputation.

Based on this analysis, we computed training time and memory estimates for our NCI+Txt knowledge base. Given the values of the quantities from our knowledge graph and $d = 50$, $\eta = 50$, and maximum path length of 5, the estimated memory for (Guu et al., 2015) and PRUNED-PATHS is 4.0×10^{18} and for ALL-PATHS the memory is 1.9×10^9. The time estimates are 2.4×10^{21}, 2.6×10^{25}, and 7.3×10^{15} for (Guu et al., 2015), PRUNED-PATHS, and ALL-PATHS, respectively.

[5]The computation uses the fact that the number of path type sequences of length l is N_r^l. We use a, the average branching factor of nodes given relation types, to derive the estimated number of triples of a given relation type for a path of length l.

Model	KB		KB and Text	
	MAP	HITS@10	MAP	HITS@10
BILINEAR-DIAG (Guu et al., 2015) d=100	12.48	19.66	12.48	19.66
BILINEAR-DIAG d=100	28.56	39.92	28.56	39.92
BILINEAR-DIAG d=2000	30.16	42.51	30.16	42.51
+Guu et al. (2015) d=100 orig.	23.20	34.84	23.20	34.84
+Guu et al. (2015) d=100 reimpl.	29.13	40.59	30.25	41.45
PRUNED-PATHS d=100 c=1000	32.31	43.16	36.42	48.22
PRUNED-PATHS d=100 c=100	32.31	43.16	36.79	48.27
PRUNED-PATHS d=100 c=1	32.31	43.16	37.03	48.26
ALL-PATHS d=100	32.31	43.16	36.24	48.60
ALL-PATHS+NODES d=100	**33.92**	**45.96**	**39.31**	**52.53**

Table 1: KB completion results on NCI-PID test: comparison of our compositional learning approach (ALL-PATHS+NODES) with baseline systems. d is the embedding dimension; sampled paths occurring less than c times were pruned in PRUNED-PATHS.

4 Experiments

Our experiments are designed to study three research questions: (*i*) What is the impact of using path representations as a source of compositional regularization as in (Guu et al., 2015) versus using them as features for scoring as in PRUNED-PATHS and ALL-PATHS? (*ii*) What is the impact of using textual mentions for KB completion in different models? (*iii*) Does modeling intermediate path nodes improve the accuracy of KB completion?

Datasets We used two datasets for evaluation: NCI-PID and WordNet.

For the first set of experiments, we used the Pathway Interaction Database (NCI-PID) (Schaefer et al., 2009) as our knowledge base, which was created by editors from the Nature Publishing Groups, in collaboration with the National Cancer Institute. It contains a collection of high-quality gene regulatory networks (also referred to as *pathways*). The original networks are in the form of hypergraphs, where nodes could be complex gene products (e.g., "protein complex" with multiple proteins bound together) and regulations could have multiple inputs and outputs. Following the convention of most network modeling approaches, we simplified the hypergraphs into binary regulations between genes (e.g., GRB2 positive_reg MAPK3), which yields a graph with 2774 genes and 14323 triples. The triples are then split into train, dev, and test sets, of size 10224, 1315, 2784, respectively. We identified genes belonging to the same family via the common letter prefix in their names, which adds 1936 triples to training.

As a second dataset, we used a WordNet KB with the same train, dev, and test splits as Guu et al. (2015). There are 38,696 entities and 11 types of knowledge base relations. The KB includes 112,581 triples for training, 2,606 triples for validation, and 10,544 triples for testing. WordNet does not contain textual relations and is used for a more direct comparison with recent works.

Textual Relations We used PubMed abstracts for text for NCI-PID. We used the gene mentions identified by Literome (Poon et al., 2014), and considered sentences with co-occurring gene pairs from NCI-PID. We defined textual relations using the fully lexicalized dependency paths between two gene mentions, as proposed in Riedel et al. (2013). Additionally, we define trigger-mapped dependency paths, where only important "trigger" words are lexicalized and the rest of the words are replaced with a wild-card character X. A set of 333 words often associated with regulation events in Literome (e.g. *induce*, *inhibit*, *reduce*, *suppress*) were used as trigger words. To avoid introducing too much noise, we only included textual relations that occur at least 5 times between mentions of two genes that have a KB relation. This resulted in 3,827 distinct textual relations and 1,244,186 mentions.[6] The number of textual relations is much larger than that of KB relations, and it helped induce much larger connectivity among genes (390,338 pairs of genes are directly connected in text versus 12,100 pairs in KB).

Systems ALL-PATHS denotes our compositional learning approach that sums over all paths using

[6]Modeling such a large number of textual relations introduces sparsity, which necessitates models such as (Toutanova et al., 2015; Verga et al., 2015) to derive composed representations of text. We leave integration with such methods for future work.

dynamic programming; ALL-PATHS+NODES additionally models nodes in the paths. PRUNED-PATHS denotes the traditional approach that learns from sampled paths detailed in §3.1.2; paths with occurrence less than a cutoff are pruned ($c = 1$ in Table 1 means that all sampled paths are used). The most relevant prior approach is Guu et al. (2015). We ran experiments using both their publicly available code and our re-implementation. We also included the BILINEAR-DIAG baseline.

Implementation Details We used batch training with RProp (Riedmiller and Braun, 1993). The L_2 penalty λ was set to 0.1 for all models, and the entity vectors x_e were normalized to unit vectors. For each positive example we sample 500 negative examples. For our implementation of (Guu et al., 2015), we run 5 random walks of each length starting from each node and we found that adding a weight β to the multi-step path triples improves the results. After preliminary experimentation, we fixed β to 0.1. Models using KB and textual relations were initialized from models using KB relations only[7]. Model training was stopped when the development set MAP did not improve for 40 iterations; the parameters with the best MAP on the development set were selected as output. Finally, we used only paths of length up to 3 for NCI-PID and up to length 5 for WordNet.[8]

Evaluation metrics We evaluate our models on their ability to predict the subjects/objects of knowledge base triples in the test set. Since the relationships in the gene regulation network are frequently not one-to-one, we use the mean average precision (MAP) measure instead of the mean reciprocal rank often used in knowledge base completion works in other domains. In addition to MAP, we use Hits@10, which is the percentage of correct arguments ranked among the top 10 predictions[9]. We compute measures for ranking both the object entities $(s, r, *)$ and the subject entities$(*, r, t)$. We report evaluation metrics computed on the union of the two query set.

NCI-PID Results Table 1 summarizes the knowledge base (KB) completion results on the NCI-PID test. The rows compare our compositional learning approach ALL-PATHS+NODES with prior approaches. The comparison of the two columns demonstrates the impact when text is jointly embedded with KB. Our compositional learning approach significantly outperforms all other approaches in both evaluation metrics (MAP and HITS@10). Moreover, jointly embedding text and KB led to substantial improvement, compared to embedding KB only.[10] Finally, modeling nodes in the paths offers significant gains (ALL-PATHS+NODE gains 3 points in MAP over ALL-PATHS), with statistical significance ($p < .001$) according to a McNemar test.

Evaluating the effect of path pruning on the traditional approach (PRUNED-PATHS) is quite illuminating. As the number of KB relations is relatively small in this domain, when only KB is embedded, most paths occur frequently. So there is little difference between the heaviest pruned version (c=1000) and the lightest (c=1). When textual relations are included, the cutoff matters more, although the difference was small as many rarer textual relations were already filtered beforehand. In either case, the accuracy difference between ALL-PATHS and PRUNED-PATHS is small, and ALL-PATHS mainly gains in efficiency. However, when nodes are modeled, the compositional learning approach gains in accuracy as well, especially when text is jointly embedded.

Comparison among the baselines also offers valuable insights. The implementation of Guu et al. (2015) with default parameters performed significantly worse than our re-implementation. Also, our re-implementation achieves only a slight gain over the BILINEAR-DIAG baseline, whereas the original implementation obtains substantial improvement over its own version of BILINEAR-DIAG. These results underscore the importance of hyper-parameters and optimization, and invite future systematic research on the impact of such modeling choices.[11]

[7] In addition, models using path training were initialized from models trained on direct relation links only.

[8] Using paths up to length 4 on NCI-PID did not perform better.

[9] As a common practice in KB completion evaluation, for both MAP and Hits@10, we filtered out the other correct answers when ranking a particular triple to eliminate ranking penalization induced by other correct predictions.

[10] Text did not help for the models in the first four rows in the Table, possibly because in these approaches text and KB information are equally weighted in the loss function and the more numerous textual triples dominate the KB ones.

[11] The differences between our two implementations are: max-margin loss versus softmax loss and stochastic gradient training versus batch training.

Model	MAP	HITS@10
BILINEAR-DIAG (Guu et al., 2015)	N/A	12.9
BILINEAR-DIAG	8.0	12.2
+Guu et al. (2015)	N/A	14.4
PRUNED-PATHS $l = 3$ c=10	9.5	14.8
PRUNED-PATHS $l = 3$ c=1	9.5	14.9
PRUNED-PATHS $l = 5$ c=10	8.9	14.4
ALL-PATHS $l = 3$	9.4	14.7
ALL-PATHS+NODES l=3	9.4	15.2
ALL-PATHS $l = 5$	9.6	16.6
ALL-PATHS+NODES l=5	**9.8**	**16.7**

Table 2: KB completion results on the WordNet test set: comparison of our compositional learning approach (ALL-PATHS) with baseline systems. The maximum length of paths is denoted by l. Sampled paths occurring less than c times were pruned in PRUNED-PATHS.

WordNet Results Table 2 presents a comparative evaluation on the WordNet dataset. This dataset has a larger number of knowledge base relation types compared to NCI-PID, and longer relation paths in this KB are expected to be beneficial. Guu et al. (2015) evaluated their compositional regularization approach on this dataset and we can directly compare to their results. The first two rows in the Table show the baseline BILINEAR-DIAG model results according to the results reported in (Guu et al., 2015) and our implementation. The MAP results were not reported in Guu et al. (2015); hence the NA value for MAP in row one.[12] On this dataset, our implementation of the baseline model does not have substantially different results than Guu et al. (2015) and we use their reported results for the baseline and compositionally trained model.

Compositional training improved performance in Hits@10 from 12.9 to 14.4 in Guu et al. (2015), and we find that using PRUNED-PATHS as features gives similar, but a bit higher performance gains.

The PRUNED-PATHS method is evaluated using count cutoffs of 1 and 10, and maximum path lengths of 3 and 5. As can be seen, lower count cutoff performed better for paths up to length 3, but we could not run the method with path lengths up to 5 and count cutoff of 1, due to excessive memory requirements (more than 248GB). When using count cutoff of 10, paths up to length 5 performed worse than paths up to length 3. This performance degradation could be avoided with a staged training regiment where models with shorter paths are first trained and used to initialize models using longer paths.

The performance of the ALL-PATHS method can be seen for maximum paths up to lengths 3 and 5, and with or without using features on intermediate path nodes.[13] As shown in Table 2, longer paths were useful, and features on intermediate nodes were also beneficial. We tested the significance of the differences between several pairs of models and found that nodes led to significant improvement ($p < .002$) for paths of length up to 3, but not for the setting with longer paths. All models using path features are significantly better than the baseline BILINEAR-DIAG model.

To summarize both sets of experiments, the ALL-PATHS approach allows us to efficiently include information from long KB relation paths as in WordNet, or paths including both text and KB relations as in NCI-PID. Our dynamic programming algorithm considers relation paths efficiently, and is also straightforwardly generalizable to include modeling of intermediate path nodes, which would not be directly possible for the PRUNED-PATHS approach. Using intermediate nodes was beneficial on both datasets, and especially when paths could include textual relations as in the NCI-PID dataset.

5 Conclusions

In this work, we propose the first approach to efficiently incorporate all relation paths of bounded length in a knowledge base, while modeling both relations and intermediate nodes in the compositional path representations. Experimental results on two datasets show that it outperforms prior approaches by modeling intermediate path nodes. In the future, we would like to study the impact of relation paths for additional basic KB embedding models and knowledge domains.

Acknowledgments

The authors would like to thank the anonymous reviewers and meta-reviewer for suggestions on making the paper stronger, as well as Kenton Lee and Aria Haghighi for their helpful comments on the draft.

[12] We ran the trained model distributed by Guu et al. (2015) and obtained a much lower Hits@10 value of 6.4 and MAP of of 3.5. Due to the discrepancy, we report the original results from the authors' paper which lack MAP values instead.

[13] To handle the larger scale of the WordNet dataset in terms of the number of nodes in the knowledge base, we enforced a maximum limit (of 100) on the degree of a node that can be an intermediate node in a multi-step relation path.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

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

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems (NIPS)*.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Alberto Garcia-Duran, Antoine Bordes, and Nicolas Usunier. 2015. Composing relationships with translations. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 286–290, Lisbon, Portugal, September. Association for Computational Linguistics.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Kelvin Guu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327, Lisbon, Portugal, September. Association for Computational Linguistics.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling relation paths for representation learning of knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714, Lisbon, Portugal, September. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Hoifung Poon, Chris Quirk, Charlie DeZiel, and David Heckerman. 2014. Literome: Pubmed-scale genomic knowledge base in the cloud. *Bioinformatics*, 30(19):2840–2842.

Hoifung Poon, Kristina Toutanova, and Chris Quirk. 2015. Distant supervision for cancer pathway extraction from text. In *PSB*.

Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. 2013. Relation extraction with matrix factorization and universal schemas. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE.

Carl F Schaefer, Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H Buetow. 2009. PID: the pathway interaction database. *Nucleic acids research*, 37(suppl 1):D674–D679.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems (NIPS)*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1045–1055. International World Wide Web Conferences Steering Committee.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66. Association for Computational Linguistics.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2015. Multilingual relation extraction using compositional universal schema. *arxiv*, abs/1511.06396.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July. Association for Computational Linguistics.

Commonsense Knowledge Base Completion

Xiang Li[*‡] **Aynaz Taheri**[†] **Lifu Tu**[‡] **Kevin Gimpel**[‡]

[*]University of Chicago, Chicago, IL, 60637, USA
[†]University of Illinois at Chicago, Chicago, IL, 60607, USA
[‡]Toyota Technological Institute at Chicago, Chicago, IL, 60637, USA

lix1@uchicago.edu, ataher2@uic.edu, {lifu,kgimpel}@ttic.edu

Abstract

We enrich a curated resource of commonsense knowledge by formulating the problem as one of knowledge base completion (KBC). Most work in KBC focuses on knowledge bases like Freebase that relate entities drawn from a fixed set. However, the tuples in ConceptNet (Speer and Havasi, 2012) define relations between an unbounded set of *phrases*. We develop neural network models for scoring tuples on arbitrary phrases and evaluate them by their ability to distinguish true held-out tuples from false ones. We find strong performance from a bilinear model using a simple additive architecture to model phrases. We manually evaluate our trained model's ability to assign quality scores to novel tuples, finding that it can propose tuples at the same quality level as medium-confidence tuples from ConceptNet.

1 Introduction

Many ambiguities in natural language processing (NLP) can be resolved by using knowledge of various forms. Our focus is on the type of knowledge that is often referred to as "commonsense" or "background" knowledge. This knowledge is rarely expressed explicitly in textual corpora (Gordon and Van Durme, 2013). Some researchers have developed techniques for inferring this knowledge from patterns in raw text (Gordon, 2014; Angeli and Manning, 2014), while others have developed curated resources of commonsense knowledge via manual annotation (Lenat and Guha, 1989; Speer and Havasi, 2012) or games with a purpose (von Ahn et al., 2006).

Curated resources typically have high precision but suffer from a lack of coverage. For certain resources, researchers have developed methods to automatically increase coverage by inferring missing entries. These methods are commonly categorized under the heading of knowledge base completion (KBC). KBC is widely-studied for knowledge bases like Freebase (Bollacker et al., 2008) which contain large sets of entities and relations among them (Mintz et al., 2009; Nickel et al., 2011; Riedel et al., 2013; West et al., 2014), including recent work using neural networks (Socher et al., 2013; Yang et al., 2014).

relation	right term	conf.
MOTIVATEDBYGOAL	relax	3.3
USEDFOR	relaxation	2.6
MOTIVATEDBYGOAL	your muscle be sore	2.3
HASPREREQUISITE	go to spa	2.0
CAUSES	get pruny skin	1.6
HASPREREQUISITE	change into swim suit	1.6

Table 1: ConceptNet tuples with left term "soak in hotspring"; final column is confidence score.

We improve the coverage of commonsense resources by formulating the problem as one of knowledge base completion. We focus on a particular curated commonsense resource called ConceptNet (Speer and Havasi, 2012). ConceptNet contains tuples consisting of a left term, a relation, and a right term. The relations come from a fixed set. While terms in Freebase tuples are entities, ConceptNet terms can be arbitrary *phrases*. Some examples are shown in Table 1. An NLP application may wish to query ConceptNet for information about soaking in a hotspring, but may use different words from those contained in the ConceptNet tuples. Our goal is to do on-the-fly knowledge base completion so that queries can be answered robustly without requiring the precise linguistic forms contained in ConceptNet.

To do this, we develop neural network models to embed terms and provide scores to arbi-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1445–1455,
Berlin, Germany, August 7-12, 2016.

trary tuples. We train them on ConceptNet tuples and evaluate them by their ability to distinguish true and false held-out tuples. We consider several functional architectures, comparing two composition functions for embedding terms and two functions for converting term embeddings into tuple scores. We find that all architectures are able to outperform several baselines and reach similar performance on classifying held-out tuples.

We also experiment with several training objectives for KBC, finding that a simple cross entropy objective with randomly-generated negative examples performs best while also being fastest. We manually evaluate our trained model's ability to assign quality scores to novel tuples, finding that it can propose tuples at the same quality level as medium-confidence tuples from ConceptNet. We release all of our resources, including our ConceptNet KBC task data, large sets of randomly-generated tuples scored with our model, training code, and pretrained models with code for calculating the confidence of novel tuples.[1]

2 Related Work

Our methods are similar to past work on KBC (Mintz et al., 2009; Nickel et al., 2011; Lao et al., 2011; Nickel et al., 2012; Riedel et al., 2013; Gardner et al., 2014; West et al., 2014), particularly methods based on distributed representations and neural networks (Socher et al., 2013; Bordes et al., 2013; Bordes et al., 2014a; Bordes et al., 2014b; Yang et al., 2014; Neelakantan et al., 2015; Gu et al., 2015; Toutanova et al., 2015). Most prior work predicts new relational links between terms drawn from a *fixed* set. In a notable exception, Neelakantan and Chang (2015) add new entities to KBs using external resources along with properties of the KB itself. Relatedly, Yao et al. (2013) induce an unbounded set of entity categories and associate them with entities in KBs.

Several researchers have developed techniques for discovering commonsense knowledge from text (Gordon et al., 2010; Gordon and Schubert, 2012; Gordon, 2014; Angeli and Manning, 2014). Open information extraction systems like REVERB (Fader et al., 2011) and NELL (Carlson et al., 2010) find tuples with arbitrary terms and relations from raw text. In contrast, we start with a set of commonsense facts to use for training, though our methods could be applied to the output of these or other extraction systems.

Our goals are similar to those of the AnalogySpace method (Speer et al., 2008), which uses matrix factorization to improve coverage of ConceptNet. However, AnalogySpace can only return a confidence score for a pair of terms drawn from the training set. Our models can assign scores to tuples that contain novel terms (as long as they consist of words in our vocabulary).

Though we use ConceptNet, similar techniques can be applied to other curated resources like WordNet (Miller, 1995) and FrameNet (Baker et al., 1998). For WordNet, tuples can contain lexical entries that are linked via synset relations (e.g., "hypernym"). WordNet contains many multiword entries (e.g., "cold sweat"), which can be modeled compositionally by our term models; alternatively, entire glosses could be used as terms. To expand frame relationships in FrameNet, tuples can draw relations from the frame relation types (e.g., "is causative of") and terms can be frame lexical units or their definitions.

Several researchers have used commonsense knowledge to improve language technologies, including sentiment analysis (Cambria et al., 2012; Agarwal et al., 2015), semantic similarity (Caro et al., 2015), and speech recognition (Lieberman et al., 2005). Our hope is that our models can enable many other NLP applications to benefit from commonsense knowledge.

Our work is most similar to that of Angeli and Manning (2013). They also developed methods to assess the plausibility of new facts based on a training set of facts, considering commonsense data from ConceptNet in one of their settings. Like us, they can handle an unbounded set of terms by using (simple) composition functions for novel terms, which is rare among work in KBC. One key difference is that their best method requires iterating over the KB at test time, which can be computationally expensive with large KBs. Our models do not require iterating over the training set. We compare to several baselines inspired by their work, and we additionally evaluate our model's ability to score novel tuples derived from both ConceptNet and Wikipedia.

3 Models

Our goal is to represent commonsense knowledge such that it can be used for NLP tasks. We as-

[1] Available at `http://ttic.uchicago.edu/~kgimpel/commonsense.html`.

sume this knowledge is given in the form of tuples $\langle t_1, R, t_2 \rangle$, where t_1 is the **left term**, t_2 is the **right term**, and R is a (directed) **relation** that exists between the terms. Examples are shown in Table 1.[2]

Given a set of tuples, our goal is to develop a parametric model that can provide a confidence score for new, unseen tuples. That is, we want to design and train models that define a function $\text{score}(t_1, R, t_2)$ that provides a quality score for an arbitrary tuple $\langle t_1, R, t_2 \rangle$. These models will be evaluated by their ability to distinguish true held-out tuples from false ones.

We describe two model families for scoring tuples. We assume that we have embeddings for words and define models that use these word embeddings to score tuples. So our models are limited to tuples in which terms consist of words in the word embedding vocabulary, though future work could consider character-based architectures for open-vocabulary modeling (Huang et al., 2013; Ling et al., 2015).

3.1 Bilinear Models

We first consider bilinear models, since they have been found useful for KBC in past work (Nickel et al., 2011; Jenatton et al., 2012; García-Durán et al., 2014; Yang et al., 2014). A bilinear model has the following form for a tuple $\langle t_1, R, t_2 \rangle$:

$$v_1^\top M_R \, v_2$$

where $v_1 \in \mathbb{R}^r$ is the (column) vector representing t_1, $v_2 \in \mathbb{R}^r$ is the vector for t_2, and $M_R \in \mathbb{R}^{r \times r}$ is the parameter matrix for relation R.

To convert terms t_1 and t_2 into term vectors v_1 and v_2, we consider two possibilities: word averaging and a bidirectional long short-term memory (LSTM) recurrent neural network (Hochreiter and Schmidhuber, 1997). This provides us with two models: **Bilinear AVG** and **Bilinear LSTM**.

One downside of this architecture is that as the length of the term vectors grows, the size of the relation matrices grows quadratically. This can slow down training while requiring more data to learn the large numbers of parameters in the matrices. To address this, we include an additional nonlinear transformation of each term:

$$u_i = a(W^{(B)} v_i + b^{(B)})$$

where a is a nonlinear activation function (tuned among ReLU, $\tanh$, and logistic sigmoid) and where we have introduced additional parameters $W^{(B)}$ and $b^{(B)}$. This gives us the following model:

$$\text{score}_{\text{bilinear}}(t_1, R, t_2) = u_1^\top M_R \, u_2$$

When using the LSTM, we tune the decision about how to produce the final term vectors to pass to the bilinear model, including possibly using the final vectors from each direction and the output of max or average pooling. We use the same LSTM parameters for each term.

3.2 Deep Neural Network Models

Our second family of models is based on deep neural networks (DNNs). While bilinear models have been shown to work well for KBC, their functional form makes restrictions about how terms can interact. DNNs make no such restrictions.

As above, we define two models, one based on using word averaging for the term model (**DNN AVG**) and one based on LSTMs (**DNN LSTM**). For the DNN AVG model, we obtain the term vectors v_1 and v_2 by averaging word vectors in the respective terms. We then concatenate v_1, v_2, and a relation vector v_R to form the input of the DNN, denoted v_{in}. The DNN uses a single hidden layer:

$$\begin{aligned} u &= a(W^{(D1)} v_{in} + b^{(D1)}) \\ \text{score}_{\text{DNN}}(t_1, R, t_2) &= W^{(D2)} u + b^{(D2)} \end{aligned} \quad (1)$$

where a is again a (tuned) nonlinear activation function. The size of the hidden vector u is tuned, but the output dimensionality (the numbers of rows in $W^{(D2)}$ and $b^{(D2)}$) is fixed to 1. We do not use a nonlinear activation for the final layer since our goal is to output a scalar score.

For the DNN LSTM model, we first create a single vector for the two terms using an LSTM. That is, we concatenate t_1, a delimiter token, and t_2 to create a single word sequence. We use a bidirectional LSTM to convert this word sequence to a vector, again possibly using pooling (the decision is tuned; details below). We concatenate the output of this bidirectional LSTM with the relation vector v_r to create the DNN input vector v_{in}, then use Eq. 1 to obtain a score. We found this to work better than separately using an LSTM on each term. We can not try this for the Bilinear LSTM model since its functional form separates the two term vectors.

[2] These examples are from the Open Mind Common Sense (OMCS) part of ConceptNet version 5 (Speer and Havasi, 2012). In our experiments below, we only use OMCS tuples.

The relation vectors v_R are learned in addition to the DNN parameters $W^{(D1)}$, $W^{(D2)}$, $b^{(D1)}$, $b^{(D2)}$, and the LSTM parameters (in the case of the DNN LSTM model). Also, word embedding parameters are updated in all settings.

4 Training

Given a tuple training set T, we train our models using two different loss functions: hinge loss and a binary cross entropy function. Both rely on ways of generating negative examples (Section 4.3). Both also use regularization (Section 4.4).

4.1 Hinge Loss

Given a training tuple $\tau = \langle t_1, R, t_2 \rangle$, the hinge loss seeks to make the score of τ larger than the score of negative examples by a margin of at least γ. This corresponds to minimizing the following loss, summed over all examples $\tau \in T$:

$$\begin{aligned}&\text{loss}_{\text{hinge}}(\tau) = \\ &\quad \max\{0, \gamma - \text{score}(\tau) + \text{score}(\tau_{neg(t_1)})\} \\ &\quad + \max\{0, \gamma - \text{score}(\tau) + \text{score}(\tau_{neg(R)})\} \\ &\quad + \max\{0, \gamma - \text{score}(\tau) + \text{score}(\tau_{neg(t_2)})\}\end{aligned}$$

where $\tau_{neg(t_1)}$ is the negative example obtained by replacing t_1 in τ with some other t_1, and $\tau_{neg(R)}$ and $\tau_{neg(t_2)}$ are defined analogously for the relation and right term. We describe how we generate these negative examples in Section 4.3 below.

4.2 Binary Cross Entropy

Though we only have true tuples in our training set, we can create a binary classification problem by assigning a label of 1 to training tuples and a label of 0 to negative examples. Then we can minimize cross entropy (CE) as is common when using neural networks for classification. To generate negative examples, we consider the methods described in Section 4.3 below. We also need to convert our models' scores into probabilities, which we do by using a logistic sigmoid σ on score. We denote the label as ℓ, where the label is 1 if the tuple is from the training set and 0 if it is a negative example. Then the loss is defined:

$$\begin{aligned}&\text{loss}_{\text{CE}}(\tau, \ell) = \\ &-\ell \log \sigma(\text{score}(\tau)) - (1-\ell) \log(1 - \sigma(\text{score}(\tau)))\end{aligned}$$

When using this loss, we generate three negative examples for each positive example (one for swapping each component of the tuple, as in the hinge loss). For a mini-batch of size β, there are β positive examples and 3β negative examples used for training. The loss is summed over these 4β examples yielded by each mini-batch.

4.3 Negative Examples

For the loss functions above, we need ways of automatically generating negative examples. For efficiency, we consider using the current mini-batch only, as our models are trained using optimization on mini-batches. We consider the following three strategies to construct negative examples. Each strategy constructs three negative examples for each positive example τ: one by replacing t_1, one by replacing R, and one by replacing t_2.

Random sampling. We create the three negative examples for τ by replacing each component with its counterpart in a randomly-chosen tuple in the same mini-batch.

Max sampling. We create the three negative examples for τ by replacing each component with its counterpart in some other tuple in the mini-batch, choosing the substitution to maximize the score of the resulting negative example. For example, when swapping out t_1 in $\tau = \langle t_1, R, t_2 \rangle$, we choose the substitution t_1' as follows:

$$t_1' = \underset{t:\langle t, R', t_2' \rangle \in \mu \setminus \tau}{\text{argmax}} \text{score}(t, R, t_2)$$

where μ is the current mini-batch of tuples. We perform the analogous procedure for R and t_2.

Mix sampling. This is a mixture of the above, using random sampling 50% of the time and max sampling the remaining 50% of the time.

4.4 Regularization

We use L_2 regularization. For the DNN models, we add the penalty term $\lambda \|\theta\|^2$ to the losses, where λ is the regularization coefficient and θ contains all other parameters. However, for the bilinear models we regularize the relation matrices M_R toward the identity matrix instead of all zeroes, adding the following to the loss:

$$\lambda_1 \|\theta\|^2 + \lambda_2 \sum_R \|M_R - I_r\|_2^2$$

where I_r is the $r \times r$ identity matrix, the summation is performed over all relations R, and θ represents all other parameters.

5 Experimental Setup

We now evaluate our tuple models. We measure whether our models can distinguish true and false tuples by training a model on a large set of tuples and testing on a held-out set.

5.1 Task Design

The tuples are obtained from the Open Mind Common Sense (OMCS) entries in the ConceptNet 5 dataset (Speer and Havasi, 2012). They are sorted by a confidence score. The most confident 1200 tuples were reserved for creating our test set (TEST). The next most confident 600 tuples (i.e., those numbered 1201–1800) were used to build a development set (DEV1) and the next most confident 600 (those numbered 1801–2400) were used to build a second development set (DEV2).

For each set S ($S \in \{\text{DEV1}, \text{DEV2}, \text{TEST}\}$), for each tuple $\tau \in S$, we created a negative example and added it to S. So each set doubled in size. To create a negative example from $\tau \in S$, we randomly swapped one of the components of τ with another tuple $\tau' \in S$. One third of the time we swapped t_1 in τ for t_1 in τ', one third of the time we swapped their R's, and the remaining third of the time we swapped their t_2's. Thus, distinguishing positive and negative examples in this task is similar to the objectives optimized during training.

Each of DEV1 and DEV2 has 1200 tuples (600 positive examples and 600 negative examples), while TEST has 2400 tuples (1200 positive and 1200 negative). For training data, we selected 100,000 tuples from the remaining tuples (numbered 2401 and beyond).

The task is to separate the true and false tuples in our test set. That is, the labels are 1 for true tuples and 0 for false tuples. Given a model for scoring tuples, we select a threshold by maximizing accuracy on DEV1 and report accuracies on DEV2. This is akin to learning the bias feature weight (using DEV1) of a linear classifier that uses our model's score as its only feature. We tuned several choices—including word embeddings, hyperparameter values, and training objectives—on DEV2 and report final performance on TEST. One annotator (a native English speaker) attempted the same classification task on a sample of 100 tuples from DEV2 and achieved an accuracy of 95%. We release these datasets to the community so that others can work on this same task.

5.2 Word Embeddings

Our tuple models rely on initial word embeddings. To help our models better capture the commonsense knowledge in ConceptNet, we generated word embedding training data using the OMCS sentences underlying our training tuples (we excluded the top 2400 tuples which were used for creating DEV1, DEV2, and TEST). We created training data by merging the information in the tuples and their OMCS sentences. Our goal was to combine the grammatical context of the OMCS sentences with the words in the actual terms, so as to ensure that we learn embeddings for the words in the terms. We also insert the relations into the OMCS sentences so that we can learn embeddings for the relations themselves.

We describe the procedure by example and also release our generated data for ease of replication. The tuple ⟨soak in a hotspring, CAUSES, get pruny skin⟩ was automatically extracted/normalized (by the ConceptNet developers) from the OMCS sentence "The effect of [soaking in a hotspring] is [getting pruny skin]" where brackets surround terms. We replace the bracketed portions with their corresponding terms and insert the relation between them: "The effect of soak in a hotspring CAUSES get pruny skin". We do this for all training tuples.[3]

We used the `word2vec` (Mikolov et al., 2013) toolkit to train skip-gram word embeddings on this data. We trained for 20 iterations, using a dimensionality of 200 and a window size of 5. We refer to these as "CN-trained" embeddings for the remainder of this paper. Similar approaches have been used to learn embeddings for particular downstream tasks, e.g., dependency parsing (Bansal et al., 2014). We use our CN-trained embeddings within baseline methods and also provide the initial word embeddings of our models. For all of our models, we update the initial word embeddings during learning.

In the baseline methods described below, we compare our CN-trained embeddings to pretrained word embeddings. We use the GloVe (Pennington et al., 2014) embeddings trained on 840 billion tokens of Common Crawl web text and the PARAGRAM-SimLex embeddings of Wieting et al. (2015), which were tuned to have strong performance on the SimLex-999 task (Hill et al., 2015).

[3]For reversed relations, indicated by an asterisk in the OMCS sentences, we swap t_1 and t_2 in the tuple.

5.3 Baselines

We consider three baselines inspired by those of Angeli and Manning (2013):

- **Similar Fact Count (Count)**: For each tuple $\tau = \langle t_1, R, t_2 \rangle$ in the evaluation set, we count the number of similar tuples in the training set. A training tuple $\tau' = \langle t_1', R', t_2' \rangle$ is considered "similar" to τ if $R = R'$, one of the terms matches exactly, and the other term has the same head word. That is, $(R = R') \wedge (t_1 = t_1') \wedge (head(t_2) = head(t_2'))$, or $(R = R') \wedge (t_2 = t_2') \wedge (head(t_1) = head(t_1'))$. The head word for a term was obtained by running the Stanford Parser (Klein and Manning, 2003) on the term. This baseline does not use word embeddings.
- **Argument Similarity (ArgSim)**: This baseline computes the cosine similarity of the vectors for t_1 and t_2, ignoring the relation. Vectors for t_1 and t_2 are obtained by word averaging.
- **Max Similarity (MaxSim)**: For tuple τ in an evaluation set, this baseline outputs the maximum similarity between τ and any tuple in the training set. The similarity is computed by concatenating the vectors for t_1, R, and t_2, then computing cosine similarity. As in ArgSim, we obtain vectors for terms by averaging their words. We only consider R when using our CN-trained embeddings since they contain embeddings for the relations. When using GloVe and PARAGRAM embeddings for this baseline, we simply use the two term vectors (still constructed via averaging the words in each term).

We chose these baselines because they can all handle unbounded term sets but differ in their other requirements. ArgSim and MaxSim use word embeddings while Count does not. Count and MaxSim require iterating over the training set during inference while ArgSim does not.

For each baseline, we tuned a threshold on DEV1 to maximize classification accuracy then tested on DEV2 and TEST.

5.4 Training and Tuning

We used AdaGrad (Duchi et al., 2011) for optimization, training for 20 epochs through the training tuples. We separately tuned hyperparameters for each model and training objective. We tuned the following hyperparameters: the relation matrix size r for the bilinear models (also the length of the transformed term vectors, denoted u_1 and u_2 above), the activation a, the hidden layer size g for the DNN models, the relation vector length d for the DNN models, the LSTM hidden vector size h for models with LSTMs, the mini-batch size β, the regularization parameters λ, λ_1, and λ_2, and the AdaGrad learning rate α.

All tuning used early stopping: periodically during training, we used the current model to find the optimal threshold on DEV1 and evaluated on DEV2. Due to computational limitations, we were unable to perform thorough grid searches for all hyperparameters. We combined limited grid searches with greedy hyperparameter tuning based on regions of values that were the most promising.

For the Bilinear LSTM and DNN LSTM, we did hyperparameter tuning by training on the full training set of 100,000 tuples for 20 epochs, computing DEV2 accuracy once per epoch. For the averaging models, we tuned by training on a subset of 1000 tuples with $\beta = 200$ for 20 epochs; the averaging models showed more stable results and did not require the full training set for tuning. Below are the tuned hyperparameter values:

- **Bilinear AVG**: for CE: $r = 150$, $a = \tanh$, $\beta = 200$, $\alpha = 0.01$, $\lambda_1 = \lambda_2 = 0.001$. Hinge loss: same values as above except $\alpha = 0.005$.
- **Bilinear LSTM**: for CE: $r = 50$, $a =$ ReLU, $h = 200$, $\beta = 800$, $\alpha = 0.02$, and $\lambda_1 = \lambda_2 = 0.00001$. To obtain vectors from the term bidirectional LSTMs, we used max pooling. For hinge loss: $r = 50$, $a = \tanh$, $h = 200$, $\beta = 400$, $\alpha = 0.007$, $\lambda_1 = 0.00001$, and $\lambda_2 = 0.01$. To obtain vectors from the term bidirectional LSTMs, we used the concatenation of average pooling and final hidden vectors in each direction. For each sampling method and loss function, α was tuned by grid search with the others fixed to the above values.
- **DNN AVG**: for both losses: $a =$ ReLU, $d = 200$, $g = 1000$, $\beta = 600$, $\alpha = 0.01$, $\lambda = 0.001$.
- **DNN LSTM**: for both losses: $a =$ ReLU, $d = 200$, bidirectional LSTM hidden layer size $h = 200$, hidden layer dimension $g = 800$, $\beta = 400$, $\alpha = 0.005$, and $\lambda = 0.00005$. To get vectors from the term LSTMs, we used max pooling.

6 Results

Word Embedding Comparison. We first evaluate the quality of our word embeddings trained on the ConceptNet training tuples. Table 2 compares

	GloVe	PARAGRAM	CN-trained
ArgSim	68	69	**73**
MaxSim	73	70	**82**

Table 2: Accuracies (%) on DEV2 of two baselines using three different sets of word embeddings. Our ConceptNet-trained embeddings outperform GloVe and PARAGRAM embeddings.

	DNN AVG		DNN LSTM	
	CE	hinge	CE	hinge
random	124	230	710	783
mix	20755	21045	25928	26380
max	39338	41867	49583	49427

Table 4: Loss function runtime comparison (seconds per epoch) of the DNN models.

accuracies on DEV2 for the two baselines that use embeddings: ArgSim and MaxSim. We find that pretrained GloVe and PARAGRAM embeddings perform comparably, but both are outperformed by our ConceptNet-trained embeddings. We use the latter for the remaining experiments in this paper.

Training Comparison. Table 3 shows the results of our models with the two loss functions and three sampling strategies. We find that the binary cross entropy loss with random sampling performs best across models. We note that our conclusion differs from some prior work that found max or mix sampling to be better than random (Wieting et al., 2016). We suspect that this difference may stem from characteristics of the ConceptNet training data. It may often be the case that the max-scoring negative example in the mini-batch is actually a true fact, due to the generic nature of the facts expressed.

Table 4 shows a runtime comparison of the losses and sampling strategies.[4] We find random sampling to be orders of magnitude faster than the others while also performing the best.

Final Results. Our final results are shown in Table 5. We show the DEV2 and TEST accuracies for our baselines and for the best configuration (tuned on DEV2) for each model. All models outperform all baselines handily. Our models perform similarly, with the Bilinear models and the DNN AVG model all exceeding 90% on both DEV2 and TEST.

We note that the AVG models performed strongly compared to those that used LSTMs for

[4]These experiments were performed using 2 threads on a 3.40-GHz Intel Core i7-3770 CPU with 8 cores.

	DEV2	TEST
Count	75.4	79.0
ArgSim	72.9	74.2
MaxSim	81.9	83.5
Bilinear AVG	90.3	91.7
Bilinear LSTM	90.8	90.7
DNN AVG	91.3	92.0
DNN LSTM	88.1	89.2
Bilinear AVG + data	91.8	92.5
human	∼95.0	—

Table 5: Accuracies (%) of baselines and final model configurations on DEV2 and TEST. "+ data" uses enlarged training set of size 300,000, and then doubles this training set by including tuples with conjugated forms; see text for details. Human performance on DEV2 was estimated from a sample of size 100.

modeling terms. We suggest two reasons for this. The first is that most terms are short, with an average term length of 2.3 words in our training tuples. An LSTM may not be needed to capture long-distance properties. The second reason may be due to hyperparameter tuning. Recall that we used a greedy search for optimal hyperparameter values; we found that models with LSTMs take more time per epoch, more epochs to converge, and exhibit more hyperparameter sensitivity compared to models based on averaging. This may have contributed to inferior hyperparameter values for the LSTM models.

We also trained the Bilinear AVG model on a larger training set (row labeled "Bilinear AVG + data"). We note that the ConceptNet tuples typically contain unconjugated forms; we sought to use both conjugated and unconjugated words. We began with a larger training set of 300,000 tuples from ConceptNet, then augmented them to include conjugated word forms as in the following example. For the tuple ⟨soak in a hotspring, CAUSES, get pruny skin⟩ obtained from the OMCS sentence "The effect of [soaking in a hotspring] is [getting pruny skin]", we generated an additional tuple ⟨soaking in a hotspring, CAUSES, getting pruny skin⟩. We thus created twice as many training tuples. The results with this larger training set improved from 91.7 to 92.5 on the test set. We release this final model to the research community.

We note that the strongest baseline, MaxSim, is a nonparametric model that requires iterating over all training tuples to provide a score to new tuples. This is a serious bottleneck for use in NLP applications that may need to issue large numbers of

	Bilinear AVG		Bilinear LSTM		DNN AVG		DNN LSTM	
	CE	hinge	CE	hinge	CE	hinge	CE	hinge
random	**90**	84	**91**	83	**91**	87	**88**	57
mix	**90**	83	90	87	90	78	82	63
max	86	75	65	66	61	52	56	52

Table 3: Accuracies (%) on DEV2 of models trained with two loss functions (cross entropy (CE) and hinge) and three sampling strategies (random, mix, and max). The best accuracy for each model is shown in bold. Cross entropy with random sampling is best across models and is also fastest (see Table 4).

queries. Our models are parametric models that can compress a large training set into a fixed number of parameters. This makes them extremely fast for answering queries, particularly the AVG models, enabling use in downstream NLP applications.

7 Generating and Scoring Novel Tuples

We now measure our model's ability to score novel tuples generated automatically from ConceptNet and Wikipedia. We first describe simple procedures to generate candidate tuples from these two datasets. We then score the tuples using our MaxSim baseline and the trained Bilinear AVG model.[5] We evaluate the highly-scoring tuples using a small-scale manual evaluation.

The DNN AVG and Bilinear AVG models reached the highest TEST accuracies in our evaluation, though in preliminary experiments we found that the Bilinear AVG model appeared to perform better when scoring the novel tuples described below. We suspect this is because the DNN function class has more flexibility than the bilinear one. When scoring novel tuples, many of which may be highly noisy, it appears that the constrained structure of the Bilinear AVG model makes it more robust to the noise.

7.1 Generating Tuples From ConceptNet

In order to get new tuples, we automatically modify existing ConceptNet tuples. We take an existing tuple and randomly change one of the three fields (t_1, t_2, or R), ensuring that the result is not a tuple existing in ConceptNet. We then score these tuples using MaxSim and the Bilinear AVG model and analyze the results.

7.2 Generating Tuples from Wikipedia

We also propose a simple method to extract candidate tuples from raw text. We first run the Stanford part-of-speech (POS) tagger (Toutanova et al., 2003) on the terms in our ConceptNet training tuples. We enumerate the 50 most frequent term pair tag sequences for each relation. We do limited manual filtering of the frequent tag sequences, namely removing the sequences "DT NN NN" and "DT JJ NN" for the ISA relation. We do this in order to reduce noise in the extracted tuples. To focus on finding nontrivial tuples, for each relation we retain the top 15 POS tag sequences in which t_1 or t_2 has at least two words.

We then run the tagger on sentences from English Wikipedia. We extract word sequence pairs corresponding to the relation POS tag sequence pairs, requiring that there be a gap of at least one word between the two terms. We then remove word sequence pairs in which one term is solely one of the following words: be, the, always, there, has, due, however. We also remove tuples containing words that are not in the vocabulary of our ConceptNet-trained embeddings. We require that one term does not include the other term. We create tuples consisting of the two terms and all possible relations that occur with the POS sequences of those two terms. Finally, we remove tuples that exactly match our ConceptNet training tuples.

We use our trained Bilinear AVG model to score these tuples. We extract term pairs that occur within the same sentence because we hope that these will have higher precision than if we were to pair together arbitrary pairs. Some example tuples for $t_1 =$ *bus* are shown in Table 6.

t_1, R, t_2	score
bus, ISA, *public transportation*	0.95
bus, ISA, *public transit*	0.90
bus, ISA, *mass transit*	0.79
bus, ATLOCATION, *downtown area*	0.98
bus, ATLOCATION, *subway station*	0.98
bus, ATLOCATION, *city center*	0.94
bus, CAPABLEOF, *low cost*	0.72
bus, CAPABLEOF, *local service*	0.65
bus, CAPABLEOF, *train service*	0.63

Table 6: Top Wikipedia tuples for 3 relations with $t_1 =$ *bus*, scored by Bilinear AVG model.

[5] For the results in this section, we used the Bilinear AVG model that achieved 91.7 on TEST rather than the one augmented with additional data.

7.3 Manual Analysis of Novel Tuples

To evaluate our models on newly generated tuples, we rank them using different models and manually score the high-ranking tuples for quality. We first randomly sampled 3000 tuples from each set of novel tuples. We do so due to the time requirements of the MaxSim baseline, which requires iterating through the entire training set for each candidate tuple. We score these sampled tuples using MaxSim and the Bilinear AVG model and rank them by their scores. The top 100 tuples under each ranking were given to an annotator who is a native English speaker. The annotator assigned a quality score to each tuple, using the same 0-4 annotation scheme as Speer et al. (2010): 0 ("Doesn't make sense"), 1 ("Not true"), 2 ("Opinion/Don't know"), 3 ("Sometimes true"), and 4 ("Generally true"). We report the average quality score across each set of 100 tuples.

The results are shown in Table 7. To calibrate the scores, we also gave two samples of ConceptNet (CN) tuples to the annotator: a sample of 100 high-confidence tuples (first row) and a sample of 100 medium-confidence tuples (second row). We find the high-confidence tuples to be of high quality, recording an average of 3.68, though the medium-confidence tuples drop to 3.14.

The next two rows show the quality scores of the MaxSim baseline and the Bilinear AVG model. The latter outperforms the baseline and matches the quality of the medium-confidence ConceptNet tuples. Since our novel tuples are not contained in ConceptNet, this result suggests that our model can be used to add medium-confidence tuples to ConceptNet.

The novel Wikipedia tuples (top 100 tuples ranked by Bilinear AVG model) had a lower quality score (2.78), but this is to be expected due to the difference in domain. Since Wikipedia contains a wide variety of text, we found the novel tuples to be noisier than those from ConceptNet. Still, we are encouraged that on average the tuples are judged to be close to "sometimes true."

7.4 Text Analysis with Commonsense Tuples

We note that our method of tuple extraction and scoring could be used as an aid in applications that require sentence understanding. Two example sentences are shown in Table 8, along with the top tuples extracted and scored using our method. The tuples capture general knowledge about phrases contained in the sentence, rather than necessarily indicating what the sentence means. This procedure could provide relevant commonsense knowledge for a downstream application that seeks to understand the sentence. We leave further investigation of this idea to future work.

tuples	quality
high-confidence CN tuples	3.68
medium-confidence CN tuples	3.14
novel CN tuples, ranked by MaxSim	2.74
novel CN tuples, ranked by Bilinear AVG	3.20
novel Wiki tuples, ranked by Bilinear AVG	2.78

Table 7: Average quality scores from manual evaluation of novel tuples. Each row corresponds to a different set of tuples. See text for details.

After nine years of primary school, students can go to the high school or to an educational institution.	
t_1, R, t_2	score
school, HASPROPERTY, *educational*	0.89
school, ISA, *educational institution*	0.80
school, ISA, *institution*	0.78
school, HASPROPERTY, *high*	0.77
high school, ISA, *institution*	0.71
On March 14, 1964, Ruby was convicted of murder with malice, for which he received a death sentence.	
t_1, R, t_2	score
murder, CAUSES, *death**	1.00
murder, CAUSES, *death sentence*	0.86
murder, HASSUBEVENT, *death*	0.84
murder, CAPABLEOF, *death*	0.51

Table 8: Top ranked tuples extracted from two example sentences and scored by Bilinear AVG model. $*$ = contained in ConceptNet.

8 Conclusion

We proposed methods to augment curated commonsense resources using techniques from knowledge base completion. By scoring novel tuples, we showed how we can increase the applicability of the knowledge contained in ConceptNet. In future work, we will explore how to use our model to improve downstream NLP tasks, and consider applying our methods to other knowledge bases. We have released all of our resources—code, data, and trained models—to the research community.[6]

Acknowledgments

We thank the anonymous reviewers, John Wieting, and Luke Zettlemoyer.

[6]Available at `http://ttic.uchicago.edu/~kgimpel/commonsense.html`.

References

Basant Agarwal, Namita Mittal, Pooja Bansal, and Sonal Garg. 2015. Sentiment analysis using common-sense and context information. *Comp. Int. and Neurosc.*

Gabor Angeli and Christopher Manning. 2013. Philosophers are mortal: Inferring the truth of unseen facts. In *Proc. of CoNLL*.

Gabor Angeli and D. Christopher Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Proc. of EMNLP*.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING*.

Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2014. Tailoring continuous word representations for dependency parsing. In *Proc. of ACL*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proc. of the ACM SIGMOD International Conference on Management of Data*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in NIPS*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In *Proc. of EMNLP*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014b. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2).

Erik Cambria, Daniel Olsher, and Kenneth Kwok. 2012. Sentic activation: A two-level affective common sense reasoning framework. In *Proc. of AAAI*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *Proc. of AAAI*.

Luigi Di Caro, Alice Ruggeri, Loredana Cupi, and Guido Boella. 2015. Common-sense knowledge for natural language understanding: Experiments in unsupervised and supervised settings. In *Proc. of AI*IA*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proc. of EMNLP*.

Alberto García-Durán, Antoine Bordes, and Nicolas Usunier. 2014. Effective blending of two and three-way interactions for modeling multi-relational data. In *Machine Learning and Knowledge Discovery in Databases*.

Matt Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *Proc. of EMNLP*.

Jonathan Gordon and Lenhart K Schubert. 2012. Using textual patterns to learn expected event frequencies. In *Proc. of Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*.

Jonathan Gordon and Benjamin Van Durme. 2013. Reporting bias and knowledge acquisition. In *Proc. of Workshop on Automated Knowledge Base Construction*.

Jonathan Gordon, Benjamin Van Durme, and Lenhart K Schubert. 2010. Learning from the web: Extracting general world knowledge from noisy text. In *Collaboratively-Built Knowledge Sources and AI*.

Jonathan Gordon. 2014. *Inferential Commonsense Knowledge from Text*. Ph.D. thesis, University of Rochester.

Kelvin Gu, John Miller, and Percy Liang. 2015. Traversing knowledge graphs in vector space. In *Proc. of EMNLP*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. SimLex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proc. of CIKM*.

Rodolphe Jenatton, Nicolas L. Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in NIPS*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proc. of EMNLP*.

Douglas B Lenat and Ramanathan V Guha. 1989. *Building large knowledge-based systems: representation and inference in the Cyc project*. Addison-Wesley Longman Publishing Co., Inc.

Henry Lieberman, Alexander Faaborg, Waseem Daher, and José Espinosa. 2005. How to wreck a nice beach you sing calm incense. In *Proc. of IUI*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proc. of EMNLP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*.

George A Miller. 1995. WordNet: a lexical database for English. *Communications of the ACM*, 38(11).

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proc of ACL*.

Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *Proc. of NAACL*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *Proc. of ACL*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proc. of ICML*.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: Scalable machine learning for linked data. In *Proc. of WWW*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and M. Benjamin Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proc. of NAACL-HLT*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in NIPS*.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in ConceptNet 5. In *Proc. of LREC*.

Robert Speer, Catherine Havasi, and Henry Lieberman. 2008. AnalogySpace: Reducing the dimensionality of common sense knowledge. In *Proc. of AAAI*.

Robert Speer, Catherine Havasi, and Harshit Surana. 2010. Using verbosity: Common sense data from games with a purpose. In *Proc. of FLAIRS*.

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

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proc. of EMNLP*.

Luis von Ahn, Mihir Kedia, and Manuel Blum. 2006. Verbosity: a game for collecting common-sense facts. In *Proc. of CHI*.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge base completion via search-based question answering. In *Proc. of WWW*.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *Transactions of the ACL*.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. In *Proc. of ICLR*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proc. of Workshop on Automated Knowledge Base Construction*.

Simpler Context-Dependent Logical Forms via Model Projections

Reginald Long
Stanford University
reggylong@cs.stanford.edu

Panupong Pasupat
Stanford University
ppasupat@cs.stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

We consider the task of learning a context-dependent mapping from utterances to denotations. With only denotations at training time, we must search over a combinatorially large space of logical forms, which is even larger with context-dependent utterances. To cope with this challenge, we perform successive projections of the full model onto simpler models that operate over equivalence classes of logical forms. Though less expressive, we find that these simpler models are much faster and can be surprisingly effective. Moreover, they can be used to bootstrap the full model. Finally, we collected three new context-dependent semantic parsing datasets, and develop a new left-to-right parser.

1 Introduction

Suppose we are only told that a piece of text (a command) in some context (state of the world) has some denotation (the effect of the command)—see Figure 1 for an example. How can we build a system to learn from examples like these with no initial knowledge about what any of the words mean?

We start with the classic paradigm of training semantic parsers that map utterances to logical forms, which are executed to produce the denotation (Zelle and Mooney, 1996; Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Zettlemoyer and Collins, 2009; Kwiatkowski et al., 2010). More recent work learns directly from denotations (Clarke et al., 2010; Liang, 2013; Berant et al., 2013; Artzi and Zettlemoyer, 2013), but in this setting, a constant struggle is to contain the exponential explosion of possible logical forms. With no initial lexicon and longer context-dependent texts, our situation is exacerbated.

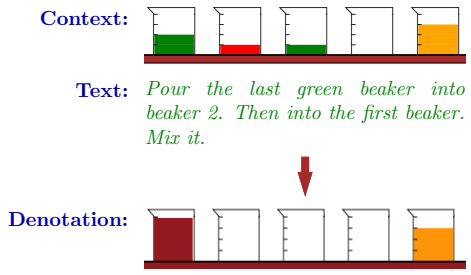


Figure 1: Our task is to learn to map a piece of text in some context to a denotation. An example from the ALCHEMY dataset is shown. In this paper, we ask: what intermediate logical form is suitable for modeling this mapping?

In this paper, we propose *projecting* a full semantic parsing model onto simpler models over equivalence classes of logical form derivations. As illustrated in Figure 2, we consider the following sequence of models:

- **Model A**: our full model that derives logical forms (e.g., in Figure 1, the last utterance maps to `mix(args[1][1])`) compositionally from the text so that spans of the utterance (e.g., "*it*") align to parts of the logical form (e.g., `args[1][1]`, which retrieves an argument from a previous logical form). This is based on standard semantic parsing (e.g., Zettlemoyer and Collins (2005)).
- **Model B**: collapse all derivations with the same logical form; we map utterances to full logical forms, but without an alignment between the utterance and logical forms. This "floating" approach was used in Pasupat and Liang (2015) and Wang et al. (2015).
- **Model C**: further collapse all logical forms whose top-level arguments have the same denotation. In other words, we map utterances

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1456–1465,
Berlin, Germany, August 7-12, 2016.

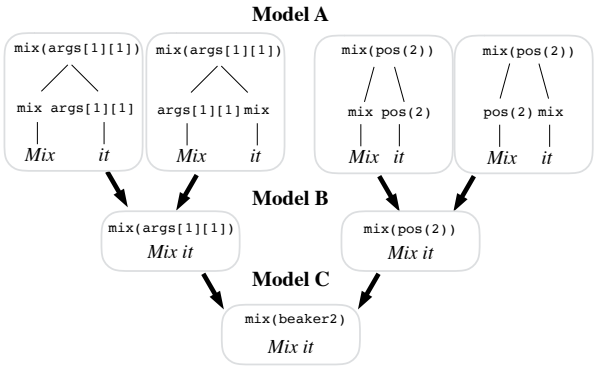


Figure 2: Derivations generated for the last utterance in Figure 1. All derivations above execute to `mix(beaker2)`. Model A generates anchored logical forms (derivations) where words are aligned to predicates, which leads to multiple derivations with the same logical form. Model B discards these alignments, and Model C collapses the arguments of the logical forms to denotations.

to flat logical forms (e.g., `mix(beaker2)`), where the arguments of the top-level predicate are objects in the world. This model is in the spirit of Yao et al. (2014) and Bordes et al. (2014), who directly predicted concrete paths in a knowledge graph for question answering.

Model A excels at credit assignment: the latent derivation explains how parts of the logical form are triggered by parts of the utterance. The price is an unmanageably large search space, given that we do not have a seed lexicon. At the other end, Model C only considers a small set of logical forms, but the mapping from text to the correct logical form is more complex and harder to model.

We collected three new context-dependent semantic parsing datasets using Amazon Mechanical Turk: ALCHEMY (Figure 1), SCENE (Figure 3), and TANGRAMS (Figure 4). Along the way, we develop a new parser which processes utterances left-to-right but can construct logical forms without an explicit alignment.

Our empirical findings are as follows: First, Model C is surprisingly effective, mostly surpassing the other two given bounded computational resources (a fixed beam size). Second, on a synthetic dataset, with infinite beam, Model A outperforms the other two models. Third, we can bootstrap up to Model A from the projected models with finite beam.

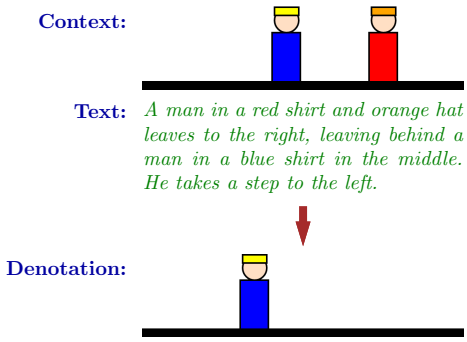


Figure 3: SCENE dataset: Each person has a shirt of some color and a hat of some color. They enter, leave, move around on a stage, and trade hats.

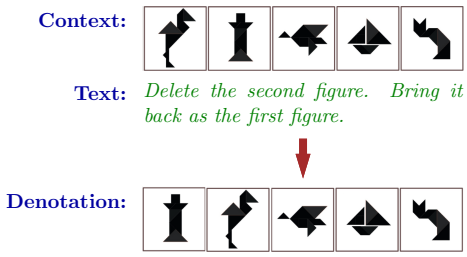


Figure 4: TANGRAMS dataset: One can add figures, remove figures, and swap the position of figures. All the figures slide to the left.

2 Task

In this section, we formalize the task and describe the new datasets we created for the task.

2.1 Setup

First, we will define the context-dependent semantic parsing task. Define w_0 as the initial world state, which consists of a set of entities (beakers in ALCHEMY) and properties (location, color(s), and amount filled). The text $\mathbf{x}$ is a sequence of utterances $x_1, \ldots, x_L$. For each utterance x_i (e.g., "*mix*"), we have a latent logical form z_i (e.g., `mix(args[1][2])`). Define the context $c_i = (w_0, z_{1:i-1})$ to include the initial world state w_0 and the history of past logical forms $z_{1:i-1}$. Each logical form z_i is executed on the context c_i to produce the next state: $w_i = \text{Exec}(c_i, z_i)$ for each $i = 1, \ldots, L$. Overloading notation, we write $w_L = \text{Exec}(w_0, \mathbf{z})$, where $\mathbf{z} = (z_1, \ldots, z_L)$.

The learning problem is: given a set of training examples $\{(w_0, \mathbf{x}, w_L)\}$, learn a mapping from the text $\mathbf{x}$ to logical forms $\mathbf{z} = (z_1, \ldots, z_L)$ that produces the correct final state ($w_L = \text{Exec}(w_0, \mathbf{z})$).

Dataset	# examples	# train	# test	words/example	utterances
SCENE	4402	3363	1039	56.2	"*then one more*", "*he moves back*"
ALCHEMY	4560	3661	899	39.9	"*mix*", "*throw the rest out*"
TANGRAMS	4989	4189	800	27.2	"*undo*", "*replace it*", "*take it away*"

Table 1: We collected three datasets. The number of examples, train/test split, number of tokens per example, along with interesting phenomena are shown for each dataset.

2.2 Datasets

We created three new context-dependent datasets, ALCHEMY, SCENE, and TANGRAMS (see Table 1 for a summary), which aim to capture a diverse set of context-dependent linguistic phenomena such as ellipsis (e.g., "*mix*" in ALCHEMY), anaphora on entities (e.g., "*he*" in SCENE), and anaphora on actions (e.g., "*repeat step 3*", "*bring it back*" in TANGRAMS).

For each dataset, we have a set of properties and actions. In ALCHEMY, properties are `color`, and `amount`; actions are `pour`, `drain`, and `mix`. In SCENE, properties are `hat-color` and `shirt-color`; actions are `enter`, `leave`, `move`, and `trade-hats`. In TANGRAMS, there is one property (`shape`), and actions are `add`, `remove`, and `swap`. In addition, we include the position property (`pos`) in each dataset. Each example has $L = 5$ utterances, each denoting some transformation of the world state.

Our datasets are unique in that they are grounded to a world state and have rich linguistic context-dependence. In the context-dependent ATIS dataset (Dahl et al., 1994) used by Zettlemoyer and Collins (2009), logical forms of utterances depend on previous logical forms, though there is no world state and the linguistic phenomena is limited to nominal references. In the map navigation dataset (Chen and Mooney, 2011), used by Artzi and Zettlemoyer (2013), utterances only reference the current world state. Vlachos and Clark (2014) released a corpus of annotated dialogues, which has interesting linguistic context-dependence, but there is no world state.

Data collection. Our strategy was to automatically generate sequences of world states and ask Amazon Mechanical Turk (AMT) workers to describe the successive transformations. Specifically, we started with a random world state w_0. For each $i = 1, \ldots, L$, we sample a valid action and argument (e.g., `pour(beaker1, beaker2)`). To encourage context-dependent descriptions, we upweight recently used actions and arguments (e.g., the next action is more like to be `drain(beaker2)` rather than `drain(beaker5)`). Next, we presented an AMT worker with states $w_0, \ldots, w_L$ and asked the worker to write a description in between each pair of successive states.

In initial experiments, we found it rather non-trivial to obtain interesting linguistic context-dependence in these micro-domains: often a context-independent utterance such as "*beaker 2*" is just clearer and not much longer than a possibly ambiguous "*it*". We modified the domains to encourage more context. For example, in SCENE, we removed any visual indication of absolute position and allowed people to only move next to other people. This way, workers would say "*to the left of the man in the red hat*" rather than "*to position 2*".

3 Model

We now describe Model A, our full context-dependent semantic parsing model. First, let $\mathcal{Z}$ denote the set of candidate logical forms (e.g., `pour(color(green),color(red))`). Each logical form consists of a top-level action with arguments, which are either primitive values (`green`, `3`, etc.), or composed via selection and superlative operations. See Table 2 for a full description. One notable feature of the logical forms is the context dependency: for example, given some context $(w_0, z_{1:4})$, the predicate `actions[2]` refers to the action of z_2 and `args[2][1]` refers to first argument of z_2.[1]

We use the term *anchored logical forms* (a.k.a. derivations) to refer to logical forms augmented with alignments between sub-logical forms of z_i and spans of the utterance x_i. In the example above, `color(green)` might align with "*green beaker*" from Figure 1; see Figure 2 for another example.

[1]These special predicates play the role of references in Zettlemoyer and Collins (2009). They perform context-independent parsing and resolve references, whereas we resolve them jointly while parsing.

Property[p] Value[v]	$\Rightarrow$ Set[$p(v)$]	all entities whose property p is v
Set[s] Property[p]	$\Rightarrow$ Value[argmin/argmax(s, p)]	element in s with smallest/largest p
Set[s] Int[i]	$\Rightarrow$ Value[$s[i]$]	i-th element of s
Action[a] Value[v_1] Value[v_2]	$\Rightarrow$ Root[$a(v_1, v_2)$]	top-level action applied to arguments v_1, v_2

Table 2: Grammar that defines the space of candidate logical forms. Values include numbers, colors, as well as special tokens `args`$[i][j]$ (for all $i \in \{1, \ldots, L\}$ and $j \in \{1, 2\}$) that refer to the j-th argument used in the i-th logical form. Actions include the fixed domain-specific set plus special tokens `actions`$[i]$ (for all $i \in \{1, \ldots, L\}$), which refers to the i-th action in the context.

	Derivation condition	Example
(F1)	z_i contains predicate r	(z_i contains predicate `pour`, "*pour*")
(F2)	property p of $z_i.b_j$ is y	(`color` of arg 1 is `green`, "*green*")
(F3)	action $z_i.a$ is a and property p of $z_i.y_j$ is y	(action is `pour` and `pos` of arg 2 is `2`, "*pour, 2*")
(F4)	properties p of $z_i.v_1$ is y and p' of $z_i.v_2$ is y'	(`color` of arg 1 is `green` and `pos` of arg 2 is `2`, "*first green, 2*")
(F5)	arg $z_i.v_j$ is one of z_{i-1}'s args	(arg reused, "*it*")
(F6)	action $z_i.a = z_{i-1}.a$	(action reused, "*pour*")
(F7)	properties p of $z_i.y_j$ is y and p' of $z_{i-1}.y_k$ is y'	(`pos` of arg 1 is `2` and `pos` of prev. arg 2 is `2`, "*then*")
(F8)	$t_1 < s_2$	spans don't overlap

Table 3: Features $\phi(x_i, c_i, z_i)$ for Model A: The left hand side describes conditions under which the system fires indicator features, and right hand side shows sample features for each condition. For each derivation condition (F1)–(F7), we conjoin the condition with the span of the utterance that the referenced actions and arguments align to. For condition (F8), we just fire the indicator by itself.

Log-linear model. We place a conditional distribution over anchored logical forms $z_i \in \mathcal{Z}$ given an utterance x_i and context $c_i = (w_0, z_{1:i-1})$, which consists of the initial world state w_0 and the history of past logical forms $z_{1:i-1}$. We use a standard log-linear model:

$$p_\theta(z_i \mid x_i, c_i) \propto \exp(\phi(x_i, c_i, z_i) \cdot \theta), \quad (1)$$

where ϕ is the feature mapping and θ is the parameter vector (to be learned). Chaining these distributions together, we get a distribution over a sequence of logical forms $\mathbf{z} = (z_1, \ldots, z_L)$ given the whole text $\mathbf{x}$:

$$p_\theta(\mathbf{z} \mid \mathbf{x}, w_0) = \prod_{i=1}^{L} p_\theta(z_i \mid x_i, (w_0, z_{1:i-1})). \quad (2)$$

Features. Our feature mapping ϕ consists of two types of indicators:

1. For each derivation, we fire features based on the structure of the logical form/spans.

2. For each span s (e.g., "*green beaker*") aligned to a sub-logical form z (e.g., `color(green)`), we fire features on unigrams, bigrams, and trigrams inside s conjoined with various conditions of z.

The exact features given in Table 3, references the first two utterances of Figure 1 and the associated logical forms below:

$x_1 =$ "*Pour the last green beaker into beaker 2.*"

$z_1 =$ `pour(argmin(color(green),pos),pos(2))`

$x_2 =$ "*Then into the first beaker.*"

$z_2 =$ `actions[1](args[1][2],pos(3))`.

We describe the notation we use for Table 3, restricting our discussion to actions that have two or fewer arguments. Our featurization scheme, however, generalizes to an arbitrary number of arguments. Given a logical form z_i, let $z_i.a$ be its action and $(z_i.b_1, z_i.b_2)$ be its arguments (e.g., `color(green)`). The first and second arguments are anchored over spans $[s_1, t_1]$ and $[s_2, t_2]$, respectively. Each argument $z_i.b_j$ has a corresponding value $z_i.v_j$ (e.g., `beaker1`), obtained by executing $z_i.b_j$ on the context c_i. Finally, let $j, k \in \{1, 2\}$ be indices of the arguments. For example, we would label the constituent parts of z_1 (defined above) as follows:

- $z_1.a =$ `pour`
- $z_1.b_1 =$ `argmin(color(green),pos)`
- $z_1.v_1 =$ `beaker3`
- $z_1.b_2 =$ `pos(2)`
- $z_1.v_2 =$ `beaker2`

4 Left-to-right parsing

We describe a new parser suitable for learning from denotations in the context-dependent setting. Like a shift-reduce parser, we proceed left to right, but each *shift* operation advances an entire utterance rather than one word. We then sit on the utterance for a while, performing a sequence of *build* operations, which either combine two logical forms on the stack (like the reduce operation) or generate fresh logical forms, similar to what is done in the floating parser of Pasupat and Liang (2015).

Our parser has two desirable properties: First, proceeding left-to-right allows us to build and score logical forms z_i that depend on the world state w_{i-1}, which is a function of the previous logical forms. Note that w_{i-1} is a random variable in our setting, whereas it is fixed in Zettlemoyer and Collins (2009). Second, the *build* operation allows us the flexibility to handle ellipsis (e.g., "*Mix.*") and anaphora on full logical forms (e.g., "*Do it again.*"), where there's not a clear alignment between the words and the predicates generated.

The parser transitions through a sequence of hypotheses. Each hypothesis is $h = (i, b, \sigma)$, where i is the index of the current utterance, where b is the number of predicates constructed on utterance x_i, and σ is a stack (list) of logical forms. The stack includes both the previous logical forms $z_{1:i-1}$ and fragments of logical forms built on the current utterance. When processing a particular hypothesis, the parser can choose to perform either the shift or build operation:

Shift: The parser moves to the next utterance by incrementing the utterance index i and resetting b, which transitions a hypothesis from (i, b, σ) to $(i + 1, 0, \sigma)$.

Build: The parser creates a new logical form by combining zero or more logical forms on the stack. There are four types of build operations:

1. Create a predicate out of thin air (e.g., `args[1][1]` in Figure 5). This is useful when the utterance does not explicitly reference the arguments or action. For example, in Figure 5, we are able to generate the logical form `args[1][1]` in the presence of ellipsis.

2. Create a predicate anchored to some span of the utterance (e.g., `actions[1]` anchored to "*Repeat*"). This allows us to do credit assignment and capture which part of the utterance explains which part of the logical form.

3. Pop z from the stack σ and push z' onto σ, where z' is created by applying a rule in Table 2 to z.

4. Pop z, z' from the stack σ and push z'' onto σ, where z'' is created by applying a rule in Table 2 to z, z' (e.g., `actions[1](args[1][1])` by the top-level root rule).

The build step stops once a maximum number of predicates B have been constructed or when the top-level rule is applied.

We have so far described the search space over logical forms. In practice, we keep a beam of the K hypotheses with the highest score under the current log-linear model.

5 Model Projections

Model A is ambitious, as it tries to learn from scratch how each word aligns to part of the logical form. For example, when Model A parses "*Mix it*", one derivation will correctly align "*Mix*" to `mix`, but others will align "*Mix*" to `args[1][1]`, "*Mix*" to `pos(2)`, and so on (Figure 2).

As we do not assume a seed lexicon that could map "*Mix*" to `mix`, the set of anchored logical forms is exponentially large. For example, parsing just the first sentence of Figure 1 would generate 1,216,140 intermediate anchored logical forms.

How can we reduce the search space? The key is that the space of logical forms is *much smaller* than the space of anchored logical forms. Even though both grow exponentially, dealing directly with logical forms allows us to generate `pour` without the combinatorial choice over alignments. We thus define Model B over the space of these logical forms. Figure 2 shows that the two anchored logical forms, which are treated differently in Model A are collapsed in Model B. This dramatically reduces the search space; parsing the first sentence of Figure 1 generates 7,047 intermediate logical forms.

We can go further and notice that many compositional logical forms reduce to the same flat logical form if we evaluate all the arguments. For example, in Figure 2, `mix(args[1][1])` and `mix(pos(2))` are equivalent to `mix(beaker2)`. We define Model C to be the

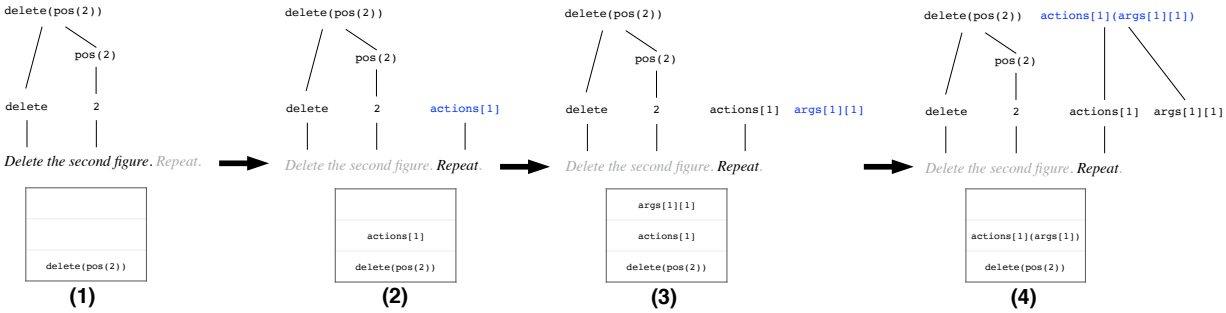


Figure 5: Suppose we have already constructed `delete(pos(2))` for "*Delete the second figure.*" Continuing, we shift the utterance "*Repeat*". Then, we build `action[1]` aligned to the word "*Repeat.*" followed by `args[1][1]`, which is unaligned. Finally, we combine the two logical forms.

space of these flat logical forms which consist of a top-level action plus primitive arguments. Using Model C, parsing the first sentence of Figure 1 generates only 349 intermediate logical forms.

Note that in the context-dependent setting, the number of flat logical forms (Model C) still increases exponentially with the number of utterances, but it is an overwhelming improvement over Model A. Furthermore, unlike other forms of relaxation, we are still generating logical forms that can express any denotation as before. The gains from Model B to Model C hinge on the fact that in our world, the number of denotations is much smaller than the number of logical forms.

Projecting the features. While we have defined the space over logical forms for Models B and C, we still need to define a distribution over these spaces to to complete the picture. To do this, we propose projecting the features of the log-linear model (1). Define $\Pi_{\text{A}\to\text{B}}$ to be a map from a anchored logical form z^{A} (e.g., `mix(pos(2))` aligned to "*mix*") to an unanchored one z^{B} (e.g., `mix(pos(2))`), and define $\Pi_{\text{B}\to\text{C}}$ to be a map from z^{B} to the flat logical form z^{C} (e.g., `mix(beaker2)`).

We construct a log-linear model for Model B by constructing features $\phi(z^{\text{B}})$ (omitting the dependence on x_i, c_i for convenience) based on the Model A features $\phi(z^{\text{A}})$. Specifically, $\phi(z^{\text{B}})$ is the component-wise maximum of $\phi(z^{\text{A}})$ over all z^{A} that project down to z^{B}; $\phi(z^{\text{C}})$ is defined similarly:

$$\phi(z^{\text{B}}) \stackrel{\text{def}}{=} \max\{\phi(z^{\text{A}}) : \Pi_{\text{A}\to\text{B}}(z^{\text{A}}) = z^{\text{B}}\}, \quad (3)$$

$$\phi(z^{\text{C}}) \stackrel{\text{def}}{=} \max\{\phi(z^{\text{B}}) : \Pi_{\text{B}\to\text{C}}(z^{\text{B}}) = z^{\text{C}}\}. \quad (4)$$

Concretely, Model B's features include indicator features over LF conditions in Table 3 conjoined with every n-gram of the entire utterance, as there is no alignment. This is similar to the model of Pasupat and Liang (2015). Note that most of the derivation conditions (F2)–(F7) already depend on properties of the denotations of the arguments, so in Model C, we can directly reason over the space of flat logical forms z^{C} (e.g., `mix(beaker2)`) rather than explicitly computing the max over more complex logical forms z^{B} (e.g., `mix(color(red))`).

Expressivity. In going from Model A to Model C, we gain in computational efficiency, but we lose in modeling expressivity. For example, for "*second green beaker*" in Figure 1, instead of predicting `color(green)[2]`, we would have to predict `beaker3`, which is not easily explained by the words "*second green beaker*" using the simple features in Table 3.

At the same time, we found that simple features can actually *simulate* some logical forms. For example, `color(green)` can be explained by the feature that looks at the `color` property of `beaker3`. Nailing `color(green)[2]`, however, is not easy. Surprisingly, Model C can use a conjunction of features to express superlatives (e.g., `argmax(color(red),pos)`) by using one feature that places more mass on selecting objects that are red and another feature that places more mass on objects that have a greater position value.

6 Experiments

Our experiments aim to explore the computation-expressivity tradeoff in going from Model A to Model B to Model C. We would expect that under the computational constraint of a finite beam size, Model A will be hurt the most, but with an

Dataset	Model	3-acc	3-ora	5-acc	5-ora
ALCHEMY	B	0.189	0.258	0.037	0.055
	C	**0.568**	**0.925**	**0.523**	**0.809**
SCENE	B	0.068	0.118	0.017	0.031
	C	**0.232**	**0.431**	**0.147**	**0.253**
TANGRAMS	B	**0.649**	**0.910**	**0.276**	0.513
	C	0.567	0.899	0.272	**0.698**

Table 4: Test set accuracy and oracle accuracy for examples containing $L = 3$ and $L = 5$ utterances. Model C surpasses Model B in both accuracy and oracle on ALCHEMY and SCENE, whereas Model B does better in TANGRAMS.

infinite beam, Model A should perform better.

We evaluate all models on *accuracy*, the fraction of examples that a model predicts correctly. A predicted logical form z is deemed to be correct for an example $(w_0, \mathbf{x}, w_L)$ if the predicted logical form $\mathbf{z}$ executes to the correct final world state w_L. We also measure the *oracle accuracy*, which is the fraction of examples where at least one $\mathbf{z}$ on the beam executes to w_L. All experiments train for 6 iterations using AdaGrad (Duchi et al., 2010) and L_1 regularization with a coefficient of 0.001.

6.1 Real data experiments

Setup. We use a beam size of 500 within each utterance, and prune to the top 5 between utterances. For the first two iterations, Models B and C train on only the first utterance of each example ($L = 1$). In the remaining iterations, the models train on two utterance examples. We then evaluate on examples with $L = 1, \ldots, 5$, which tests our models ability to extrapolate to longer texts.

Accuracy with finite beam. We compare models B and C on the three real datasets for both $L = 3$ and $L = 5$ utterances (Model A was too expensive to use). Table 4 shows that on 5 utterance examples, the flatter Model C achieves an average accuracy of 20% higher than the more compositional Model B. Similarly, the average oracle accuracy is 39% higher. This suggests that (i) the correct logical form often falls off the beam for Model B due to a larger search space, and (ii) the expressivity of Model C is sufficient in many cases.

On the other hand, Model B outperforms Model C on the TANGRAMS dataset. This happens for two reasons. The TANGRAMS dataset has the smallest search space, since all of the utterances refer to objects using position only. Additionally, many utterances reference logical forms that Model C is unable to express, such as "*repeat the first step*", or "*add it back*".

Figure 6 shows how the models perform as the number of utterances per example varies. When the search space is small (fewer number of utterances), Model B outperforms or is competitive with Model C. However, as the search space increases (tighter computational constraints), Model C does increasingly better.

Overall, both models perform worse as L increases, since to predict the final world state w_L correctly, a model essentially needs to predict an entire sequence of logical forms $z_1, \ldots, z_L$, and errors cascade. Furthermore, for larger L, the utterances tend to have richer context-dependence.

6.2 Artificial data experiments

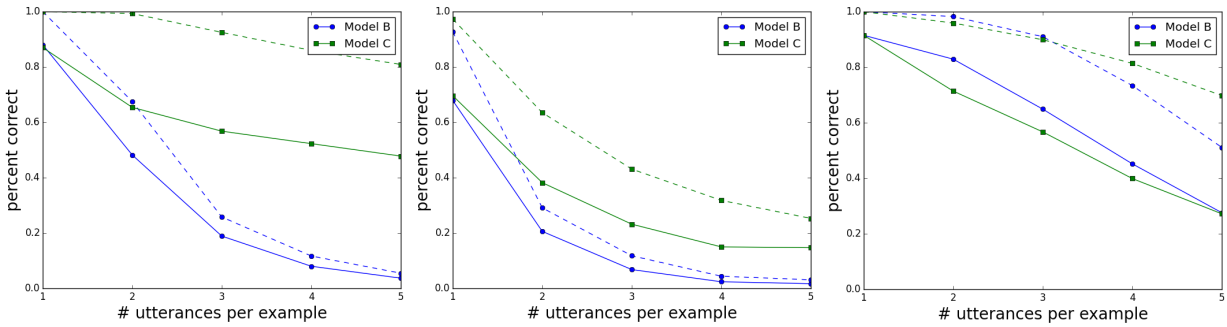
Setup. Due to the large search space, running model A on real data is impractical. In order feasibly evaluate Model A, we constructed an artificial dataset. The worlds are created using the procedure described in Section 2.2. We use a simple template to generate utterances (e.g., "*drain 1 from the 2 green beaker*").

To reduce the search space for Model A, we only allow actions (e.g., `drain`) to align to verbs and property values (e.g., `green`) to align to adjectives. Using these linguistic constraints provides a slightly optimistic assessment of Model A's performance.

We train on a dataset of 500 training examples and evaluate on 500 test examples. We repeat this procedure for varying beam sizes, from 40 to 260. The model only uses features (F1) through (F3).

Accuracy under infinite beam. Since Model A is more expressive, we would expect it to be more powerful when we have no computational constraints. Figure 7 shows that this is indeed the case: When the beam size is greater than 250, all models attain an oracle of 1, and Model A outperforms Model B, which performs similarly to Model C. This is because the alignments provide a powerful signal for constructing the logical forms. Without alignments, Models B and C learn noisier features, and accuracy suffers accordingly.

Bootstrapping. Model A performs the best with unconstrained computation, and Model C performs the best with constrained computation. Is there some way to bridge the two? Even though



(a) ALCHEMY (b) SCENE (c) TANGRAMS

Figure 6: Test results on our three datasets as we vary the number of utterances. The solid lines are the accuracy, and the dashed line are the oracles: With finite beam, Model C significantly outperforms Model B on ALCHEMY and SCENE, but is slightly worse on TANGRAMS.

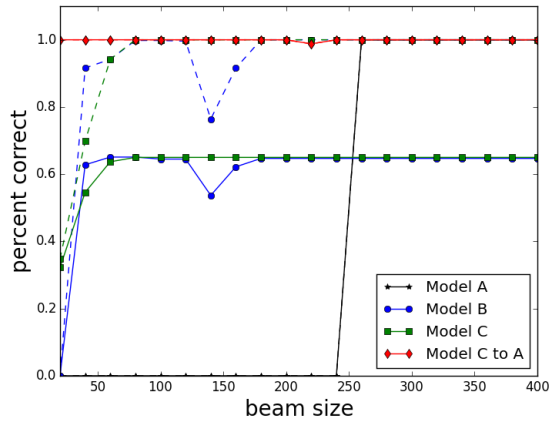


Figure 7: Test results on our artificial dataset with varying beam sizes. The solid lines are the accuracies, and the dashed line are the oracle accuracies. Model A is unable to learn anything with beam size < 240. However, for beam sizes larger than 240, Model A attains 100% accuracy. Model C does better than Models A and B when the beam size is small < 40, but otherwise performs comparably to Model B. Bootstrapping Model A using Model C parameters outperforms all of the other models and attains 100% even with smaller beams.

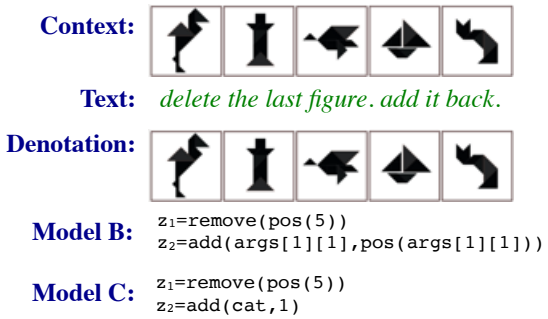


Figure 8: Predicted logical forms for this text: The logical form `add` takes a figure and position as input. Model B predicts the correct logical form. Model C does not understand that "*back*" refers to position 5, and adds the cat figure to position 1.

model	beam	action	argument	context	noise
B	0.47	0.03	0.17	0.23	0.04
C	0.15	0.03	0.25	0.5	0.07

Table 5: Percentage of errors for Model B and C: Model B suffers predominantly from computation constraints, while Model C suffers predominantly from a lack of expressivity.

Model C has limited expressivity, it can still learn to associate words like "*green*" with their corresponding predicate `green`. These should be useful for Model A too.

To operationalize this, we first train Model C and use the parameters to initialize model A. Then we train Model A. Figure 7 shows that although Model A and C predict different logical forms, the initialization allows Model C to A to perform well in constrained beam settings. This bootstrapping works here because Model C is a *projection* of Model A, and thus they share the same features.

6.3 Error Analysis

We randomly sampled 20 incorrect predictions on 3 utterance examples from each of the three real datasets for Model B and Model C. We categorized each prediction error into one of the following categories: (i) logical forms falling off the beam; (ii) choosing the wrong action (e.g., mapping "*drain*" to `pour`); (iii) choosing the wrong

argument due to misunderstanding the description (e.g., mapping "*third beaker*" to `pos(1)`); (iv) choosing the wrong action or argument due to misunderstanding of context (see Figure 8); (v) noise in the dataset. Table 5 shows the fraction of each error category.

7 Related Work and Discussion

Context-dependent semantic parsing. Utterances can depend on either linguistic context or world state context. Zettlemoyer and Collins (2009) developed a model that handles references to previous logical forms; Artzi and Zettlemoyer (2013) developed a model that handles references to the current world state. Our system considers both types of context, handling linguistic phenomena such as ellipsis and anaphora that reference both previous world states and logical forms.

Logical form generation. Traditional semantic parsers generate logical forms by aligning each part of the logical form to the utterance (Zelle and Mooney, 1996; Wong and Mooney, 2007; Zettlemoyer and Collins, 2007; Kwiatkowski et al., 2011). In general, such systems rely on a lexicon, which can be hand-engineered, extracted (Cai and Yates, 2013; Berant et al., 2013), or automatically learned from annotated logical forms (Kwiatkowski et al., 2010; Chen, 2012).

Recent work on learning from denotations has moved away from anchored logical forms. Pasupat and Liang (2014) and Wang et al. (2015) proposed generating logical forms without alignments, similar to our Model B. Yao et al. (2014) and Bordes et al. (2014) have explored predicting paths in a knowledge graph directly, which is similar to the flat logical forms of Model C.

Relaxation and bootstrapping. The idea of first training a simpler model in order to work up to a more complex one has been explored other contexts. In the unsupervised learning of generative models, bootstrapping can help escape local optima and provide helpful regularization (Och and Ney, 2003; Liang et al., 2009). When it is difficult to even find one logical form that reaches the denotation, one can use the relaxation technique of Steinhardt and Liang (2015).

Recall that projecting from Model A to C creates a more computationally tractable model at the cost of expressivity. However, this is because Model C used a linear model. One might imagine that a non-linear model would be able to recuperate some of the loss of expressivity. Indeed, Neelakantan et al. (2016) use recurrent neural networks attempt to perform logical operations. One could go one step further and bypass logical forms altogether, performing all the logical reasoning in a continuous space (Bowman et al., 2014; Weston et al., 2015; Guu et al., 2015; Reed and de Freitas, 2016). This certainly avoids the combinatorial explosion of logical forms in Model A, but could also present additional optimization challenges. It would be worth exploring this avenue to completely understand the computation-expressivity tradeoff.

Reproducibility

Our code, data, and experiments are available on CodaLab at `https://worksheets.codalab.org/worksheets/0xad3fc9f52f514e849b282a105b1e3f02/`.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. The third author is supported by a Microsoft Research Faculty Fellowship.

References

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

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

A. Bordes, S. Chopra, and J. Weston. 2014. Question answering with subgraph embeddings. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. R. Bowman, C. Potts, and C. D. Manning. 2014. Can recursive neural tensor networks learn logical reasoning? In *International Conference on Learning Representations (ICLR)*.

Q. Cai and A. Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Association for Computational Linguistics (ACL)*.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 859–865.

D. L. Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Association for Computational Linguistics (ACL)*.

J. Clarke, D. Goldwasser, M. Chang, and D. Roth. 2010. Driving semantic parsing from the world's response. In *Computational Natural Language Learning (CoNLL)*, pages 18–27.

D. A. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Workshop on Human Language Technology*, pages 43–48.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In *Empirical Methods in Natural Language Processing (EMNLP)*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1223–1233.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2011. Lexical generalization in CCG grammar induction for semantic parsing. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1512–1523.

P. Liang, M. I. Jordan, and D. Klein. 2009. Learning semantic correspondences with less supervision. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pages 91–99.

P. Liang. 2013. Lambda dependency-based compositional semantics. *arXiv*.

A. Neelakantan, Q. V. Le, and I. Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *International Conference on Learning Representations (ICLR)*.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29:19–51.

P. Pasupat and P. Liang. 2014. Zero-shot entity extraction from web pages. In *Association for Computational Linguistics (ACL)*.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

S. Reed and N. de Freitas. 2016. Neural programmer-interpreters. In *International Conference on Learning Representations (ICLR)*.

J. Steinhardt and P. Liang. 2015. Learning with relaxed supervision. In *Advances in Neural Information Processing Systems (NIPS)*.

A. Vlachos and S. Clark. 2014. A new corpus and imitation learning framework for context-dependent semantic parsing. *Transactions of the Association for Computational Linguistics (TACL)*, 2:547–559.

Y. Wang, J. Berant, and P. Liang. 2015. Building a semantic parser overnight. In *Association for Computational Linguistics (ACL)*.

J. Weston, A. Bordes, S. Chopra, and T. Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv*.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*, pages 960–967.

X. Yao, J. Berant, and B. Van-Durme. 2014. Freebase QA: Information extraction or semantic parsing. In *Workshop on Semantic parsing*.

M. Zelle and R. J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Association for the Advancement of Artificial Intelligence (AAAI)*, pages 1050–1055.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*, pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*, pages 678–687.

L. S. Zettlemoyer and M. Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

A Fast Unified Model for Parsing and Sentence Understanding

Samuel R. Bowman[1,2,3,*] sbowman@stanford.edu
Jon Gauthier[2,3,4,*] jgauthie@stanford.edu
Abhinav Rastogi[3,5] arastogi@stanford.edu
Raghav Gupta[2,3,6] rgupta93@stanford.edu
Christopher D. Manning[1,2,3,6] manning@stanford.edu
Christopher Potts[1,6] cgpotts@stanford.edu

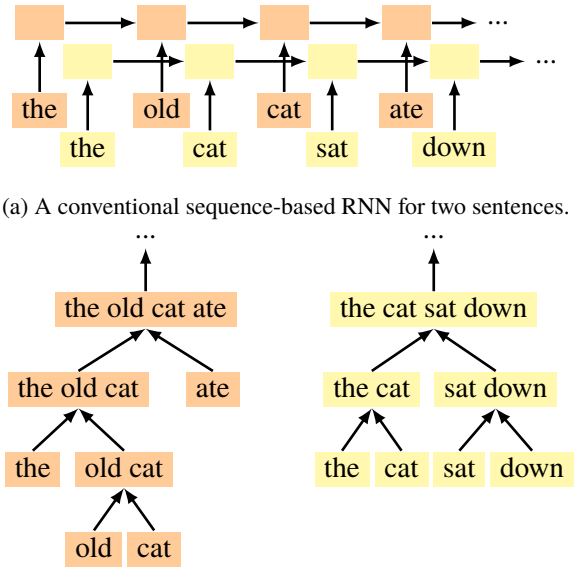
[1]Stanford Linguistics [2]Stanford NLP Group [3]Stanford AI Lab
[4]Stanford Symbolic Systems [5]Stanford Electrical Engineering [6]Stanford Computer Science

Abstract

Tree-structured neural networks exploit valuable syntactic parse information as they interpret the meanings of sentences. However, they suffer from two key technical problems that make them slow and unwieldy for large-scale NLP tasks: they usually operate on parsed sentences and they do not directly support batched computation. We address these issues by introducing the Stack-augmented Parser-Interpreter Neural Network (SPINN), which combines parsing and interpretation within a single tree-sequence hybrid model by integrating tree-structured sentence interpretation into the linear sequential structure of a shift-reduce parser. Our model supports batched computation for a speedup of up to 25× over other tree-structured models, and its integrated parser can operate on unparsed data with little loss in accuracy. We evaluate it on the Stanford NLI entailment task and show that it significantly outperforms other sentence-encoding models.

1 Introduction

A wide range of current models in NLP are built around a neural network component that produces vector representations of sentence meaning (e.g., Sutskever et al., 2014; Tai et al., 2015). This component, the sentence encoder, is generally formulated as a learned parametric function from a sequence of word vectors to a sentence vector, and this function can take a range of different forms. Common sentence encoders include sequence-based recurrent neural network models (RNNs, see Figure 1a) with Long Short-Term Memory (LSTM, Hochreiter and Schmidhuber, 1997), which accumulate information over the sentence sequentially; convolutional neural networks (Kalchbrenner et al., 2014; Zhang et al., 2015), which accumulate information using filters over short local sequences of words or characters; and tree-structured recursive neural networks (TreeRNNs, Goller and Küchler, 1996; Socher et al., 2011a, see Figure 1b), which propagate information up a binary parse tree.



(a) A conventional sequence-based RNN for two sentences.

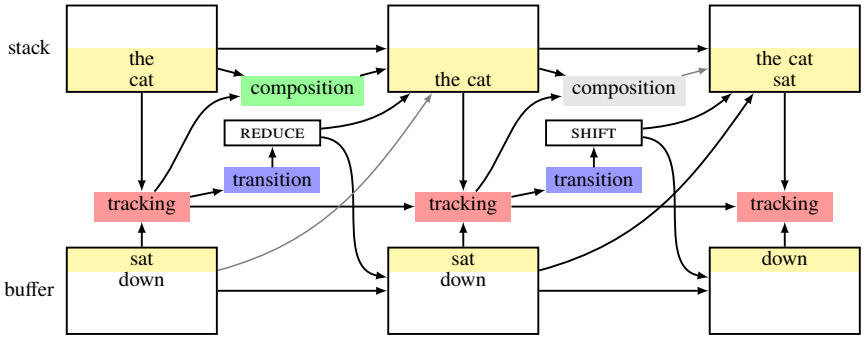
(b) A conventional TreeRNN for two sentences.

Figure 1: An illustration of two standard designs for sentence encoders. The TreeRNN, unlike the sequence-based RNN, requires a substantially different connection structure for each sentence, making batched computation impractical.

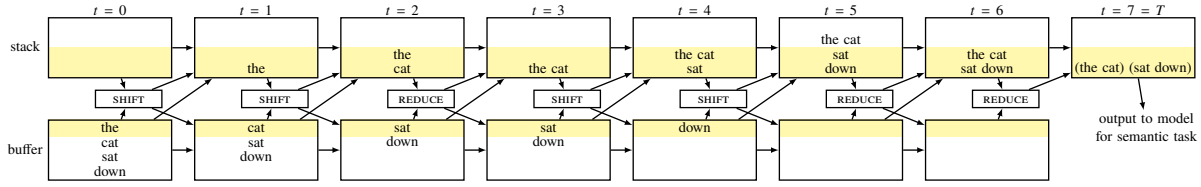
Of these, the TreeRNN appears to be the principled choice, since meaning in natural language sentences is known to be constructed recursively according to a tree structure (Dowty, 2007, i.a.). TreeRNNs have shown promise (Tai et al., 2015; Li et al., 2015; Bowman et al., 2015b), but have

*The first two authors contributed equally.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1466–1477,
Berlin, Germany, August 7-12, 2016.



(a) The SPINN model unrolled for two transitions during the processing of the sentence *the cat sat down*. 'Tracking', 'transition', and 'composition' are neural network layers. Gray arrows indicate connections which are blocked by a gating function.



(b) The fully unrolled SPINN for *the cat sat down*, with neural network layers omitted for clarity.

Figure 2: Two views of the Stack-augmented Parser-Interpreter Neural Network (SPINN).

largely been overlooked in favor of sequence-based RNNs because of their incompatibility with batched computation and their reliance on external parsers. Batched computation—performing synchronized computation across many examples at once—yields order-of-magnitude improvements in model run time, and is crucial in enabling neural networks to be trained efficiently on large datasets. Because TreeRNNs use a different model structure for each sentence, as in Figure 1, efficient batching is impossible in standard implementations. Partly to address efficiency problems, standard TreeRNN models commonly only operate on sentences that have already been processed by a syntactic parser, which slows and complicates the use of these models at test time for most applications.

This paper introduces a new model to address both these issues: the Stack-augmented Parser-Interpreter Neural Network, or SPINN, shown in Figure 2. SPINN executes the computations of a tree-structured model in a linearized sequence, and can incorporate a neural network parser that produces the required parse structure on the fly. This design improves upon the TreeRNN architecture in three ways: At test time, it can simultaneously parse and interpret unparsed sentences, removing the dependence on an external parser at nearly no additional computational cost. Secondly, it supports batched computation for both parsed and unparsed sentences, yielding dramatic speedups over standard TreeRNNs. Finally, it supports a novel tree-sequence hybrid architecture for handling local linear context in sentence interpretation. This model is a basically plausible model of human sentence processing and yields substantial accuracy gains over pure sequence- or tree-based models.

We evaluate SPINN on the Stanford Natural Language Inference entailment task (SNLI, Bowman et al., 2015a), and find that it significantly outperforms other sentence-encoding-based models, even with a relatively simple and underpowered implementation of the built-in parser. We also find that SPINN yields speed increases of up to 25× over a standard TreeRNN implementation.

2 Related work

There is a fairly long history of work on building neural network-based parsers that use the core operations and data structures from transition-based parsing, of which shift-reduce parsing is a variant (Henderson, 2004; Emami and Jelinek, 2005; Titov and Henderson, 2010; Chen and Manning, 2014; Buys and Blunsom, 2015; Dyer et al., 2015; Kiperwasser and Goldberg, 2016). In addition, there has been recent work proposing models designed primarily for generative language modeling tasks that use this architecture as well (Zhang et al., 2016; Dyer et al., 2016). To our knowledge, SPINN is the first model to use this architecture for the purpose of sentence interpretation, rather than parsing

or generation.

Socher et al. (2011a,b) present versions of the TreeRNN model which are capable of operating over unparsed inputs. However, these methods require an expensive search process at test time. Our model presents a much faster alternative approach.

3 Our model: SPINN

3.1 Background: Shift-reduce parsing

SPINN is inspired by shift-reduce parsing (Aho and Ullman, 1972), which builds a tree structure over a sequence (e.g., a natural language sentence) by a single left-to-right scan over its tokens. The formalism is widely used in natural language parsing (e.g., Shieber, 1983; Nivre, 2003).

A shift-reduce parser accepts a sequence of input tokens $\mathbf{x} = (x_0, \ldots, x_{N-1})$ and consumes transitions $\mathbf{a} = (a_0, \ldots, a_{T-1})$, where each $a_t \in \{\text{SHIFT}, \text{REDUCE}\}$ specifies one step of the parsing process. In general a parser may also generate these transitions on the fly as it reads the tokens. It proceeds left-to-right through a transition sequence, combining the input tokens $\mathbf{x}$ incrementally into a tree structure. For any binary-branching tree structure over N words, this requires $T = 2N - 1$ transitions through a total of $T + 1$ states.

The parser uses two auxiliary data structures: a stack S of partially completed subtrees and a buffer B of tokens yet to be parsed. The parser is initialized with the stack empty and the buffer containing the tokens $\mathbf{x}$ of the sentence in order. Let $\langle S, B\rangle = \langle \emptyset, \mathbf{x}\rangle$ denote this starting state. It next proceeds through the transition sequence, where each transition a_t selects one of the two following operations. Below, the | symbol denotes the *cons* (concatenation) operator. We arbitrarily choose to always *cons* on the left in the notation below.

SHIFT: $\langle S, x \mid B\rangle \rightarrow \langle x \mid S, B\rangle$. This operation pops an element from the buffer and pushes it on to the top of the stack.

REDUCE: $\langle x \mid y \mid S, B\rangle \rightarrow \langle (x, y) \mid S, B\rangle$. This operation pops the top two elements from the stack, merges them, and pushes the result back on to the stack.

3.2 Composition and representation

SPINN is based on a shift-reduce parser, but it is designed to produce a vector representation of a sentence as its output, rather than a tree as in standard shift-reduce parsing. It modifies the shift-reduce formalism by using fixed length vectors to represent each entry in the stack and the buffer. Correspondingly, its REDUCE operation combines two vector representations from the stack into another vector using a neural network function.

The composition function When a REDUCE operation is performed, the vector representations of two tree nodes are popped off of the stack and fed into a *composition function*, which is a neural network function that produces a representation for a new tree node that is the parent of the two popped nodes. This new node is pushed on to the stack.

The TreeLSTM composition function (Tai et al., 2015) generalizes the LSTM neural network layer to tree- rather than sequence-based inputs, and it shares with the LSTM the idea of representing intermediate states as a pair of an active state representation $\vec{h}$ and a memory representation $\vec{c}$. Our version is formulated as:

$$\text{(1)} \quad \begin{bmatrix} \vec{i} \\ \vec{f}_l \\ \vec{f}_r \\ \vec{o} \\ \vec{g} \end{bmatrix} = \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(W_{\text{comp}} \begin{bmatrix} \vec{h}_s^1 \\ \vec{h}_s^2 \\ \vec{e} \end{bmatrix} + \vec{b}_{\text{comp}} \right)$$

$$\text{(2)} \quad \vec{c} = \vec{f}_l \odot \vec{c}_s^2 + \vec{f}_r \odot \vec{c}_s^1 + \vec{i} \odot \vec{g}$$

$$\text{(3)} \quad \vec{h} = \vec{o} \odot \vec{c}$$

where σ is the sigmoid activation function, $\odot$ is the elementwise product, the pairs $\langle \vec{h}_s^1, \vec{c}_s^1 \rangle$ and $\langle \vec{h}_s^2, \vec{c}_s^2 \rangle$ are the two input tree nodes popped off the stack, and $\vec{e}$ is an optional vector-valued input argument which is either empty or comes from an external source like the tracking LSTM (see Section 3.3). The result of this function, the pair $\langle \vec{h}, \vec{c} \rangle$, is placed back on the stack. Each vector-valued variable listed is of dimension D except $\vec{e}$, of the independent dimension D_{tracking}.

The stack and buffer The stack and the buffer are arrays of N elements each (for sentences of up to N words), with the two D-dimensional vectors $\vec{h}$ and $\vec{c}$ in each element.

Word representations We use word representations based on the 300D vectors provided with GloVe (Pennington et al., 2014). We do not update these representations during training. Instead, we use a learned linear transformation to map each input word vector $\vec{x}_{\text{GloVe}}$ into a vector pair $\langle \vec{h}, \vec{c} \rangle$ that is stored in the buffer:

$$\text{(4)} \quad \begin{bmatrix} \vec{h} \\ \vec{c} \end{bmatrix} = W_{\text{wd}} \vec{x}_{\text{GloVe}} + \vec{b}_{\text{wd}}$$

3.3 The tracking LSTM

In addition to the stack, buffer, and composition function, our full model includes an additional component: the tracking LSTM. This is a simple sequence-based LSTM RNN that operates in tandem with the model, taking inputs from the buffer and stack at each step. It is meant to maintain a low-resolution summary of the portion of the sentence that has been processed so far, which is used for two purposes: it supplies feature representations to the transition classifier, which allows the model to stand alone as a parser, and it additionally supplies a secondary input $\vec{e}$ to the composition function—see (1)—allowing context information to enter the construction of sentence meaning and forming what is effectively a tree-sequence hybrid model.

The tracking LSTM's inputs (yellow in Figure 2) are the top element of the buffer $\vec{h}_b^1$ (which would be moved in a SHIFT operation) and the top two elements of the stack $\vec{h}_s^1$ and $\vec{h}_s^2$ (which would be composed in a REDUCE operation).

Why a tree-sequence hybrid? Lexical ambiguity is ubiquitous in natural language. Most words have multiple senses or meanings, and it is generally necessary to use the context in which a word occurs to determine which of its senses or meanings is meant in a given sentence. Even though TreeRNNs are more effective at composing meanings in principle, this ambiguity can give simpler sequence-based sentence-encoding models an advantage: when a sequence-based model first processes a word, it has direct access to a state vector that summarizes the left context of that word, which acts as a cue for disambiguation. In contrast, when a standard tree-structured model first processes a word, it only has access to the constituent that the word is merging with, which is often just a single additional word. Feeding a context representation from the tracking LSTM into the composition function is a simple and efficient way to mitigate this disadvantage of tree-structured models. Using left linear context to disambiguate is also a plausible model of human interpretation.

It would be straightforward to augment SPINN to support the use of some amount of right-side context as well, but this would add complexity to the model that we think is largely unnecessary: humans are very effective at understanding the beginnings of sentences before having seen or heard the ends, suggesting that it is possible to get by without the unavailable right-side context.

3.4 Parsing: Predicting transitions

For SPINN to operate on unparsed inputs, it needs to produce its own transition sequence **a** rather than relying on an external parser to supply it as part of the input. To do this, the model predicts a_t at each step using a simple two-way softmax classifier whose input is the state of the tracking LSTM:

$$\vec{p}_{\text{a}} = \text{softmax}(W_{\text{trans}}\vec{h}_{\text{tracking}} + \vec{b}_{\text{trans}}) \quad (5)$$

The above model is nearly the simplest viable implementation of a transition decision function. In contrast, the decision functions in state-of-the-art transition-based parsers tend to use significantly richer feature sets as inputs, including features containing information about several upcoming words on the buffer. The value $\vec{h}_{\text{tracking}}$ is a function of only the very top of the buffer and the top two stack elements at each timestep.

At test time, the model uses whichever transition (i.e., SHIFT or REDUCE) is assigned a higher (unnormalized) probability. The prediction function is trained to mimic the decisions of an external parser. These decisions are used as inputs to the model during training. For SNLI, we use the binary Stanford PCFG Parser parses that are included with the corpus. We did not find scheduled sampling (Bengio et al., 2015)—having the model use its own transition decisions sometimes at training time—to help.

3.5 Implementation issues

Representing the stack efficiently A naïve implementation of SPINN needs to handle a size $O(N)$ stack at each timestep, any element of which may be involved in later computations. A naïve backpropagation implementation would then require storing each of the $O(N)$ stacks for a backward pass, leading to a per-example space requirement of $O(NTD)$ floats. This requirement is prohibitively large for significant batch sizes or sentence lengths N. Such a naïve implementation would also require copying a largely unchanged stack at each timestep, since each SHIFT or REDUCE operation writes only one new representation to the top of the stack.

We propose a space-efficient stack representation inspired by the zipper technique (Huet, 1997) that we call *thin stack*. For each input sentence, we

Algorithm 1 The thin stack algorithm

```
function STEP(bufferTop, a, t, S, Q)
    if a = SHIFT then
        S[t] := bufferTop
    else if a = REDUCE then
        right := S[Q.pop()]
        left := S[Q.pop()]
        S[t] := COMPOSE(left, right)
    Q.push(t)
```

represent the stack with a single $T \times D$ matrix S. Each row $S[t]$ (for $0 < t \leq T$) represents the top of the actual stack at timestep t. At each timestep we can SHIFT a new element onto the stack, or REDUCE the top two elements of the stack into a single element. To shift an element from the buffer to the top of the stack at timestep t, we simply write it into the location $S[t]$. In order to perform the REDUCE operation, we need to retrieve the top two elements of the actual stack. We maintain a queue Q of pointers into S which contains the row indices of S which are still present in the actual stack. The top two elements of the stack can be found by using the final two pointers in the queue Q. These retrieved elements are used to perform the REDUCE operation, which modifies Q to mark that some rows of S have now been replaced in the actual stack. Algorithm 1 describes the full mechanics of a stack feedforward in this compressed representation. It operates on the single $T \times D$ matrix S and a backpointer queue Q. Table 1 shows an example run.

This stack representation requires substantially less space. It stores each element involved in the feedforward computation exactly once, meaning that this representation can still support efficient backpropagation. Furthermore, all of the updates to S and Q can be performed batched and in-place on a GPU, yielding substantial speed gains over both a more naïve SPINN implementation and a standard TreeRNN implementation. We describe speed results in Section 3.7.

Preparing the data At training time, SPINN requires both a transition sequence **a** and a token sequence **x** as its inputs for each sentence. The token sequence is simply the words in the sentence in order. **a** can be obtained from any constituency parse for the sentence by first converting that parse into an unlabeled binary parse, then linearizing it (with the usual in-order traversal), then taking each word token as a SHIFT transition and each ')' as a REDUCE transition, as here:

Unlabeled binary parse: ((the cat) (sat down))
x: *the*, *cat*, *sat*, *down*
a: SHIFT, SHIFT, REDUCE, SHIFT, SHIFT, REDUCE, REDUCE

t	$S[t]$	Q_t	a_t
0		—	SHIFT
1	*Spot*	1	SHIFT
2	*sat*	1 2	SHIFT
3	*down*	1 2 3	REDUCE
4	(*sat down*)	1 4	REDUCE
5	(*Spot* (*sat down*))	5	

Table 1: The thin-stack algorithm operating on the input sequence **x** = (*Spot*, *sat*, *down*) and the transition sequence shown in the rightmost column. $S[t]$ shows the top of the stack at each step t. The last two elements of Q (underlined) specify which rows t would be involved in a REDUCE operation at the next step.

Handling variable sentence lengths For any sentence model to be trained with batched computation, it is necessary to pad or crop sentences to a fixed length. We fix this length at $N = 25$ words, longer than about 98% of sentences in SNLI. Transition sequences **a** are cropped at the left or padded at the left with SHIFTS. Token sequences **x** are then cropped or padded with empty tokens at the left to match the number of SHIFTS added or removed from **a**, and can then be padded with empty tokens at the right to meet the desired length N.

3.6 TreeRNN-equivalence

Without the addition of the tracking LSTM, SPINN (in particular the SPINN-PI-NT variant, for *parsed input, no tracking*) is precisely equivalent to a conventional tree-structured neural network model in the function that it computes, and therefore it also has the same learning dynamics. In both, the representation of each sentence consists of the representations of the words combined recursively using a TreeRNN composition function (in our case, the TreeLSTM function). SPINN, however, is dramatically faster, and supports both integrated parsing and a novel approach to context through the tracking LSTM.

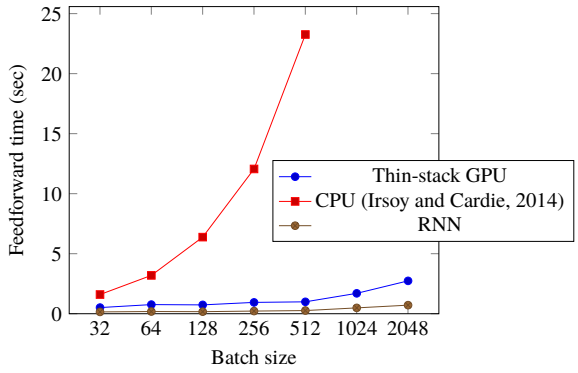


Figure 3: Feedforward speed comparison.

3.7 Inference speed

In this section, we compare the test-time speed of our SPINN-PI-NT with an equivalent TreeRNN implemented in the conventional fashion and with a standard RNN sequence model. While the full models evaluated below are implemented and trained using Theano (Theano Development Team, 2016), which is reasonably efficient but not perfect for our model, we wish to compare well-optimized implementations of all three models. To do this, we reimplement the feedforward[1] of SPINN-PI-NT and an LSTM RNN baseline in C++/CUDA, and compare that implementation with a CPU-based C++/Eigen TreeRNN implementation from Irsoy and Cardie (2014), which we modified to perform exactly the same computations as SPINN-PI-NT.[2] TreeRNNs like this can only operate on a single example at a time and are thus poorly suited for GPU computation.

Each model is restricted to run on sentences of 30 tokens or fewer. We fix the model dimension D and the word embedding dimension at 300. We run the CPU performance test on a 2.20 GHz 16-core Intel Xeon E5-2660 processor with hyperthreading enabled. We test our thin-stack implementation and the RNN model on an NVIDIA Titan X GPU.

Figure 3 compares the sentence encoding speed of the three models on random input data. We observe a substantial difference in runtime between the CPU and thin-stack implementations that increases with batch size. With a large but practical batch size of 512, the largest on which we tested the TreeRNN, our model is about 25× faster than the standard CPU implementation, and about 4× slower than the RNN baseline.

Though this experiment only covers SPINN-PI-NT, the results should be similar for the full SPINN model: most of the computation involved in running SPINN is involved in populating the buffer, applying the composition function, and manipulating the buffer and the stack, with the low-dimensional tracking and parsing components adding only a small additional load.

4 NLI Experiments

We evaluate SPINN on the task of natural language inference (NLI, a.k.a. recognizing textual entailment, or RTE; Dagan et al., 2006). NLI is a sentence pair classification task, in which a model reads two sentences (a premise and a hypothesis), and outputs a judgment of *entailment*, *contradiction*, or *neutral*, reflecting the relationship between the meanings of the two sentences. Below is an example sentence pair and judgment from the SNLI corpus which we use in our experiments:

Premise: Girl in a red coat, blue head wrap and jeans is making a snow angel.

Hypothesis: A girl outside plays in the snow.

Label: entailment

SNLI is a corpus of 570k human-labeled pairs of scene descriptions like this one. We use the standard train–test split and ignore unlabeled examples, which leaves about 549k examples for training, 9,842 for development, and 9,824 for testing. SNLI labels are roughly balanced, with the most frequent label, *entailment*, making up 34.2% of the test set.

Although NLI is framed as a simple three-way classification task, it is nonetheless an effective way of evaluating the ability of a model to extract broadly informative representations of sentence meaning. In order for a model to perform reliably well on NLI, it must be able to represent and reason with the core phenomena of natural language semantics, including quantification, coreference, scope, and several types of ambiguity.

4.1 Applying SPINN to SNLI

Creating a sentence-pair classifier To classify an SNLI sentence pair, we run two copies of SPINN with shared parameters: one on the premise sentence and another on the hypothesis sentence. We then use their outputs (the $\vec{h}$ states at the top of each

[1] We chose to reimplement and evaluate only the feedforward/inference pass, as inference speed is the relevant performance metric for most practical applications.

[2] The original code for Irsoy & Cardie's model is available at `https://github.com/oir/deep-recursive`. Our optimized C++/CUDA models and the Theano source code for the full SPINN are available at `https://github.com/stanfordnlp/spinn`.

Param.	Range	Strategy	RNN	SP.-PI-NT	SP.-PI	SP.
Initial LR	2×10^{-4}–2×10^{-2}	LOG	5×10^{-3}	3×10^{-4}	7×10^{-3}	2×10^{-3}
L2 regularization λ	8×10^{-7}–3×10^{-5}	LOG	4×10^{-6}	3×10^{-6}	2×10^{-5}	3×10^{-5}
Transition cost α	0.5–4.0	LIN	—	—	—	3.9
Embedding transformation dropout	80–95%	LIN	—	83%	92%	86%
Classifier MLP dropout	80–95%	LIN	94%	94%	93%	94%
Tracking LSTM size D_{tracking}	24–128	LOG	—	—	61	79
Classifier MLP layers	1–3	LIN	2	2	2	1

Table 2: Hyperparameter ranges and values. *Range* shows the hyperparameter ranges explored during random search. *Strategy* indicates whether sampling from the range was uniform, or log-uniform. Dropout parameters are expressed as keep rates rather than drop rates.

stack at time $t = T$) to construct a feature vector $\vec{x}_{\text{classifier}}$ for the pair. This feature vector consists of the concatenation of these two sentence vectors, their difference, and their elementwise product (following Mou et al., 2016):

$$\vec{x}_{\text{classifier}} = \begin{bmatrix} \vec{h}_{\text{premise}} \\ \vec{h}_{\text{hypothesis}} \\ \vec{h}_{\text{premise}} - \vec{h}_{\text{hypothesis}} \\ \vec{h}_{\text{premise}} \odot \vec{h}_{\text{hypothesis}} \end{bmatrix} \quad (6)$$

This feature vector is then passed to a series of 1024D ReLU neural network layers (i.e., an MLP; the number of layers is tuned as a hyperparameter), then passed into a linear transformation, and then finally passed to a softmax layer, which yields a distribution over the three labels.

The objective function Our objective combines a cross-entropy objective $\mathcal{L}_{\text{s}}$ for the SNLI classification task, cross-entropy objectives $\mathcal{L}_{\text{p}}^t$ and $\mathcal{L}_{\text{h}}^t$ for the parsing decision for each of the two sentences at each step t, and an L2 regularization term on the trained parameters. The terms are weighted using the tuned hyperparameters α and λ:

$$\mathcal{L}_{\text{m}} = \mathcal{L}_{\text{s}} + \alpha \sum_{t=0}^{T-1} (\mathcal{L}_{\text{p}}^t + \mathcal{L}_{\text{h}}^t) + \lambda \|\theta\|_2^2 \quad (7)$$

Initialization, optimization, and tuning We initialize the model parameters using the nonparametric strategy of He et al. (2015), with the exception of the softmax classifier parameters, which we initialize using random uniform samples from $[-0.005, 0.005]$.

We use minibatch SGD with the RMSProp optimizer (Tieleman and Hinton, 2012) and a tuned starting learning rate that decays by a factor of 0.75 every 10k steps. We apply both dropout (Srivastava et al., 2014) and batch normalization (Ioffe and Szegedy, 2015) to the output of the word embedding projection layer and to the feature vectors that serve as the inputs and outputs to the MLP that precedes the final entailment classifier.

We train each model for 250k steps in each run, using a batch size of 32. We track each model's performance on the development set during training and save parameters when this performance reaches a new peak. We use early stopping, evaluating on the test set using the parameters that perform best on the development set.

We use random search to tune the hyperparameters of each model, setting the ranges for search for each hyperparameter heuristically (and validating the reasonableness of the ranges on the development set), and then launching eight copies of each experiment each with newly sampled hyperparameters from those ranges. Table 2 shows the hyperparameters used in the best run of each model.

4.2 Models evaluated

We evaluate four models. The four all use the sentence-pair classifier architecture described in Section 4.1, and differ only in the function computing the sentence encodings. First, a single-layer LSTM RNN (similar to that of Bowman et al., 2015a) serves as a baseline encoder. Next, the minimal SPINN-PI-NT model (equivalent to a TreeLSTM) introduces the SPINN model design. SPINN-PI adds the tracking LSTM to that design. Finally, the full SPINN adds the integrated parser.

We compare our models against several baselines, including the strongest published non-neural network-based result from Bowman et al. (2015a) and previous neural network models built around several types of sentence encoders.

Model	Params.	Trans. acc. (%)	Train acc. (%)	Test acc. (%)
Previous non-NN results				
Lexicalized classifier (Bowman et al., 2015a)	—	—	99.7	78.2
Previous sentence encoder-based NN results				
100D LSTM encoders (Bowman et al., 2015a)	221k	—	84.8	77.6
1024D pretrained GRU encoders (Vendrov et al., 2016)	15m	—	98.8	81.4
300D Tree-based CNN encoders (Mou et al., 2016)	3.5m	—	83.4	82.1
Our results				
300D LSTM RNN encoders	3.0m	—	83.9	80.6
300D SPINN-PI-NT (*parsed input, no tracking*) encoders	3.4m	—	84.4	80.9
300D SPINN-PI (*parsed input*) encoders	3.7m	—	89.2	**83.2**
300D SPINN (unparsed input) encoders	2.7m	92.4	87.2	82.6

Table 3: Results on SNLI 3-way inference classification. Params. is the approximate number of trained parameters (excluding word embeddings for all models). Trans. acc. is the model's accuracy in predicting parsing transitions at test time. Train and test are SNLI classification accuracy.

4.3 Results

Table 3 shows our results on SNLI. For the full SPINN, we also report a measure of agreement between this model's parses and the parses included with SNLI, calculated as classification accuracy over transitions averaged across timesteps.

We find that the bare SPINN-PI-NT model performs little better than the RNN baseline, but that SPINN-PI with the added tracking LSTM performs well. The success of SPINN-PI, which is the hybrid tree-sequence model, suggests that the tree- and sequence-based encoding methods are at least partially complementary, with the sequence model presumably providing useful local word disambiguation. The full SPINN model with its relatively weak internal parser performs slightly less well, but nonetheless robustly exceeds the performance of the RNN baseline.

Both SPINN-PI and the full SPINN significantly outperform all previous sentence-encoding models. Most notably, these models outperform the tree-based CNN of Mou et al. (2016), which also uses tree-structured composition for local feature extraction, but uses simpler pooling techniques to build sentence features in the interest of efficiency. Our results show that a model that uses tree-structured composition fully (SPINN) outperforms one which uses it only partially (tree-based CNN), which in turn outperforms one which does not use it at all (RNN).

The full SPINN performed moderately well at reproducing the Stanford Parser's parses of the SNLI data at a transition-by-transition level, with 92.4% accuracy at test time.[3] However, its transition prediction errors are fairly evenly distributed across sentences, and most sentences were assigned partially invalid transition sequences that either left a few words out of the final representation or incorporated a few padding tokens into the final representation.

4.4 Discussion

The use of tree structure improves the performance of sentence-encoding models for SNLI. We suspect that this improvement is largely due to the more efficient learning of accurate generalizations overall, and not to any particular few phenomena. However, some patterns are identifiable in the results.

While all four models under study have trouble with negation, the tree-structured SPINN models do quite substantially better on these pairs. This is likely due to the fact that parse trees make the scope of any instance of negation (the portion of the sentence's content that is negated) relatively easy to identify and separate from the rest of the sentence. For test set sentence pairs like the one below where negation (*not* or *n't*) does not appear in the premise but does appear in the hypothesis, the RNN shows 67% accuracy, while all three tree-structured models exceed 73%. Only the RNN got the below example wrong:

Premise: The rhythmic gymnast completes her floor exercise at the competition.

Hypothesis: The gymnast cannot finish her exercise.

Label: contradiction

Note that the presence of negation in the hypothesis is correlated with a label of *contradiction* in SNLI, but not as strongly as one might intuit—only 45% of these examples in the test set are labeled as

[3]Note that this is scoring the model against automatic parses, not a human-judged gold standard.

contradictions.

In addition, it seems that tree-structured models, and especially the tree-sequence hybrid models, are more effective than RNNs at extracting informative representations of long sentences. The RNN model falls off in test accuracy more quickly with increasing sentence length than SPINN-PI-NT, which in turn falls of substantially faster than the two hybrid models, repeating a pattern seen more dramatically on artificial data in Bowman et al. (2015b). On pairs with premises of 20 or more words, the RNN's 76.7% accuracy, while SPINN-PI reaches 80.2%. All three SPINN models labeled the following example correctly, while the RNN did not:

Premise: A man wearing glasses and a ragged costume is playing a Jaguar electric guitar and singing with the accompaniment of a drummer.

Hypothesis: A man with glasses and a disheveled outfit is playing a guitar and singing along with a drummer.

Label: entailment

We suspect that the hybrid nature of the full SPINN model is also responsible for its surprising ability to perform better than an RNN baseline even when its internal parser is relatively ineffective at producing correct full-sentence parses. It may act somewhat like the tree-based CNN, only with access to larger trees: using tree structure to build up local phrase meanings, and then using the tracking LSTM, at least in part, to combine those meanings.

Finally, as is likely inevitable for models evaluated on SNLI, all four models under study did several percent worse on test examples whose ground truth label is *neutral* than on examples of the other two classes. *Entailment–neutral* and *neutral–contradiction* confusions appear to be much harder to avoid than *entailment–contradiction* confusions, where relatively superficial cues might be more readily useful.

5 Conclusions and future work

We introduce a model architecture (SPINN-PI-NT) that is equivalent to a TreeLSTM, but an order of magnitude faster at test time. We expand that architecture into a tree-sequence hybrid model (SPINN-PI), and show that this yields significant gains on the SNLI entailment task. Finally, we show that it is possible to exploit the strengths of this model without the need for an external parser by integrating a fast parser into the model (as in the full SPINN), and that the lack of external parse information yields little loss in accuracy.

Because this paper aims to introduce a general purpose model for sentence encoding, we do not pursue the use of soft attention (Bahdanau et al., 2015; Rocktäschel et al., 2016), despite its demonstrated effectiveness on the SNLI task.[4] However, we expect that it should be possible to productively combine our model with soft attention to reach state-of-the-art performance.

Our tracking LSTM uses only simple, quick-to-compute features drawn from the head of the buffer and the head of the stack. It is plausible that giving the tracking LSTM access to more information from the buffer and stack at each step would allow it to better represent the context at each tree node, yielding both better parsing and better sentence encoding. One promising way to pursue this goal would be to encode the full contents of the stack and buffer at each time step following the method used by Dyer et al. (2015).

For a more ambitious goal, we expect that it should be possible to implement a variant of SPINN on top of a modified stack data structure with differentiable PUSH and POP operations (as in Grefenstette et al., 2015; Joulin and Mikolov, 2015). This would make it possible for the model to learn to parse using guidance from the semantic representation objective, which currently is blocked from influencing the key parsing parameters by our use of hard SHIFT/REDUCE decisions. This change would allow the model to learn to produce parses that are, in aggregate, better suited to supporting semantic interpretation than those supplied in the training data.

Acknowledgments

We acknowledge financial support from a Google Faculty Research Award, the Stanford Data Science Initiative, and the National Science Foundation under grant nos. BCS 1456077 and IIS 1514268. Some of the Tesla K40s used for this research were donated by the NVIDIA Corporation. We also thank Kelvin Guu, Noah Goodman, and many others in the Stanford NLP group for helpful comments.

[4] Attention-based models like Rocktäschel et al. (2016), Wang and Jiang (2016), and the unpublished Cheng et al. (2016) have shown accuracies as high as 86.3% on SNLI, but are more narrowly engineered to suit the task and do not yield sentence encodings.

References

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The theory of parsing, translation, and compiling*. Prentice-Hall, Inc.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*. San Diego, CA.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems (NIPS) 29*. Montréal, Québec, pages 1171–1179.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 632–642.

Samuel R. Bowman, Christopher D. Manning, and Christopher Potts. 2015b. Tree-structured composition in neural networks without tree-structured architectures. In *Proceedings of the 2015 NIPS Workshop on Cognitive Computation: Integrating Neural and Symbolic Approaches*. Montréal, Québec, pages 50–55.

Jan Buys and Phil Blunsom. 2015. Generative incremental dependency parsing with neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 863–869.

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

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long short-term memory-networks for machine reading. arXiv:1601.06733.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognising textual entailment challenge. In *Machine learning challenges. Evaluating predictive uncertainty, visual object classification, and recognising tectual entailment*, Springer, pages 177–190.

David Dowty. 2007. Compositionality as an empirical problem. In *Proceedings of the Brown University Conference on Direct Compositionality*. Oxford Univ. Press.

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

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 199–209.

Ahmad Emami and Frederick Jelinek. 2005. A neural syntactic language model. *Machine learning* 60(1–3):195–227.

Christoph Goller and Andreas Küchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. In *Proceedings of the IEEE International Conference on Neural Networks*. Washington, DC, pages 347–352.

Edward Grefenstette, Karl Moritz Hermann, Mustafa Suleyman, and Phil Blunsom. 2015. Learning to transduce with unbounded memory. In *Advances in Neural Information Processing Systems (NIPS) 29*. Montréal, Québec, pages 1828–1836.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the Interna-*

tional Conference on Computer Vision (ICCV). Santiago, Chile, pages 1026–1034.

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*. Barcelona, Spain, pages 95–102.

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

Gérard Huet. 1997. The zipper. *Journal of functional programming* 7(5):549–554.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the International Conference on Machine Learning (ICML)*. Lille, France.

Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems (NIPS) 27*, Curran Associates, Inc., pages 2096–2104.

Armand Joulin and Tomas Mikolov. 2015. Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in Neural Information Processing Systems (NIPS) 29*. Montréal, Québec.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 655–665.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Easy-first dependency parsing with hierarchical tree LSTMs. arXiv:1603.00375.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, pages 2304–2314.

Lili Mou, Men Rui, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the ACL*. Berlin, Germany.

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

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1532–1543.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.

Stuart M. Shieber. 1983. Sentence disambiguation by a shift-reduce parsing technique. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Cambridge, Massachusetts, USA, pages 113–118.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Ng, and Chris Manning. 2011a. Parsing natural scenes and natural language with recursive neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*. ACM, New York, NY, USA, ICML '11, pages 129–136.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011b. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Edinburgh, Scotland, UK., pages 151–161.

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

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neu-

ral networks. In *Advances in Neural Information Processing Systems (NIPS) 28*.

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

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. arXiv:1605.02688.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5 – RMSProp: Divide the gradient by a running average of its recent magnitude. In *Neural Networks for Machine Learning*, Coursera.

Ivan Titov and James Henderson. 2010. A latent variable model for generative dependency parsing. In Harry Bunt, Paola Merlo, and Joakim Nivre, editors, *Trends in Parsing Technology*, Springer, Netherlands, pages 35–55.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. 2016. Order-embeddings of images and language. In *Proceedings of the International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico.

Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 1442–1451.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems (NIPS) 29*. Montréal, Québec, pages 649–657.

Xingxing Zhang, Liang Lu, and Mirella Lapata. 2016. Top-down tree long short-term memory networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, pages 310–320.

Investigating Language Universal and Specific Properties in Word Embeddings

Peng Qian Xipeng Qiu* Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pqian11, xpqiu, xjhuang}@fudan.edu.cn

Abstract

Recently, many NLP tasks have benefited from distributed word representation. However, it remains unknown whether embedding models are really immune to the typological diversity of languages, despite the language-independent architecture. Here we investigate three representative models on a large set of language samples by mapping dense embedding to sparse linguistic property space. Experiment results reveal the language universal and specific properties encoded in various word representation. Additionally, strong evidence supports the utility of word form, especially for inflectional languages.

1 Introduction

Word representation is a core issue in natural language processing. Context-based word representation, which is inspired by Harris (1954), has achieved huge successes in many NLP applications. Despite its popularity, character-based approach also comes out as an equal competitor (Santos and Zadrozny, 2014; Kim et al., 2016; Ling et al., 2015b; Ling et al., 2015a; Faruqui et al., 2016; Ballesteros et al., 2015) . Moreover, questions arise when we consider what these models could capture from linguistic cues under the perspective of cross-language typological diversity, as is argued by Bender (2009).

Despite previous efforts in empirically interpreting word embedding and exploring the intrinsic/extrinsic factors in learning process (Andreas and Klein, 2014; Lai et al., 2015; Köhn, 2015; Melamud et al., 2016), it remains unknown whether embedding models are really immune to the structural variance of languages.

Current research has gaps for understanding model behaviours towards language typological diversity as well as the utility of context and form for different languages. Thus, we select three representative types of models and design a series of experiments to reveal the universals and specifics of various word representations on decoding linguistic properties. Our work contributes to shedding new insights into the following topics:

a) How do typological differences of language structure influence a word embedding model? Does a model behave similarly towards phylogenetically-related languages?

b) Is word form a more efficient predictor of a certain grammatical function than word context for specific languages?

c) How do the neurons of a model respond to linguistic features? Can we explain the utility of context and form by analyzing neuron activation pattern?

2 Experiment Design

To study the proposed questions above, we design four series of experiments to comprehensively compare context-based and character-based word representations on different languages, covering syntactic, morphological and semantic properties. The basic paradigm is to decode interpretable linguistic features from a target collection of word representations. We hypothesize that there exists a linear/nonlinear map between a word representation x and a high-level sparse feature vector y if the word vector implicitly encode sufficient information[1]. Figure1 visualizes how a

*Corresponding author.

[1]Our experiment results show that nonlinear mapping model significantly works better than linear map for all languages. Only nonlinear mapping accuracies are mentioned in the following sections due to the space limit.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1478–1488,
Berlin, Germany, August 7-12, 2016.

word embedding is mapped to different linguistic attribute vectors. For example, the Czech word *dětem* means children in English. Its grammatical gender is female. It is in the plural form and should be used in dative case. These are all important properties of a word. The word embedding of *dětem* is mapped to different sparse representation of these lexical properties respectively.

Listed in Table1 is the outline of the experiments.

ID	Attribute	Category
I	Part-of-Speech	Syntax
II	Dependency Relation	
III	Gender / Number / Case Animacy / Definite / Person Tense / Aspect / Mood / Voice PronType / VerbForm	Morphology
IV	Sentiment Score	Semantics

Table 1: Outline of Experiment Design.

For linear map, we train a matrix Θ that maps word embedding $\mathbf{x}$ to a sparse feature vector $\mathbf{y}$ with the least L_2 error. For nonlinear map, we train a neural network (MLP) with 4 hidden layers via back propagation. Their dimensions are 50, 80, 80, and 50 in order. For each linguistic feature of each language, a mapping model is trained on the randomly-selected 90% of the words with the target feature and tested over the remaining 10%. Details about the construction of the linguistic feature vectors will be mentioned in the specific section of a certain experiment.

For syntactic and morphological features, we construct the corresponding feature vectors of a word from the Universal Dependencies Treebank (Joakim Nivre and Zhu, 2015) and the Chinese Treebank (CTB 7.0) (Xue et al., 2010). For a certain word w with a certain linguistic attribute a (e.g. POS), w may be annotated with one or different labels (e.g. NOUN, VERB, etc) from the possible label set of a in the whole treebank. We calculate the normalized label frequency distribution $\vec{\mathbf{y}}_w^a$ from the manual annotation of the corpus as the representation of the linguistic attribute a for the word w in each language.

For word sentiment feature, we use the manually annotated data collected by Dodds et al. (2015). The data contains emotion scores for a list of words in several languages. In our experiment, the original score scale in Dodds et al. (2015) is transformed into the interval $[0, 1]$.

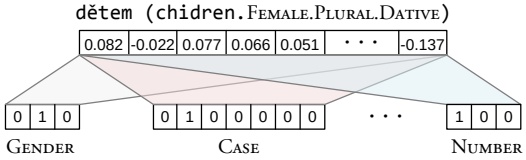


Figure 1: Visualizing experiment paradigm. The dense representation of a Czech word *dětem* is mapped to different sparse representation of the lexical properties respectively.

3 Embedding Model Description

Faced with three questions proposed before, we select the following models from various candidates, as they are popular, representative and based on either word context or purely word form.

Type I C&W Model (referred as CW in short), which aims to estimate the joint probability of a word sequence (Collobert et al., 2011). In this paper, C&W word vectors are all from the released version of the polyglot multilingual embeddings (Al-Rfou et al., 2013) trained on Wikipedia.

Type II Skip-gram[2] (referred as SG in short), which aims to predict the context words based on the target word. We use word2vec (Mikolov et al., 2013) to train SG on multilingual Wikipediea provided by (Al-Rfou et al., 2013).

Type III Character-based LSTM autoencoder (referred as AE in short), which takes the character sequence of a word as the input and reconstruct the input character sequence. It takes the advantage of pure word form instead of the context. The hidden layer vector of the model is used as a representation of the word. In this way, we are able to quantify the utility of pure word form by evaluating the representation generated from the character-based LSTM autoencoder on different decoding tasks. We trained one-hidden layer AE with the words covered in CW for each language independently.

To ensure a fair comparison, all the word vectors have the same dimension 64. CW and SG are trained with a common 5-word window size.

4 Results

4.1 Part-of-Speech

In experiment I, we decode Part-of-Speech, the most basic syntactic feature, from word embed-

[2]SG results for some languages are missed due to the lack of the corpus data or special preprocessing.

ISO	Language	$\|V\|$	CW	SG	AE	
ar	Arabic	24967	0.712	0.658	0.648	I
ga	Irish	3164	0.826	–	0.697	
zh	Chinese	30496	0.780	0.721	n/a	II
fa	Persian	11471	0.895	0.827	0.746	III
la	Latin	6678	0.746	–	0.707	
hi	Hindi	12703	0.858	0.799	0.592	IV
ta	Tamil	1940	0.768	–	0.541	
eu	Basque	11212	0.857	–	0.711	
et	Estonian	2166	0.862	0.765	0.530	V
fi	Finnish	26086	0.910	0.818	0.715	
hu	Hungarian	6105	0.912	0.831	0.674	
de	German	29899	0.916	0.902	0.74	VI
fr	French	29445	0.905	0.889	0.759	VII
pt	Portuguese	17715	0.927	0.903	0.746	
he	Hebrew	22754	0.911	–	0.680	
ru	Russian	55416	0.959	0.913	0.906	
hr	Croatian	12581	0.926	0.862	0.790	
da	Danish	10705	0.913	0.913	0.666	
sv	Swedish	8408	0.938	0.888	0.670	
no	Norwegian	18709	0.926	0.861	0.704	
sl	Slovenian	19514	0.919	0.820	0.756	
cs	Czech	55789	0.949	0.883	0.853	
ro	Romanian	3170	0.858	0.814	0.618	
en	English	15116	0.857	0.839	0.659	
id	Indonesian	15635	0.852	0.819	0.801	
it	Italian	21184	0.902	0.880	0.700	VIII
es	Spanish	33696	0.906	0.883	0.75	
el	Greek	8499	0.937	0.879	0.801	
pl	Polish	18062	0.941	0.842	0.800	
bg	Bulgarian	17079	0.920	0.852	0.741	

Table 2: Model comparison on decoding POS, along with WALS word-order features. Type I: VS+VO+Pre+NR. II: SV+VO+Pre+RN. III: SV+OV+Pre+NR. IV: SV+OV+Post+RN/Co. V: SV+OV+Post+NR. VI: SV+ND+Pre+NR. VII: SV+VO+Pre+NR. VIII: ND+VO+Pre+NR.

ding. To construct the POS vector for each word, we calculate the normalized POS-tag frequency distribution from the manual annotation of the Universal Dependencies (Version 1.2) (De Marneffe et al., 2014) and Chinese Treebank (CTB 7.0) (Xue et al., 2010) for each language.

We evaluate the predicted results by judging whether the most probable POS tag of a word predicted by the model equals to the most probable correct POS tag of the word. Formally, for a set of words W in a language, the correct tag of the i^{th} word W_i is $y^a_{W_i}$ and the predicted tag is $\hat{y}^a_{W_i}$. The accuracy is computed as:

$$acc = \frac{1}{|W|}\sum_{i}^{|W|}\Delta(\hat{y}^a_{W_i}, y^a_{W_i}) \quad (1)$$

$$\Delta(\hat{y}^a_{W_i}, y^a_{W_i}) = \begin{cases} 1 & \hat{y}^a_{W_i} = y^a_{W_i} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

It is obvious that context-based representation (CW and SG) performs better than character-based representation (AE). We, however, notice that AE peforms nearly as well as the context-based embedding on Russian, Czech and Indonesian.

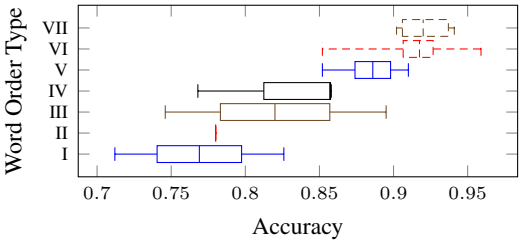


Figure 2: Interaction between CW performances on decoding POS tag and WALS word order features.

It turns out that these languages employ affix markers to indicate the POS category of a word. For example, in Indonesian, co-occurrence of the prefix *'me-'* and the suffix *'-kan'* in the word form means that this word is a verb.

Besides, we explore the relationship between CW performances on decoding POS tags and the word order typology of different languages, since CW is sensitive to word order. We classify the languages into 8 types, based on the basic word order features (Order of Subject and Verb; Order of Object and Verb; Order of Noun and Adposition; Order of Noun and Relative clause) from the World Atlas of Language Structures (Dryer and Haspelmath, 2013). Figure 2 shows that CW performs similar in this experiment for languages of the same word order type, indicating an implicit interaction between typological diversity and model performance.

4.2 Dependency Relation

In this section, we will get into the details of Experiment II: decoding dependency relation from word representation. Dependency relation refers to how a word is syntactically related to other words in a sentence. It is the label annotated on the arc of the dependency tree.

We compute the normalized frequency distribution of dependency relations for each word in the Universal Dependency Treebank and Chinese Treebank (CTB 7.0) (Xue et al., 2010). The distribution of dependency relations is the probabilistic distribution of different arc types, such as subject, object, nmod, etc. Evaluation is similar to that in Section 4.1.

We can see from Figure 3 that the overall performance is worse than that in Experiment I, as dependency analysis is more difficult than POS induction. CW achieves the best performance. It

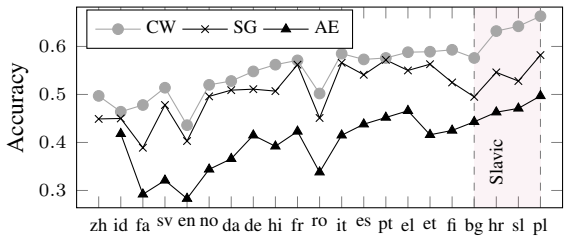


Figure 3: Comparison of models on decoding DEPENDENCY RELATION.

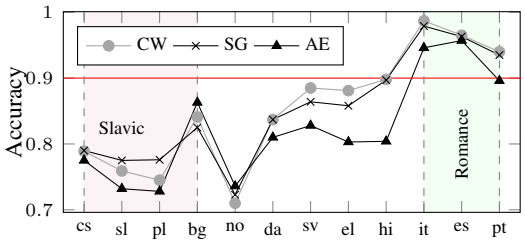


Figure 4: Comparison of models on decoding GENDER.

is also interesting to see that all the embeddings work slightly better on Slavic languages.

4.3 Morphological Features

Experiment III aims to decode morphological information from various word representation. Evaluation is similar to that in Section 4.1. Morphological information refers to the explicit marker of the grammatical functions. We consider 12 morphological features, as is shown in Table 1. They can be split into 5 nominal features (GENDER, NUMBER, CASE, ANIMACY, DEFINITENESS) and 7 verbal features (PERSON, TENSE, ASPECT, MOOD, VOICE, PRONOUNTYPE, VERBFORM).

Gender is a very special feature for western languages. It is partially based on semantics, such as biological sex. In most of the languages with gender features, there are agreements between the noun and the determiners. This could be a good indicator for context-based model. On the other hand, gender is also expressed as an inflectional feature via declension or umlaut, especially for adjectives and verbs. Therefore, we can see from Figure 4 that the AE also achieves some good results without using context information.

From a typological perspective, we found that all the embeddings work well on decoding word gender of Romance languages (Italian, Spanish and Portuguese) but worst on Slavic languages (e.g. Czech, Slovenian). This is probably

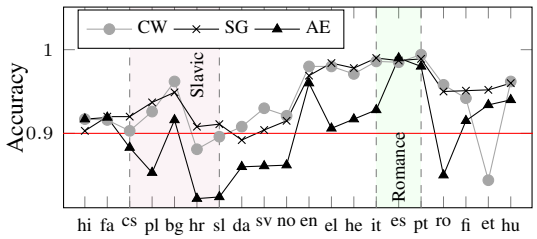


Figure 5: Model comparison on decoding NUMBER.

	Language	$\|V\|$	C&W	SG	AE	# Case
Analy.	Danish	372	0.947	0.946	1.000	3
	Swedish	5893	0.995	0.990	0.981	2
	Bulgarian	104	0.636	0.546	0.818	4
Agglut.	Finnish	21094	0.868	0.871	**0.908**	15
	Hungarian	4536	0.852		**0.901**	22
	Tamil	1144	0.896	–	**0.835**	7
	Basque	8020	0.761	–	**0.857**	15
Fusional	Hindi	10682	0.712	0.704	0.646	7
	Czech	38666	0.788	0.776	0.663	7
	Polish	13715	0.828	0.785	0.636	7
	Slovenian	15150	0.796	0.768	0.617	6
	Croatian	9945	0.807	0.789	0.628	7
	Greek	5790	0.841	0.851	0.774	5
	Latin	4773	0.674	–	0.636	7

Table 3: Model comparison on decoding CASE.

because that Romance languages employ regular rules to judge the gender of a word. However, Slavic languages have other nonlinear fusional morphological features that are not easy to tackle.

Number refers to the linguistic abstraction of objects' quantities. It is an inflectional feature of noun and other parts of speech (adjective, verb) that have agreement with noun. The basic value can be singular, dual or plural. We can see from Figure 5 that SG, CW and AE all perform well. AE performs almost as well as CW and SG on English, Spanish and Portuguese.

Case is one of the most significant features. Gender and number are indexical morphemes, which means that there is a phrase in the sentence that necessarily agrees with the target item. Case, on the contrary, is a relational morpheme, according to (Croft, 2002). Case reflects the semantic role of a noun, relative to the pivot verb. All the languages studied in this paper, more or less, employ word inflection to explicitly express the specific case role. The model performances are listed in Table 3.

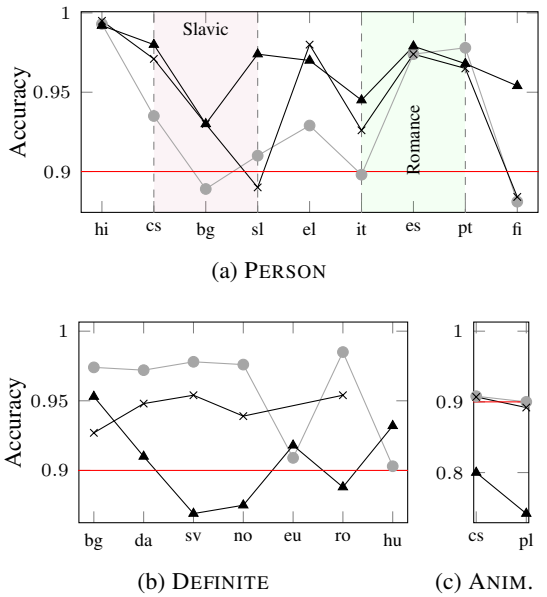
We notice some important inter-language differences. Swedish has only two cases, nominal and genitive. The form of genitive case is very simple. Adding an *s* to the coda of a noun will change it to genitive case. Thus, we can see that character-based encoding performs well

on Swedish. Since genitive case usually means possession, we also notice that context-based distributed representation also performs well in decoding case information from Swedish words.

By classifying these languages into different morphological types in Table 3, we find that word vectors of highly inflected fusional languages (e.g. Czech) performs worse than agglutinative languages (e.g. Finnish). This is typically reflected in AE, as agglutinative languages simply concatenate the case marker with the nominative form of a noun. The morphological transformation of agglutinative languages is linear and simple. Besides, the case system of the analytic languages has been largely simplified due to historical change. Therefore, all the embeddings perform well on analytic languages. This evidence supports that morphological complexity is positively correlated with the quality of word embedding.

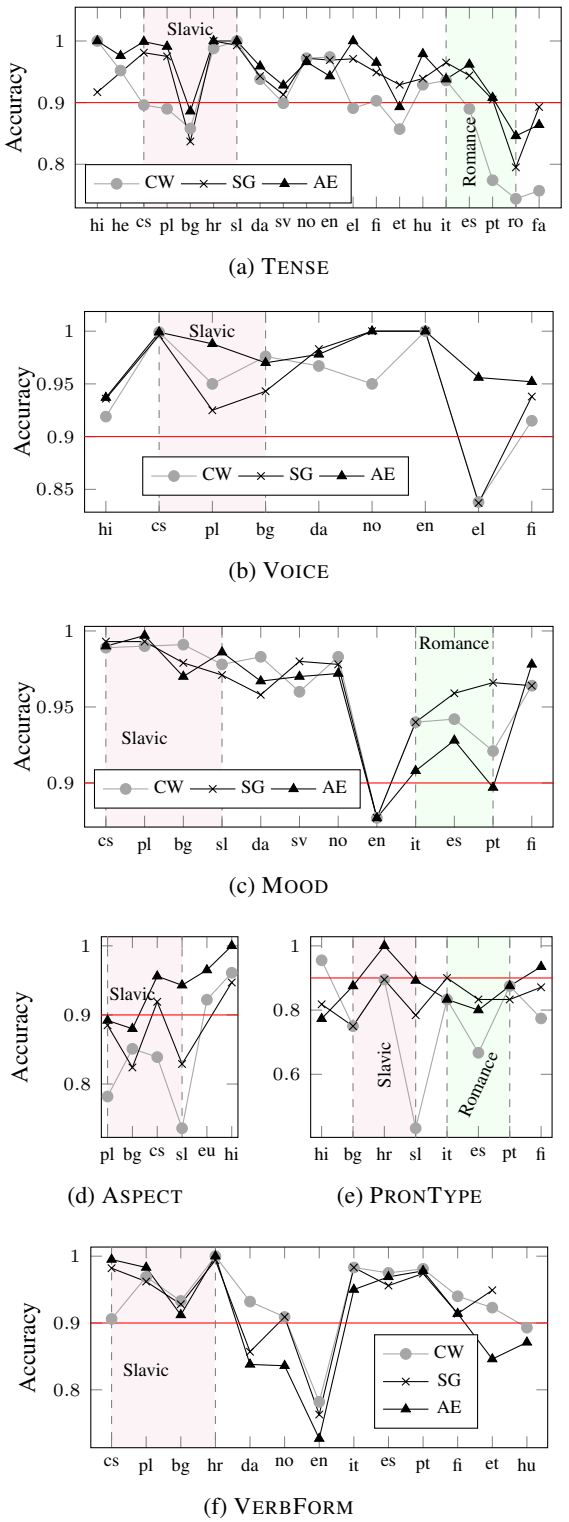
Besides, for fusional languages, using distributed representation and context information would largely increase the performance. This, in turn, indicates that cases are a special semantic relations distributed in the words around the target noun. Although a case is not explicitly agreed with other components in an utterance, the word category might serve as a good indicator, such as preposition and verb.

Animacy is a special nominal feature in a few languages, which is used to discriminate alive and animate objects from inanimate nouns. Generally, it is based on the lexical semantic feature. As is shown in Figure 6, it is easier to decode animacy from the context-based representations than character-based representation.



(a) PERSON (b) DEFINITE (c) ANIM.

Figure 6: Model comparison on decoding PERSON, DEFINITENESS and ANIMACY.



(a) TENSE (b) VOICE (c) MOOD (d) ASPECT (e) PRONTYPE (f) VERBFORM

Figure 7: Model comparison on TENSE, VOICE, MOOD, ASPECT, PRONTYPE and VERBFORM

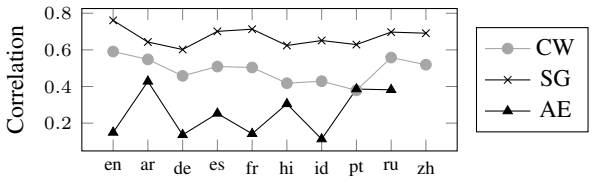


Figure 8: Model comparison on EMOTION.

From Figure 6, 7, we can see that all the three models give quite perfect performance on decoding person, definiteness, tense, voice, mood, aspect, pronoun type and verb form.

Overall, character-based representation is most effective for Slavic languages on decoding verbal morphological features but not nominal features. The result is vice versa for Romance languages, which is not as morphologically complex as Slavic. It is worth noticing that models behave differently on Bulgarian, an analytic language, although Bulgarian belongs to Slavic language from the phylogenetic perspective. We think that this is because many morphological features in Bulgarian have been simplified or weakened.

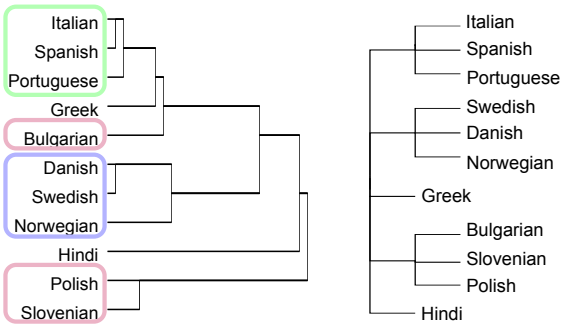
4.4 Emotion Score

In Experiment IV, we use the manually annotated data collected by Dodds et al. (2015). The data contains emotion scores for a list of words in several languages. In our experiment, the original score scale is transformed into the interval $[0, 1]$. A nonlinear map is trained to regress the representation of a word (CW, SG, AE) to its emotion score.

To evaluate the predicted results, we measure the Spearman correlation between the gold scores and predicted scores. The result in Figure 8 reveals a significantly strong correlation between the predicted emotion scores of SG and the real emotion scores. CW comes the second. For AE, it is hard to decode emotion just from the word form.

5 Contrastive Analysis

As we have mentioned before, Type I C&W model utilizes ordered context information to train the distributed word representation. Type II skip-gram model utilizes unordered context information. Type III character-based LSTM autoencoder model utilizes the grapheme information to represent a word. Towards the key questions that we raised at the very beginning of the paper, we propose our contrastive analysis based on the experiment results.



(a) Hierarchical tree based on model performances

(b) WALS Genus Tree

Figure 9: Comparison of the tree based on model performances and the WALS dendrogram manually constructed by linguists.

5.1 Typology vs. Phylogeny

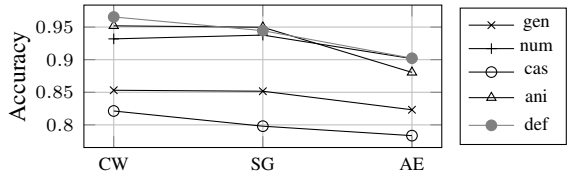
Experiment results have shown that word embedding models are influenced by the syntactic and morphological diversity more or less. Here we display how typological similarity and phylogenetic relation is revealed from the observed model performance variation. We hierarchically cluster languages according to the model performance on decoding syntactic and morphological features. The dendrogram of the languages in Figure 9 vividly shows that most of the phylogenetic-related languages are clustered together.

However, there is some interesting exceptions. Bulgarian does not form a primary cluster with other Slavic languages (e.g. Slovenian). We think that this is because Bulgarian is typologically dissimilar to Slavic language family. Therefore, Figure 9 reflects that language typology explains the model variation better than language phylogeny.

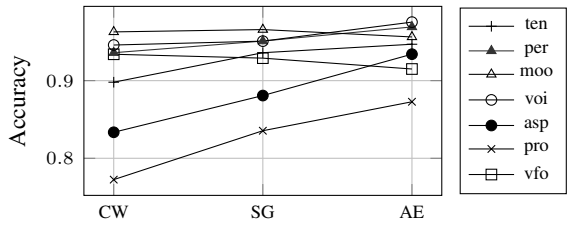
5.2 Form vs. Context

Here we discuss the effectiveness of word form and different types of word context.

Regarding the correlation between context type and language function, previous results show that SG performs worse than CW on decoding POS and dependency relation while SG performs better than CW on decoding emotion score. Since CW keeps word order of the context, this comparison suggests that word order information is vital to syntactic information, but it might also be a kind of noise for the word vectors to encode semantic information.



(a) Nominal features



(b) Verbal features

Figure 10: Overall performances of different models (averaged over languages) on decoding morphological features.

Lan.	Raw	Shuf.	Lan.	Raw	Shuf.
Russian	0.906	0.671	Slovenian	0.800	0.653

Table 4: Comparison of original and shuffled character-based word representation on decoding POS tag.

Source Language	**Arabic**		**Finnish**		
Target Language	fa	ud	en	shuf en	rand
Bigram type overlap.	0.176	0.761	0.891	0.864	0.648
Bigram token overlap.	0.689	0.881	0.999	0.993	0.650
Trigram type overlap.	0.523	0.522	0.665	0.449	0.078
Trigram token overlap.	0.526	0.585	0.978	0.796	0.078
Reconstruction Acc.	0.586	0.689	0.95	0.83	0.22

Table 5: Comparison of morpho-phonological knowledge transfer on different language pairs. The reconstruction accuracy is correlated with the overlapping proportion of grapheme patterns between source language and target language.

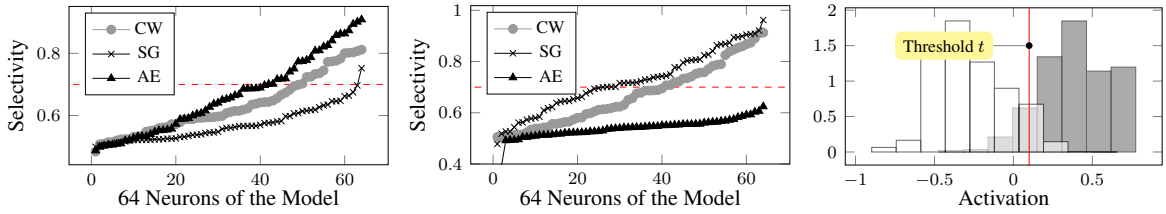
Regarding the correlation between form and language function, previous results on POS, dependency relation and emotion scores show the effectiveness of the word context. However, for morphological features, results in Table 10 indicate that context-based word representation works slightly better than character-based representation. Specifically, character-based embedding (AE) does outperform context-based embedding (CW, SG) on decoding verbal morphological features, even though AE does not access any context information. In other words, word form could be an explicit and discriminative cue for the model to decode the morphological feature of a word.

To prove that word form could provides informative and explicit cues for grammatical functions, we train another shuffled character-based word representation, which means that the autoencoder inputs shuffled letters and outputs the shuffled letters again. We use the hidden layer of the shuffled autoencoder as the representation for each word. The result in Table 4 shows that now the character-based model cannot perform as well as the original character-based autoencoder representation does, which again proves that the order of the word form is necessary for learning the grammatical function of a word.

Since many languages share similar phonographic writing systems, we naturally want to know whether the grapheme-phoneme knowledge from one language can be transferred to another language. We train an autoencoder purely on Finnish and directly test the trained model on memorizing raw English words, letter-shuffled English words and random letter sequences. Results in Table 5 indicate that the character autoencoder can successfully reconstruct raw English words instead of the letter-shuffled English words or random letter sequences. However, if we train an autoencoder purely on Arabic and then directly test the trained model on memorizing Urdu (ud) words or Persian (fa) words, the reconstruction accuracy is quite low, although Arabic, Persian and Urdu use the same Arabic writing system.

To explain the behaviour of AE, we calculate the correlation between the bigram character frequency in the words of the training language (e.g. Finnish) and the bigram character frequency in the words of the testing language (e.g. English). Table 5 reveals that phonological knowledge can be transferred if two languages share similar bigram and trigram character frequency distribution. For example, Finnish and English are both Indo-European language. Their writing system stores similar phonological structure. Arabic is a Semitic language. Persian is an Indo-European language. Their writing system stores different phonological structures respectively. This again proves that character-based LSTM autoencoder does 'memorize' the grapheme or phoneme clusters of a words. Morpho-phonological knowledge can be transferred among typologically-related languages.

Additionally, we are surprised to find that using the English word representations encoded



(a) me-- kan (b) Country names (c) Single neuron

Figure 11: Visualising the Neuron activation pattern for different word embedding models

by AE model trained on Finnish can increase the accuracy of English AE embedding in Experiment I (up to 0.7076), compared with the original accuracy 0.6587. This is probably due to the shared knowledge about the morphemes in the word form.

5.3 Neuronal Activation Pattern

Le (2011) found out that it is empirically possible to learn a 'Grandmother neuron'-like face detector from unlabelled data. In their experiment on unlabeled data, one neuron learns to activate specifically towards the pictures with cat faces instead of other pictures. Based on this finding, we hypothesize that there should exist selective neuron activation towards a linguistic feature trigger. The feature trigger can be a special consonant cluster, a specific suffix or the syntactic category of a word.

To quantitatively show the collective neuron behaviours and the individual neuron response towards different linguistic trigger, we compute the maximum probability that a neuron discriminates the words with trigger f from the words without trigger f. We defined this probability as the Degree of Selectivity p. For a given neuron $\mathbf{n}$ in a given model $\mathbf{M}$ towards linguistic trigger f, we try to find a threshold t that maximizes $p_{f,t}$,

$$c_{f,t} = \frac{N^+_{f,t}}{N_f}, c_{\neg f,t} = \frac{N^+_{\neg f,t}}{N_{\neg f}},$$

$$Selectivity = p_{f,t} = \frac{2 \times c_{f,t} \times c_{\neg f,t}}{c_{f,t} + c_{\neg f,t}}$$

where $N^+_{f,t}$ is the number of correctly discriminated words with linguistic feature f based on the threshold t. N_f is the real number of words with linguistic feature f. $N^+_{\neg f,t}$ is the number of correctly discriminated words without linguistic feature f based on the threshold t. $N_{\neg f}$ means the real number of words without linguistic feature f. $c_{f,t}$ / $c_{\neg f,t}$ is the accuracy for the neuron $\mathbf{n}$ of model $\mathbf{M}$ to detect the existence / nonexistence of the linguistic feature f. $p_{f,t}$ is the F-score of $c_{f,t}$ and $c_{\neg f,t}$, indicating the degree to which a certain neuron discriminates the words with/without a certain trigger f at a certain threshold t.

After calculating the selectivity of 64 neurons in an embedding model towards a linguistic trigger f, we sort the neurons according to the value of selectivity and draw the curve in Figure 11 for each model. The x-axis is the rank of the model neurons based on their selectivity towards a certain linguistic trigger. The y-axis is the selectivity of the corresponding neuron. The curve can tell us how many neurons selectively respond to trigger f to a certain degree. For example, we can see from Figure 11 that the max selectivity of the AE neurons reaches nearly 0.9. This means that one neuron of the AE model is especially sensitive to the prefix *'Me-'* and affix *'-an'*. It can detect the words with the prefix *'Me-'* and the affix *'-an'* just from its activation pattern.

It is also interesting to see from Figure 11 that neurons of AE respond more selectively to morphological triggers than those of the word-based model. For example, almost 30% of the AE neurons fall in the selectivity level [0.7, 1] towards the verb marker, namely prefix *'Me-'* and affix *'-an'*, in Indonesian. Context-based model also shows some selectivity towards this morphological triggers. For SG model, the max selectivity of the model neurons is only just above 0.7.

On the contrary, the context-based distributed models showed strong selective activation towards country names in Indonesian. However, the selectivity of all the AE neurons is below 0.7 towards these semantically-related words.

Similar patterns are found also in other languages. We conclude that the character-based model captures much morphological information

/ syntactic marker than semantic information. The popular word-based model captures both semantic information and syntactic information, although the latter is not displayed as explicitly as the former.

6 Related works

There have been a lot of research on interpreting or relating word embedding with linguistic features. Yogatama et al. (2014) projects word embedding into a sparse vector. They found some linguistically interpretable dimensions. Faruqui and Dyer (2015) use linguistic features to build word vector. Their results show that these representation of word meaning can also achieve good performance in the analogy and similarity tasks. These work can be regarded as the foreshadowing of our experiment paradigm that mapping dense vector to a sparse linguistic property space.

Besides, a lot of study focus on empirical comparison of different word embedding model. Melamud et al. (2016) investigates the influence of context type and vector dimension on word embedding. Their main finding is that concatenating two different types of embeddings can still improve performance even if the utility of dimensionality has run out. Andreas and Klein (2014) assess the potential syntactic information encoded in word embeddings by directly apply word embeddings to parser and they concluded that embeddings add redundant information to what the conventional parser has already extracted. Tsvetkov et al. (2015) propose a method to evaluate word embeddings through the alignment of distributional vectors and linguistic word vectors. However, the method still lacks a direct and comprehensive investigation of the utility of form, context and language typological diversity. This is exactly our novelty and contribution.

It is worth noticing that Köhn (2015) evaluates multilingual word embedding and compares skip-gram, language model and other competitive embedding models. They show that dependency-based skip-gram embedding is effective, even at low dimension. Although Köhn (2015) work involves different languages, they focus on the similarity among multilingual embeddings with only 7 languages. Our work, however, not only provides a comprehensive investigation with massive language samples (30 for Experiment I) and nonlinear mapping models, but also reveal the utility of pure word form and novelly point out the cross-language differences in word representation, which have been overlooked by huge amount of monolingual/bilingual research on well-studied languages.

7 Conclusion

In this paper, we quantify the utility of word form and the effect of language typological diversity in learning word representations. Cross-language perspective and novel analysis of neuron behaviours provide us with new evidence about the typological universal and specific revealed in word embedding. We summarize from our experiments on a massive set of languages that:

- Language typological diversity, especially the specific word order type and morphological complexity, does influence how linguistic information is encoded in word embedding.
- It is plausible (and sometimes even better) to decode grammatical function just from the word form, for certain inflectional languages.
- Quantification of neuron activation pattern reveals different characteristics of the context-based model and the character-based counterpart.

Therefore, we think that it is necessary to maximize both the utility of word form and the advantage of the context for a better word representation. It would also be a promising direction to incorporate the factor of language typological diversity when designing advanced word representation model for languages other than English.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proceedings of the*

Seventeenth Conference on Computational Natural Language Learning, pages 183–192, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jacob Andreas and Dan Klein. 2014. How much do word embeddings encode about syntax? In *Proceedings of ACL*, pages 822–827.

Miguel Ballesteros, Chris Dyer, and Noah A Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of EMNLP*.

Emily M Bender. 2009. Linguistically naïve!= language independent: why nlp needs linguistic typology. In *Proceedings of the EACL 2009 Workshop on the Interaction between Linguistics and Computational Linguistics: Virtuous, Vicious or Vacuous?*, pages 26–32. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

William Croft. 2002. *Typology and universals*. Cambridge University Press.

Marie-Catherine De Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. 2014. Universal stanford dependencies: A cross-linguistic typology. In *LREC*, volume 14, pages 4585–4592.

Peter Sheridan Dodds, Eric M Clark, Suma Desu, Morgan R Frank, Andrew J Reagan, Jake Ryland Williams, Lewis Mitchell, Kameron Decker Harris, Isabel M Kloumann, James P Bagrow, et al. 2015. Human language reveals a universal positivity bias. *Proceedings of the National Academy of Sciences*, 112(8):2389–2394.

Matthew S. Dryer and Martin Haspelmath, editors. 2013. *WALS Online*. Max Planck Institute for Evolutionary Anthropology. `http://wals.info/`.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. In *Proceedings of ACL*.

Manaal Faruqui, Yulia Tsvetkov, Graham Neubig, and Chris Dyer. 2016. Morphological inflection generation using character sequence to sequence learning. In *Proceedings of NAACL*.

Zellig S. Harris. 1954. Distributional structure. *Synthese Language Library*, 10:146–162.

Maria Jesus Aranzabe Masayuki Asahara Aitziber Atutxa Miguel Ballesteros John Bauer Kepa Bengoetxea Riyaz Ahmad Bhat Cristina Bosco Sam Bowman Giuseppe G. A. Celano Miriam Connor Marie-Catherine de Marneffe Arantza Diaz de Ilarraza Kaja Dobrovoljc Timothy Dozat Tomaž Erjavec Richárd Farkas Jennifer Foster Daniel Galbraith Filip Ginter Iakes Goenaga Koldo Gojenola Yoav Goldberg Berta Gonzales Bruno Guillaume Jan Hajič Dag Haug Radu Ion Elena Irimia Anders Johannsen Hiroshi Kanayama Jenna Kanerva Simon Krek Veronika Laippala Alessandro Lenci Nikola Ljubešić Teresa Lynn Christopher Manning Ctlina Mrnduc David Mareček Héctor Martínez Alonso Jan Mašek Yuji Matsumoto Ryan McDonald Anna Missilä Verginica Mititelu Yusuke Miyao Simonetta Montemagni Shunsuke Mori Hanna Nurmi Petya Osenova Lilja Øvrelid Elena Pascual Marco Passarotti Cenel-Augusto Perez Slav Petrov Jussi Piitulainen Barbara Plank Martin Popel Prokopis Prokopidis Sampo Pyysalo Loganathan Ramasamy Rudolf Rosa Shadi Saleh Sebastian Schuster Wolfgang Seeker Mojgan Seraji Natalia Silveira Maria Simi Radu Simionescu Katalin Simkó Kiril Simov Aaron Smith Jan Štěpánek Alane Suhr Zsolt Szántó Takaaki Tanaka Reut Tsarfaty Sumire Uematsu Larraitz Uria Viktor Varga Veronika Vincze Zdeněk Žabokrtský Daniel Zeman Joakim Nivre, Željko Agić and Hanzhi Zhu. 2015. Universal dependencies 1.2. In *LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2016. Character-aware neural language models. In *Proceedings of AAAI*.

Arne Köhn. 2015. Whats in an embedding? analyzing word embeddings through multilingual evaluation. In *Poceedings of EMNLP*.

Siwei Lai, Kang Liu, Liheng Xu, and Jun Zhao. 2015. How to generate a good word embedding? *arXiv preprint arXiv:1507.05523*.

Q. V. Le. 2011. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595 – 8598.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of EMNLP*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Oren Melamud, David McClosky, Siddharth Patwardhan, and Mohit Bansal. 2016. The role of context types and dimensionality in learning word embeddings. In *Proceedings of NAACL*.

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

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of EMNLP*.

Nianwen Xue, Zixin Jiang, Xiuhong Zhong, Martha Palmer, Fei Xia, Fu-Dong Chiou, and Meiyu Chang. 2010. Chinese treebank 7.0. *Linguistic Data Consortium, Philadelphia*.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A Smith. 2014. Learning word representations with hierarchical sparse coding. In *Proceedings of ICML*.

Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change

William L. Hamilton, Jure Leskovec, Dan Jurafsky
Department of Computer Science, Stanford University, Stanford CA, 94305
wleif,jure,jurafsky@stanford.edu

Abstract

Understanding how words change their meanings over time is key to models of language and cultural evolution, but historical data on meaning is scarce, making theories hard to develop and test. Word embeddings show promise as a diachronic tool, but have not been carefully evaluated. We develop a robust methodology for quantifying semantic change by evaluating word embeddings (PPMI, SVD, word2vec) against known historical changes. We then use this methodology to reveal statistical laws of semantic evolution. Using six historical corpora spanning four languages and two centuries, we propose two quantitative laws of semantic change: (i) the *law of conformity*—the rate of semantic change scales with an inverse power-law of word frequency; (ii) the *law of innovation*—independent of frequency, words that are more polysemous have higher rates of semantic change.

1 Introduction

Shifts in word meaning exhibit systematic regularities (Bréal, 1897; Ullmann, 1962). The rate of semantic change, for example, is higher in some words than others (Blank, 1999) — compare the stable semantic history of *cat* (from Proto-Germanic *kattuz*, "cat") to the varied meanings of English *cast*: "to mould", "a collection of actors', "a hardened bandage", etc. (all from Old Norse *kasta*, "to throw", Simpson et al., 1989).

Various hypotheses have been offered about such regularities in semantic change, such as an increasing subjectification of meaning, or the grammaticalization of inferences (e.g., Geeraerts, 1997; Blank, 1999; Traugott and Dasher, 2001).

But many core questions about semantic change remain unanswered. One is the role of *frequency*. Frequency plays a key role in other linguistic changes, associated sometimes with faster change—sound changes like lenition occur in more frequent words—and sometimes with slower change—high frequency words are more resistant to morphological regularization (Bybee, 2007; Pagel et al., 2007; Lieberman et al., 2007). What is the role of word frequency in meaning change?

Another unanswered question is the relationship between semantic change and *polysemy*. Words gain senses over time as they semantically drift (Bréal, 1897; Wilkins, 1993; Hopper and Traugott, 2003), and polysemous words[1] occur in more diverse contexts, affecting lexical access speed (Adelman et al., 2006) and rates of L2 learning (Crossley et al., 2010). But we don't know whether the diverse contextual use of polysemous words makes them more or less likely to undergo change (Geeraerts, 1997; Winter et al., 2014; Xu et al., 2015). Furthermore, polysemy is strongly correlated with frequency—high frequency words have more senses (Zipf, 1945; İlgen and Karaoglan, 2007)—so understanding how polysemy relates to semantic change requires controling for word frequency.

Answering these questions requires new methods that can go beyond the case-studies of a few words (often followed over widely different time-periods) that are our most common diachronic data (Bréal, 1897; Ullmann, 1962; Blank, 1999; Hopper and Traugott, 2003; Traugott and Dasher, 2001). One promising avenue is the use of distributional semantics, in which words are embedded in vector spaces according to their co-occurrence relationships (Bullinaria and Levy, 2007; Turney and Pantel, 2010), and the embeddings of words

[1] We use 'polysemy' here to refer to related senses as well as rarer cases of accidental homonymy.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1489–1501,
Berlin, Germany, August 7-12, 2016.

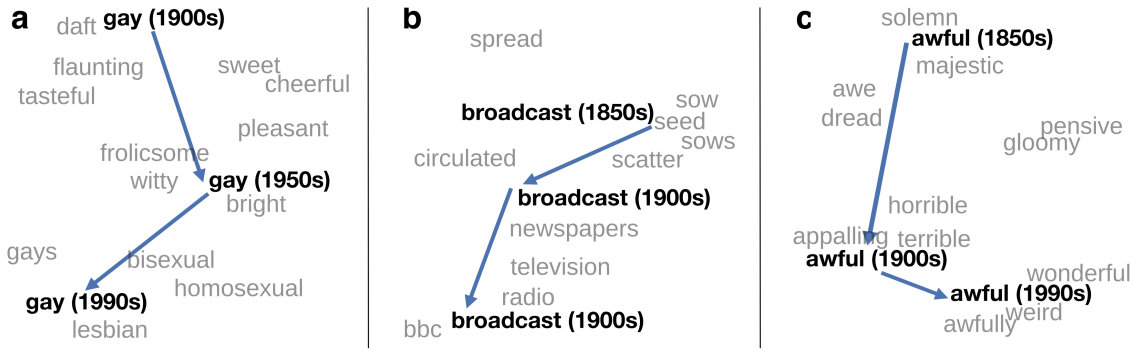


Figure 1: Two-dimensional visualization of semantic change in English using SGNS vectors.[2] **a**, The word *gay* shifted from meaning "cheerful" or "frolicsome" to referring to homosexuality. **b**, In the early 20th century *broadcast* referred to "casting out seeds"; with the rise of television and radio its meaning shifted to "transmitting signals". **c**, *Awful* underwent a process of pejoration, as it shifted from meaning "full of awe" to meaning "terrible or appalling" (Simpson et al., 1989).

are then compared across time-periods. This new direction has been effectively demonstrated in a number of case-studies (Sagi et al., 2011; Wijaya and Yeniterzi, 2011; Gulordava and Baroni, 2011; Jatowt and Duh, 2014) and used to perform large-scale linguistic change-point detection (Kulkarni et al., 2014) as well as to test a few specific hypotheses, such as whether English synonyms tend to change meaning in similar ways (Xu and Kemp, 2015). However, these works employ widely different embedding approaches and test their approaches only on English.

In this work, we develop a robust methodology for quantifying semantic change using embeddings by comparing state-of-the-art approaches (PPMI, SVD, word2vec) on novel benchmarks.

We then apply this methodology in a large-scale cross-linguistic analysis using 6 corpora spanning 200 years and 4 languages (English, German, French, and Chinese). Based on this analysis, we propose two statistical laws relating frequency and polysemy to semantic change:

- **The law of conformity:** Rates of semantic change scale with a negative power of word frequency.
- **The law of innovation:** After controlling for frequency, polysemous words have significantly higher rates of semantic change.

2 Diachronic embedding methods

The following sections outline how we construct diachronic (historical) word embeddings, by first constructing embeddings in each time-period and then aligning them over time, and the metrics that we use to quantify semantic change. All of the learned embeddings and the code we used to analyze them are made publicly available.[3]

2.1 Embedding algorithms

We use three methods to construct word embeddings within each time-period: PPMI, SVD, and SGNS (i.e., word2vec).[4] These distributional methods represent each word w_i by a vector $\mathbf{w}_i$ that captures information about its co-occurrence statistics. These methods operationalize the 'distributional hypothesis' that word semantics are implicit in co-occurrence relationships (Harris, 1954; Firth, 1957). The semantic similarity/distance between two words is approximated by the cosine similarity/distance between their vectors (Turney and Pantel, 2010).

2.1.1 PPMI

In the PPMI representations, the vector embedding for word $w_i \in \mathcal{V}$ contains the positive point-wise mutual information (PPMI) values between w_i and a large set of pre-specified 'context' words. The word vectors correspond to the rows of the matrix $\mathbf{M}^{\text{PPMI}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}_C|}$ with entries given by

$$\mathbf{M}^{\text{PPMI}}_{i,j} = \max\left\{\log\left(\frac{\hat{p}(w_i, c_j)}{\hat{p}(w)\hat{p}(c_j)}\right) - \alpha, 0\right\}, \quad (1)$$

where $c_j \in \mathcal{V}_C$ is a context word and $\alpha > 0$ is a negative prior, which provides a smoothing bias (Levy et al., 2015). The $\hat{p}$ correspond to the smoothed empirical probabilities of word

[2] Appendix B details the visualization method.

[3] http://nlp.stanford.edu/projects/histwords

[4] Synchronic applications of these three methods are reviewed in detail in Levy et al. (2015).

Name	Language	Description	Tokens	Years	POS Source
ENGALL	English	Google books (all genres)	8.5×10^{11}	1800-1999	(Davies, 2010)
ENGFIC	English	Fiction from Google books	7.5×10^{10}	1800-1999	(Davies, 2010)
COHA	English	Genre-balanced sample	4.1×10^{8}	1810-2009	(Davies, 2010)
FREALL	French	Google books (all genres)	1.9×10^{11}	1800-1999	(Sagot et al., 2006)
GERALL	German	Google books (all genres)	4.3×10^{10}	1800-1999	(Schneider and Volk, 1998)
CHIALL	Chinese	Google books (all genres)	6.0×10^{10}	1950-1999	(Xue et al., 2005)

Table 1: Six large historical datasets from various languages and sources are used.

(co-)occurrences within fixed-size sliding windows of text. Clipping the PPMI values above zero ensures they remain finite and has been shown to dramatically improve results (Bullinaria and Levy, 2007; Levy et al., 2015); intuitively, this clipping ensures that the representations emphasize positive word-word correlations over negative ones.

2.1.2 SVD

SVD embeddings correspond to low-dimensional approximations of the PPMI embeddings learned via singular value decomposition (Levy et al., 2015). The vector embedding for word w_i is given by

$$\mathbf{w}_i^{\text{SVD}} = (\mathbf{U}\mathbf{\Sigma}^{\gamma})_i\,, \tag{2}$$

where $\mathbf{M}^{\text{PPMI}} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top}$ is the truncated singular value decomposition of $\mathbf{M}^{\text{PPMI}}$ and $\gamma \in [0, 1]$ is an eigenvalue weighting parameter. Setting $\gamma < 1$ has been shown to dramatically improve embedding qualities (Turney and Pantel, 2010; Bullinaria and Levy, 2012). This SVD approach can be viewed as a generalization of Latent Semantic Analysis (Landauer and Dumais, 1997), where the term-document matrix is replaced with $\mathbf{M}^{\text{PPMI}}$. Compared to PPMI, SVD representations can be more robust, as the dimensionality reduction acts as a form of regularization.

2.1.3 Skip-gram with negative sampling

SGNS 'neural' embeddings are optimized to predict co-occurrence relationships using an approximate objective known as 'skip-gram with negative sampling' (Mikolov et al., 2013). In SGNS, each word w_i is represented by two dense, low-dimensional vectors: a word vector ($\mathbf{w}_i^{\text{SGNS}}$) and context vector ($\mathbf{c}_i^{\text{SGNS}}$). These embeddings are optimized via stochastic gradient descent so that

$$\hat{p}(c_i|w_i) \propto \exp(\mathbf{w}_i^{\text{SGNS}} \cdot \mathbf{c}_j^{\text{SGNS}}), \tag{3}$$

where $p(c_i|w_i)$ is the empirical probability of seeing context word c_i within a fixed-length window of text, given that this window contains w_i. The SGNS optimization avoids computing the normalizing constant in (3) by randomly drawing 'negative' context words, c_n, for each target word and ensuring that $\exp(\mathbf{w}_i^{\text{SGNS}} \cdot \mathbf{c}_n^{\text{SGNS}})$ is small for these examples.

SGNS has the benefit of allowing incremental initialization during learning, where the embeddings for time t are initialized with the embeddings from time $t - \Delta$ (Kim et al., 2014). We employ this trick here, though we found that it had a negligible impact on our results.

2.2 Datasets, pre-processing, and hyperparameters

We trained models on the 6 datasets described in Table 1, taken from Google N-Grams (Lin et al., 2012) and the COHA corpus (Davies, 2010). The Google N-Gram datasets are extremely large (comprising $\approx$6% of all books ever published), but they also contain many corpus artifacts due, e.g., to shifting sampling biases over time (Pechenick et al., 2015). In contrast, the COHA corpus was carefully selected to be genre-balanced and representative of American English over the last 200 years, though as a result it is two orders of magnitude smaller. The COHA corpus also contains pre-extracted word lemmas, which we used to validate that our results hold at both the lemma and raw token levels. All the datasets were aggregated to the granularity of decades.[5]

We follow the recommendations of Levy et al. (2015) in setting the hyperparameters for the embedding methods, though preliminary experiments were used to tune key settings. For all methods, we used symmetric context windows of size 4 (on each side). For SGNS and SVD, we use embeddings of size 300. See Appendix A for further implementation and pre-processing details.

[5]The 2000s decade of the Google data was discarded due to shifts in the sampling methodology (Michel et al., 2011).

2.3 Aligning historical embeddings

In order to compare word vectors from different time-periods we must ensure that the vectors are aligned to the same coordinate axes. Explicit PPMI vectors are naturally aligned, as each column simply corresponds to a context word. Low-dimensional embeddings will not be naturally aligned due to the non-unique nature of the SVD and the stochastic nature of SGNS. In particular, both these methods may result in arbitrary orthogonal transformations, which do not affect pairwise cosine-similarities within-years but will preclude comparison of the same word across time. Previous work circumvented this problem by either avoiding low-dimensional embeddings (e.g., Gulordava and Baroni, 2011; Jatowt and Duh, 2014) or by performing heuristic local alignments per word (Kulkarni et al., 2014).

We use orthogonal Procrustes to align the learned low-dimensional embeddings. Defining $\mathbf{W}^{(t)} \in \mathbb{R}^{d \times |\mathcal{V}|}$ as the matrix of word embeddings learned at year t, we align across time-periods while preserving cosine similarities by optimizing:

$$\mathbf{R}^{(t)} = \arg \min_{\mathbf{Q}^\top \mathbf{Q} = \mathbf{I}} \|\mathbf{W}^{(t)}\mathbf{Q} - \mathbf{W}^{(t+1)}\|_F, \quad (4)$$

with $\mathbf{R}^{(t)} \in \mathbb{R}^{d \times d}$. The solution corresponds to the best rotational alignment and can be obtained efficiently using an application of SVD (Schönemann, 1966).

2.4 Time-series from historical embeddings

Diachronic word embeddings can be used in two ways to quantify semantic change: (i) we can measure changes in pair-wise word similarities over time, or (ii) we can measure how an individual word's embedding shifts over time.

Pair-wise similarity time-series Measuring how the cosine-similarity between pairs of words changes over time allows us to test hypotheses about specific linguistic or cultural shifts in a controlled manner. We quantify shifts by computing the similarity time-series

$$s^{(t)}(w_i, w_j) = \text{cos-sim}(\mathbf{w}_i^{(t)}, \mathbf{w}_j^{(t)}) \quad (5)$$

between two words w_i and w_j over a time-period $(t, ..., t + \Delta)$. We then measure the Spearman correlation (ρ) of this series against time, which allows us to assess the magnitude and significance of pairwise similarity shifts; since the Spearman correlation is non-parametric, this measure essentially detects whether the similarity series increased/decreased over time in a significant manner, regardless of the 'shape' of this curve.[6]

Measuring semantic displacement After aligning the embeddings for individual time-periods, we can use the aligned word vectors to compute the semantic displacement that a word has undergone during a certain time-period. In particular, we can directly compute the cosine-distance between a word's representation for different time-periods, i.e. cos-dist$(\mathbf{w}_t, \mathbf{w}_{t+\Delta})$, as a measure of semantic change. We can also use this measure to quantify 'rates' of semantic change for different words by looking at the displacement between consecutive time-points.

3 Comparison of different approaches

We compare the different distributional approaches on a set of benchmarks designed to test their scientific utility. We evaluate both their *synchronic* accuracy (i.e., ability to capture word similarity within individual time-periods) and their *diachronic* validity (i.e., ability to quantify semantic changes over time).

3.1 Synchronic Accuracy

We evaluated the synchronic (within-time-period) accuracy of the methods using a standard modern benchmark and the 1990s portion of the ENGALL data. On Bruni et al. (2012)'s MEN similarity task of matching human judgments of word similarities, SVD performed best ($\rho = 0.739$), followed by PPMI ($\rho = 0.687$) and SGNS ($\rho = 0.649$). These results echo the findings of Levy et al. (2015), who found SVD to perform best on similarity tasks while SGNS performed best on analogy tasks (which are not the focus of this work).

3.2 Diachronic Validity

We evaluate the diachronic validity of the methods on two historical semantic tasks: detecting known shifts and discovering shifts from data. For both these tasks, we performed detailed evaluations on a small set of examples (28 known shifts and the top-10 "discovered" shifts by each method). Using these reasonably-sized evaluation sets allowed the authors to evaluate each case rigorously using existing literature and historical corpora.

[6] Other metrics or change-point detection approaches, e.g. mean shifts (Kulkarni et al., 2014) could also be used.

Word	Moving towards	Moving away	Shift start	Source
gay	homosexual, lesbian	happy, showy	ca 1920	(Kulkarni et al., 2014)
fatal	illness, lethal	fate, inevitable	<1800	(Jatowt and Duh, 2014)
awful	disgusting, mess	impressive, majestic	<1800	(Simpson et al., 1989)
nice	pleasant, lovely	refined, dainty	ca 1900	(Wijaya and Yeniterzi, 2011)
broadcast	transmit, radio	scatter, seed	ca 1920	(Jeffers and Lehiste, 1979)
monitor	display, screen	—	ca 1930	(Simpson et al., 1989)
record	tape, album	—	ca 1920	(Kulkarni et al., 2014)
guy	fellow, man	—	ca 1850	(Wijaya and Yeniterzi, 2011)
call	phone, message	—	ca 1890	(Simpson et al., 1989)

Table 2: Set of attested historical shifts used to evaluate the methods. The examples are taken from previous works on semantic change and from the Oxford English Dictionary (OED), e.g. using 'obsolete' tags. The shift start points were estimated using attestation dates in the OED. The first six examples are words that shifted dramatically in meaning while the remaining four are words that acquired new meanings (while potentially also keeping their old ones).

Method	Corpus	% Correct	%Sig.
PPMI	ENGALL	96.9	84.4
	COHA	100.0	88.0
SVD	ENGALL	100.0	90.6
	COHA	100.0	96.0
SGNS	ENGALL	100.0	93.8
	COHA	100.0	72.0

Table 3: Performance on detection task, i.e. ability to capture the attested shifts from Table 2. SGNS and SVD capture the correct directionality of the shifts in all cases (%Correct), e.g., *gay* becomes more similar to *homosexual*, but there are differences in whether the methods deem the shifts to be statistically significant at the $p < 0.05$ level (%Sig).

Detecting known shifts. First, we tested whether the methods capture known historical shifts in meaning. The goal in this task is for the methods to correctly capture whether pairs of words moved closer or further apart in semantic space during a pre-determined time-period. We use a set of independently attested shifts as an evaluation set (Table 2). For comparison, we evaluated the methods on both the large (but messy) ENGALL data and the smaller (but clean) COHA data. On this task, all the methods performed almost perfectly in terms of capturing the correct directionality of the shifts (i.e., the pairwise similarity series have the correct sign on their Spearman correlation with time), but there were some differences in whether the methods deemed the shifts statistically significant at the $p < 0.05$ level.[7] Overall, SGNS performed the best on the full English data, but its performance dropped significantly on the smaller COHA dataset, where SVD performed best. PPMI was noticeably worse than the other two approaches (Table 3).

Discovering shifts from data. We tested whether the methods discover reasonable shifts by examining the top-10 words that changed the most from the 1900s to the 1990s according to the semantic displacement metric introduced in Section 2.4 (limiting our analysis to words with relative frequencies above 10^{-5} in both decades). We used the ENGFIC data as the most-changed list for ENGALL was dominated by scientific terms due to changes in the corpus sample.

Table 4 shows the top-10 words discovered by each method. These shifts were judged by the authors as being either clearly genuine, borderline, or clearly corpus artifacts. SGNS performed by far the best on this task, with 70% of its top-10 list corresponding to genuine semantic shifts, followed by 40% for SVD, and 10% for PPMI. However, a large portion of the discovered words for PPMI (and less so SVD) correspond to borderline cases, e.g. *know*, that have not necessarily shifted significantly in meaning but that occur in different contexts due to global genre/discourse shifts. The poor quality of the nearest neighbors generated by the PPMI algorithm—which are skewed by PPMI's sensitivity to rare events—also made it difficult to assess the quality of its discovered shifts. SVD was the most sensitive to corpus artifacts (e.g., co-occurrences due to cover pages and advertisements), but it still captured a number of genuine semantic shifts.

We opted for this small evaluation set and relied on detailed expert judgments to minimize ambiguity; each potential shift was analyzed in detail by consulting consulting existing literature (especially the OED; Simpson et al., 1989) and all disagreements were discussed.

Table 5 details representative example shifts in English, French, and German. Chinese lacks sufficient historical data for this task, as only years 1950-1999 are usable; however, we do still see

[7] All subsequent significance tests are at $p < 0.05$.

Method	Top-10 words that changed from 1900s to 1990s
PPMI	know, got, would, decided, think, stop, remember, **started**, must, wanted
SVD	harry, **headed**, **calls**, **gay**, wherever, male, **actually**, special, cover, naturally
SGNS	**wanting**, **gay**, **check**, **starting**, **major**, **actually**, touching, harry, **headed**, romance

Table 4: Top-10 English words with the highest semantic displacement values between the 1900s and 1990s. Bolded entries correspond to real semantic shifts, as deemed by examining the literature and their nearest neighbors; for example, *headed* shifted from primarily referring to the "top of a body/entity" to referring to "a direction of travel." Underlined entries are borderline cases that are largely due to global genre/discourse shifts; for example, *male* has not changed in meaning, but its usage in discussions of "gender equality" is relatively new. Finally, unmarked entries are clear corpus artifacts; for example, *special*, *cover*, and *romance* are artifacts from the covers of fiction books occasionally including advertisements etc.

Word	Language	Nearest-neighbors in 1900s	Nearest-neighbors in 1990s
wanting	English	lacking, deficient, lacked, lack, needed	wanted, something, wishing, anything, anybody
asile	French	refuge, asiles, hospice, vieillards, infirmerie	demandeurs, refuge, hospice, visas, admission
widerstand	German	scheiterte, volt, stromstärke, leisten, brechen	opposition, verfolgung, nationalsozialistische, nationalsozialismus, kollaboration

Table 5: Example words that changed dramatically in meaning in three languages, discovered using SGNS embeddings. The examples were selected from the top-10 most-changed lists between 1900s and 1990s as in Table 4. In English, *wanting* underwent subjectification and shifted from meaning "lacking" to referring to subjective "desire", as in "the education system is wanting" (1900s) vs. "I've been wanting to tell you" (1990s). In French *asile* ("asylum") shifted from primarily referring to "hospitals, or infirmaries" to also referring to "asylum seekers, or refugees". Finally, in German *Widerstand* ("resistance") gained a formal meaning as referring to the local German resistance to Nazism during World War II.

some significant changes for Chinese in this short time-period, such as 病毒 ("virus") moving closer to 电脑 ("computer", $\rho = 0.89$).

3.3 Methodological recommendations

PPMI is clearly worse than the other two methods; it performs poorly on all the benchmark tasks, is extremely sensitive to rare events, and is prone to false discoveries from global genre shifts. Between SVD and SGNS the results are somewhat equivocal, as both perform best on two out of the four tasks (synchronic accuracy, ENGALL detection, COHA detection, discovery). Overall, SVD performs best on the synchronic accuracy task and has higher average accuracy on the 'detection' task, while SGNS performs best on the 'discovery' task. These results suggest that both these methods are reasonable choices for studies of semantic change but that they each have their own tradeoffs: SVD is more sensitive, as it performs well on detection tasks even when using a small dataset, but this sensitivity also results in false discoveries due to corpus artifacts. In contrast, SGNS is robust to corpus artifacts in the discovery task, but it is not sensitive enough to perform well on the detection task with a small dataset. Qualitatively, we found SGNS to be most useful for discovering new shifts and visualizing changes (e.g., Figure 1), while SVD was most effective for detecting subtle shifts in usage.

4 Statistical laws of semantic change

We now show how diachronic embeddings can be used in a large-scale cross-linguistic analysis to reveal statistical laws that relate frequency and polysemy to semantic change. In particular, we analyze how a word's rate of semantic change,

$$\Delta^{(t)}(w_i) = \text{cos-dist}(\mathbf{w}_i^{(t)}, \mathbf{w}_i^{(t+1)}) \qquad (6)$$

depends on its frequency, $f^{(t)}(w_i)$ and a measure of its polysemy, $d^{(t)}(w_i)$ (defined in Section 4.4).

4.1 Setup

We present results using SVD embeddings (though analogous results were found to hold with SGNS). Using all four languages and all four conditions for English (ENGALL, ENGFIC, and COHA with and without lemmatization), we performed regression analysis on rates of semantic change, $\Delta^{(t)}(w_i)$; thus, we examined one data-point per word for each pair of consecutive decades and analyzed how a word's frequency and polysemy at time t correlate with its degree of semantic displacement over the next decade. To ensure the robustness of our results, we analyzed only the top-10000 non–stop words by aver-

Top-10 most polysemous	yet, always, even, little, called, also, sometimes, great, still, quite
Top-10 least polysemous	photocopying, retrieval, thirties, mom, sweater, forties, seventeenth, fifteenth, holster, postage

Table 6: The top-10 most and least polysemous words in the ENGFIC data. Words like *yet*, *even*, and *still* are used in many diverse ways and are highly polysemous. In contrast, words like *photocopying*, *postage*, and *holster* tend to be used in very specific well-clustered contexts, corresponding to a single sense; for example, *mail* and *letter* are both very likely to occur in the context of *postage* and are also likely to co-occur with each other, independent of *postage*.

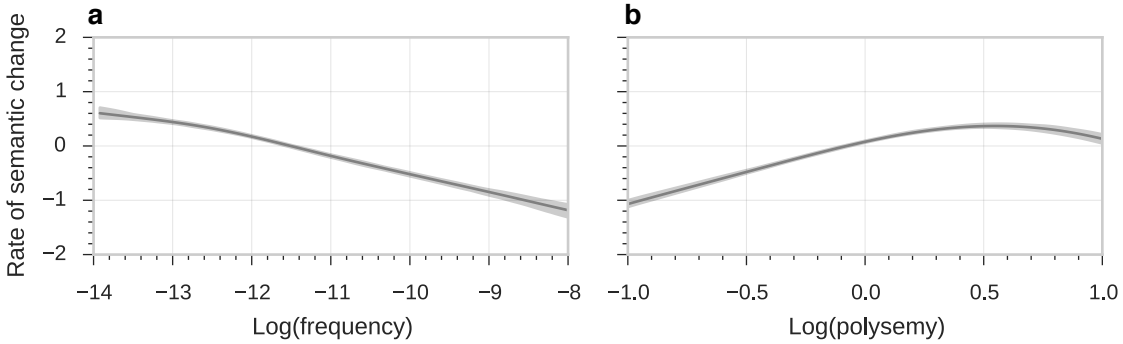


Figure 2: Higher frequency words have lower rates of change (**a**), while polysemous words have higher rates of change (**b**). The negative curvature for polysemy—which is significant only at high $d(w_i)$—varies across datasets and was not present with SGNS, so it is not as robust as the clear linear trend that was seen with all methods and across all datasets. The trendlines show 95% CIs from bootstrapped kernel regressions on the ENGALL data (Li and Racine, 2007).

age historical frequency (lower-frequency words tend to lack sufficient co-occurrence data across years) and we discarded proper nouns (changes in proper noun usage are primarily driven by non-linguistic factors, e.g. historical events, Traugott and Dasher, 2001). We also log-transformed the semantic displacement scores and normalized the scores to have zero mean and unit variance; we denote these normalized scores by $\tilde{\Delta}^{(t)}(w_i)$.

We performed our analysis using a linear mixed model with random intercepts per word and fixed effects per decade; i.e., we fit β_f, β_d, and β_t s.t.

$$\tilde{\Delta}^{(t)}(w_i) = \beta_f \log\left(f^{(t)}(w_i)\right) + \beta_d \log\left(d^{(t)}(w_i)\right) + \beta_t + z_{w_i} + \epsilon_{w_i}^{(t)} \;\; \forall w_i \in \mathcal{V}, t \in \{t_0, ..., t_n\}, \quad (7)$$

where $z_{w_i} \sim \mathcal{N}(0, \sigma_{w_i})$ is the random intercept for word w_i and $\epsilon_{w_i}^{(t)} \in \mathcal{N}(0, \sigma)$ is an error term. β_f, β_d and β_t correspond to the fixed effects for frequency, polysemy and the decade t, respectively[8]. Intuitively, this model estimates the effects of frequency and polysemy on semantic change, while controlling for temporal trends and correcting for the fact that measurements on same word will be correlated across time. We fit (7) using the standard restricted maximum likelihood algorithm (McCulloch and Neuhaus, 2001; Appendix C).

[8]Note that time is treated as a categorical variable, as each decade has its own fixed effect.

4.2 Overview of results

We find that, across languages, rates of semantic change obey a scaling relation of the form

$$\Delta(w_i) \propto f(w_i)^{\beta_f} \times d(w_i)^{\beta_d}, \quad (8)$$

with $\beta_f < 0$ and $\beta_d > 0$. This finding implies that frequent words change at slower rates while polysemous words change faster, and that both these relations scale as power laws.

4.3 Law of conformity: Frequently used words change at slower rates

Using the model in equation (7), we found that the logarithm of a word's frequency, $\log(f(w_i))$, has a significant and substantial negative effect on rates of semantic change in all settings (Figures 2a and 3a). Given the use of log-transforms in pre-processing the data this implies rates of semantic change are proportional to a negative power (β_f) of frequency, i.e.

$$\Delta(w_i) \propto f(w_i)^{\beta_f}, \quad (9)$$

with $\beta_f \in [-1.26, -0.27]$ across languages/datasets. The relatively large range of values for β_f is due to the fact that the COHA datasets are outliers due to their substantially smaller sample sizes (Figure 3; the range is $\beta_f \in [-0.66, -0.27]$ with COHA excluded).

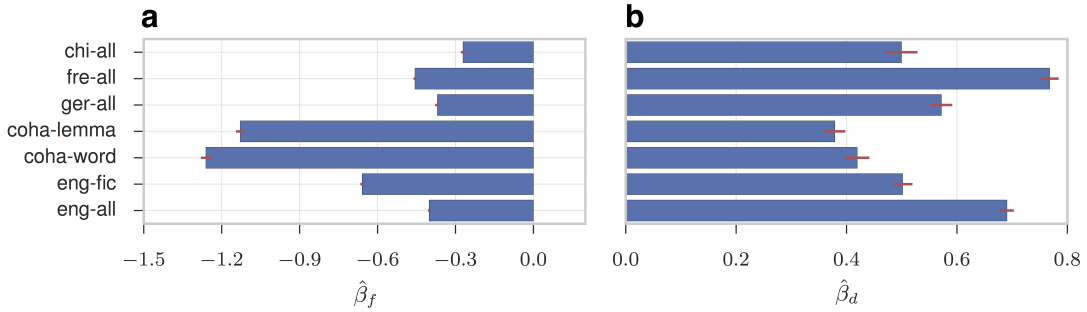


Figure 3: **a**, The estimated linear effect of log-frequency ($\hat{\beta}_f$) is significantly negative across all languages. The effect is significantly stronger in the COHA data, but this is likely due to its small sample size ($\sim$100$\times$ smaller than the other datasets); the small sample size introduces random variance that may artificially inflate the effect of frequency. From the COHA data, we also see that the result holds regardless of whether lemmatization is used. **b**, Analogous trends hold for the linear effect of the polysemy score ($\hat{\beta}_d$), which is strong and significantly positive across all conditions. Again, we see that the smaller COHA datasets are mild outliers.[9] 95% CIs are shown.

4.4 Law of innovation: Polysemous words change at faster rates

There is a common hypothesis in the linguistic literature that "words become semantically extended by being used in diverse contexts" (Winter et al., 2014), an idea that dates back to the writings of Bréal (1897). We tested this notion by examining the relationship between polysemy and semantic change in our data.

Quantifying polysemy

Measuring word polysemy is a difficult and fraught task, as even "ground truth" dictionaries differ in the number of senses they assign to words (Simpson et al., 1989; Fellbaum, 1998). We circumvent this issue by measuring a word's *contextual diversity* as a proxy for its polysemousness. The intuition behind our measure is that words that occur in many distinct, unrelated contexts will tend to be highly polysemous. This view of polysemy also fits with previous work on semantic change, which emphasizes the role of contextual diversity (Bréal, 1897; Winter et al., 2014).

We measure a word's contextual diversity, and thus polysemy, by examining its neighborhood in an empirical co-occurrence network. We construct empirical co-occurrence networks using the PPMI measure defined in Section 2. In these networks words are connected to each other if they co-occur more than one would expect by chance (after smoothing). The polysemy of a word is then measured as its local clustering coefficient within this network (Watts and Strogatz, 1998):

$$d(w_i) = -\frac{\sum_{c_i,c_j \in N_{\text{PPMI}}(w_i)} \mathbb{I}\left\{\text{PPMI}(c_i, c_j) > 0\right\}}{|N_{\text{PPMI}}(w_i)|(|N_{\text{PPMI}}(w_i)| - 1)}, \tag{10}$$

where $N_{\text{PPMI}}(w_i) = \{w_j : \text{PPMI}(w_i, w_j) > 0\}$. This measure counts the proportion of w_i's neighbors that are also neighbors of each other. According to this measure, a word will have a high clustering coefficient (and thus a low polysemy score) if the words that it co-occurs with also tend to co-occur with each other. Polysemous words that are contextually diverse will have low clustering coefficients, since they appear in disjointed or unrelated contexts.

Variants of this measure are often used in word-sense discrimination and correlate with, e.g., number of senses in WordNet (Dorow and Widdows, 2003; Ferret, 2004). However, we found that it was slightly biased towards rating contextually diverse discourse function words (e.g., *also*) as highly polysemous, which needs to be taken into account when interpreting our results. We opted to use this measure, despite this bias, because it has the strong benefit of being clearly interpretable: it simply measures the extent to which a word appears in diverse textual contexts. Table 6 gives examples of the least and most polysemous words in the ENGFIC data, according to this score.

As expected, this measure has significant intrinsic positive correlation with frequency. Across datasets, we found Pearson correlations in the range $0.45 < r < 0.8$ (all $p < 0.05$), confirming frequent words tend to be used in a greater diversity of contexts. As a consequence of this high correlation, we interpret the effect of this measure only after controlling for frequency (this control is naturally captured in equation (7)).

[9] The COHA data is $\sim$100$\times$ smaller, which has a global effect on the construction of the co-occurrence network (e.g., lower average degree) used to compute polysemy scores.

Polysemy and semantic change

After fitting the model in equation (7), we found that the logarithm of the polysemy score exhibits a strong positive effect on rates of semantic change, throughout all four languages (Figure 3b). As with frequency, the relation takes the form of a power law

$$\Delta(w_i) \propto d(w_i)^{\beta_d}, \tag{11}$$

with a language/corpus dependent scaling constant in $\beta_d \in [0.37, 0.77]$. Note that this relationship is a complete reversal from what one would expect according to $d(w_i)$'s positive correlation with frequency; i.e., since frequency and polysemy are highly positively correlated, one would expect them to have similar effects on semantic change, but we found that the effect of polysemy completely reversed after controlling for frequency. Figure 2b shows the relationship of polysemy with rates of semantic change in the ENGALL data after regressing out effect of frequency (using the method of Graham, 2003).

5 Discussion

We show how distributional methods can reveal statistical laws of semantic change and offer a robust methodology for future work in this area.

Our work builds upon a wealth of previous research on quantitative approaches to semantic change, including prior work with distributional methods (Sagi et al., 2011; Wijaya and Yeniterzi, 2011; Gulordava and Baroni, 2011; Jatowt and Duh, 2014; Kulkarni et al., 2014; Xu and Kemp, 2015), as well as recent work on detecting the emergence of novel word senses (Lau et al., 2012; Mitra et al., 2014; Cook et al., 2014; Mitra et al., 2015; Frermann and Lapata, 2016). We extend these lines of work by rigorously comparing different approaches to quantifying semantic change and by using these methods to propose new statistical laws of semantic change.

The two statistical laws we propose have strong implications for future work in historical semantics. The *law of conformity*—frequent words change more slowly—clarifies frequency's role in semantic change. Future studies of semantic change must account for frequency's conforming effect: when examining the interaction between some linguistic process and semantic change, the *law of conformity* should serve as a null model in which the interaction is driven primarily by underlying frequency effects.

The *law of innovation*—polysemous words change more quickly—quantifies the central role polysemy plays in semantic change, an issue that has concerned linguists for more than 100 years (Bréal, 1897). Previous works argued that semantic change leads to polysemy (Wilkins, 1993; Hopper and Traugott, 2003). However, our results show that polysemous words change faster, which suggests that polysemy may actually lead to semantic change.

Overall, these two factors—frequency and polysemy—explain between 48% and 88% of the variance[10] in rates of semantic change (across conditions). This remarkable degree of explanatory power indicates that frequency and polysemy are perhaps the two most crucial linguistic factors that explain rates of semantic change over time.

These empirical statistical laws also lend themselves to various causal mechanisms. The *law of conformity* might be a consequence of learning: perhaps people are more likely to use rare words mistakenly in novel ways, a mechanism formalizable by Bayesian models of word learning and corresponding to the biological notion of genetic drift (Reali and Griffiths, 2010). Or perhaps a sociocultural conformity bias makes people less likely to accept novel innovations of common words, a mechanism analogous to the biological process of purifying selection (Boyd and Richerson, 1988; Pagel et al., 2007). Moreover, such mechanisms may also be partially responsible for the *law of innovation*. Highly polysemous words tend to have more rare senses (Kilgarriff, 2004), and rare senses may be unstable by the *law of conformity*. While our results cannot confirm such causal links, they nonetheless highlight a new role for frequency and polysemy in language change and the importance of distributional models in historical research.

Acknowledgments

The authors thank D. Friedman, R. Sosic, C. Manning, V. Prabhakaran, and S. Todd for their helpful comments and discussions. We are also indebted to our anonymous reviewers. W.H. was supported by an NSERC PGS-D grant and the SAP Stanford Graduate Fellowship. W.H., D.J., and J.L. were supported by the Stanford Data Science Initiative, and NSF Awards IIS-1514268, IIS-1149837, and IIS-1159679.

[10] Marginal R^2 (Nakagawa and Schielzeth, 2013).

References

James S. Adelman, Gordon D. A. Brown, and José F. Quesada. 2006. Contextual diversity, not word frequency, determines word-naming and lexical decision times. *Psychol. Sci.*, 17(9):814–823.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Andreas Blank. 1999. Why do new meanings occur? A cognitive typology of the motivations for lexical semantic change. In Peter Koch and Andreas Blank, editors, *Historical Semantics and Cognition*. Walter de Gruyter, Berlin, Germany.

Robert Boyd and Peter J Richerson. 1988. *Culture and the Evolutionary Process*. University of Chicago Press, Chicago, IL.

Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran. 2012. Distributional semantics in technicolor. In *Proc. ACL*, pages 136–145.

Michel Bréal. 1897. *Essai de Sémantique: Science des significations*. Hachette, Paris, France.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behav. Res. Methods*, 39(3):510–526.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: stop-lists, stemming, and SVD. *Behav. Res. Methods*, 44(3):890–907.

J.L. Bybee. 2007. *Frequency of Use And the Organization of Language*. Oxford University Press, New York City, NY.

Paul Cook, Jey Han Lau, Diana McCarthy, and Timothy Baldwin. 2014. Novel Word-sense Identification. In *Proc. COLING*, pages 1624–1635.

Scott Crossley, Tom Salsbury, and Danielle McNamara. 2010. The development of polysemy and frequency use in english second language speakers. *Language Learning*, 60(3):573–605.

Mark Davies. 2010. The Corpus of Historical American English: 400 million words, 1810-2009. http://corpus.byu.edu/coha/.

Beate Dorow and Dominic Widdows. 2003. Discovering corpus-specific word senses. In *Proc. EACL*, pages 79–82.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Olivier Ferret. 2004. Discovering word senses from a network of lexical cooccurrences. In *Proc. COLING*, page 1326.

J.R. Firth. 1957. A Synopsis of Linguistic Theory, 1930-1955. In *Studies in Linguistic Analysis. Special volume of the Philological Society*. Basil Blackwell, Oxford, UK.

Lea Frermann and Mirella Lapata. 2016. A Bayesian Model of Diachronic Meaning Change. *Trans. ACL*, 4:31–45.

Dirk Geeraerts. 1997. *Diachronic Prototype Semantics: A Contribution to Historical Lexicology*. Clarendon Press, Oxford, UK.

Michael H. Graham. 2003. Confronting multicollinearity in ecological multiple regression. *Ecology*, 84(11):2809–2815.

Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proc. GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*, pages 67–71. Association for Computational Linguistics.

Zellig S. Harris. 1954. Distributional structure. *Word*, 10:146–162.

Paul J. Hopper and Elizabeth Closs Traugott. 2003. *Grammaticalization*. Cambridge University Press, Cambridge, UK.

Adam Jatowt and Kevin Duh. 2014. A framework for analyzing semantic change of words across time. In *Proc. ACM/IEEE-CS Conf. on Digital Libraries*, pages 229–238. IEEE Press.

R. Jeffers and Ilse Lehiste. 1979. *Principles and Methods for Historical Linguistics*. MIT Press, Cambridge, MA.

Adam Kilgarriff. 2004. How dominant is the commonest sense of a word? In *Text, Speech and Dialogue*, pages 103–111. Springer.

Yoon Kim, Yi-I. Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *arXiv preprint arXiv:1405.3515*.

Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2014. Statistically significant detection of linguistic change. In *Proc. WWW*, pages 625–635.

Thomas K. Landauer and Susan T. Dumais. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychol. Rev.*, 104(2):211.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proc. EACL*, pages 591–601.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Trans. ACL*, 3.

Qi Li and Jeffrey Scott Racine. 2007. *Nonparametric econometrics: theory and practice*. Princeton University Press, Princeton, NJ.

Erez Lieberman, Jean-Baptiste Michel, Joe Jackson, Tina Tang, and Martin A. Nowak. 2007. Quantifying the evolutionary dynamics of language. *Nature*, 449(7163):713–716.

Yuri Lin, Jean-Baptiste Michel, Erez Lieberman Aiden, Jon Orwant, Will Brockman, and Slav Petrov. 2012. Syntactic annotations for the google books ngram corpus. In *Proc. ACL, System Demonstrations*, pages 169–174.

Charles E McCulloch and John M Neuhaus. 2001. *Generalized linear mixed models*. Wiley-Interscience, Hoboken, NJ.

Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K. Gray, Joseph P. Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, and others. 2011. Quantitative analysis of culture using millions of digitized books. *Science*, 331(6014):176–182.

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

Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. In *Proc. ACL*.

Sunny Mitra, Ritwik Mitra, Suman Kalyan Maity, Martin Riedl, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2015. An automatic approach to identify word sense changes in text media across timescales. *Natural Language Engineering*, 21(05):773–798.

Shinichi Nakagawa and Holger Schielzeth. 2013. A general and simple method for obtaining R^2 from generalized linear mixed-effects models. *Methods Ecol. Evol.*, 4(2):133–142.

Mark Pagel, Quentin D. Atkinson, and Andrew Meade. 2007. Frequency of word-use predicts rates of lexical evolution throughout Indo-European history. *Nature*, 449(7163):717–720.

Eitan Adam Pechenick, Christopher M. Danforth, and Peter Sheridan Dodds. 2015. Characterizing the Google Books corpus: Strong limits to inferences of socio-cultural and linguistic evolution. *PLoS ONE*, 10(10).

F. Reali and T. L. Griffiths. 2010. Words as alleles: connecting language evolution with Bayesian learners to models of genetic drift. *Proc. R. Soc. B*, 277(1680):429–436.

Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2011. Tracing semantic change with latent semantic analysis. In Kathryn Allan and Justyna A. Robinson, editors, *Current Methods in Historical Semantics*, page 161. De Gruyter Mouton, Berlin, Germany.

Benôit Sagot, Lionel Clément, Eric de La Clergerie, and Pierre Boullier. 2006. The Lefff 2 syntactic lexicon for French: architecture, acquisition, use. In *Proc. LREC*, pages 1–4.

Gerold Schneider and Martin Volk. 1998. Adding manual constraints and lexical look-up to a Brill-tagger for German. In *Proceedings of the ESSLLI-98 Workshop on Recent Advances in Corpus Annotation, Saarbrücken*.

Peter H Schönemann. 1966. A generalized solution of the orthogonal Procrustes problem. *Psychometrika*, 31(1):1–10.

J.S. Seabold and J. Perktold. 2010. Statsmodels: Econometric and statistical modeling with python. In *Proc. 9th Python in Science Conference*.

John Andrew Simpson, Edmund SC Weiner, et al. 1989. *The Oxford English Dictionary*, volume 2. Clarendon Press Oxford, Oxford, UK.

Elizabeth Closs Traugott and Richard B Dasher. 2001. *Regularity in Semantic Change*. Cambridge University Press, Cambridge, UK.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res.*, 37(1):141–188.

S. Ullmann. 1962. *Semantics: An Introduction to the Science of Meaning*. Barnes & Noble, New York City, NY.

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

Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world'networks. *Nature*, 393(6684):440–442.

Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proc. Workshop on Detecting and Exploiting Cultural Diversity on the Social Web*, pages 35–40. ACM.

David P Wilkins. 1993. *From part to person: Natural tendencies of semantic change and the search for cognates*. Cognitive Anthropology Research Group at the Max Planck Institute for Psycholinguistics.

B. Winter, Graham Thompson, and Matthias Urban. 2014. Cognitive Factors Motivating The Evolution Of Word Meanings: Evidence From Corpora, Behavioral Data And Encyclopedic Network Structure. In *Proc. EVOLANG*, pages 353–360.

Yang Xu and Charles Kemp. 2015. A computational evaluation of two laws of semantic change. In *Proc. Annual Conf. of the Cognitive Science Society*.

Yang Xu, Terry Regier, and Barbara C. Malt. 2015. Historical Semantic Chaining and Efficient Communication: The Case of Container Names. *Cognitive Science*.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Marta Palmer. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural language engineering*, 11(02):207–238.

George Kingsley Zipf. 1945. The meaning-frequency relationship of words. *J. Gen. Psychol.*, 33(2):251–256.

Bahar İlgen and Bahar Karaoglan. 2007. Investigation of Zipf's 'law-of-meaning'on Turkish corpora. In *International Symposium on Computer and Information Sciences*, pages 1–6. IEEE.

A Hyperparameter and pre-processing details

For all datasets, words were lowercased and stripped of punctuation. For the Google datasets we built models using the top-100000 words by their average frequency over the entire historical time-periods, and we used the top-50000 for COHA. During model learning we also discarded all words within a year that occurred below a certain threshold (500 for the Google data, 100 for the COHA data).

For all methods, we used the hyperparameters recommended in Levy et al. (2015). For the context word distributions in all methods, we used context distribution smoothing with a smoothing parameter of 0.75. Note that for SGNS this corresponds to smoothing the unigram negative sampling distribution. For both, SGNS and PPMI, we set the negative sample prior $\alpha = \log(5)$, while we set this value to $\alpha = 0$ for SVD, as this improved results. When using SGNS on the Google data, we also subsampled, with words being random removed with probability $p_r(w_i) = 1 - \sqrt{\frac{10^{-5}}{f(w_i)}}$, as recommended by Levy et al. (2015) and Mikolov et al. (2013). Furthermore, to improve the computational efficiency of SGNS (which works with text streams and not co-occurrence counts), we downsampled the larger years in the Google N-Gram data to have at most 10^9 tokens. No such subsampling was performed on the COHA data.

For all methods, we defined the context set to simply be the same vocabulary as the target words, as is standard in most word vector applications (Levy et al., 2015). However, we found that the PPMI method benefited substantially from larger contexts (similar results were found in Bullinaria and Levy, 2007), so we did not remove any low-frequency words per year from the context for that method. The other embedding approaches did not appear to benefit from the inclusion of these low-frequency terms, so they were dropped for computational efficiency.

For SGNS, we used the implementation provided in Levy et al. (2015). The implementations for PPMI and SVD are released with the code package associated with this work.

B Visualization algorithm

To visualize semantic change for a word w_i in two dimensions we employed the following procedure, which relies on the t-SNE embedding method (Van der Maaten and Hinton, 2008) as a subroutine:

1. Find the union of the word w_i's k nearest neighbors over all necessary time-points.

2. Compute the t-SNE embedding of these words on the most recent (i.e., the modern) time-point.

3. For each of the previous time-points, hold all embeddings fixed, except for the target word's (i.e., the embedding for w_i), and optimize a new t-SNE embedding only for the target word. We found that initializing the embedding for the target word to be the centroid of its k'-nearest neighbors in a time-point was highly effective.

Thus, in this procedure the background words are always shown in their "modern" positions, which makes sense given that these are the current meanings of these words. This approximation is necessary, since in reality all words are moving.

C Regression analysis details

In addition to the pre-processing mentioned in the main text, we also normalized the contextual diversity scores $d(w_i)$ within years by subtracting the yearly median. This was necessary because there was substantial changes in the median contextual diversity scores over years due to changes in corpus sample sizes etc. Data points corresponding to words that occurred less than 500 times during a time-period were also discarded, as

these points lack sufficient data to robustly estimate change rates (this threshold only came into effect on the COHA data, however). We removed stop words and proper nouns by (i) removing all stop-words from the available lists in Python's NLTK package (Bird et al., 2009) and (ii) restricting our analysis to words with part-of-speech (POS) tags corresponding to four main linguistic categories (common nouns, verbs, adverbs, and adjectives), using the POS sources in Table 1.

When analyzing the effects of frequency and contextual diversity, the model contained fixed effects for these features and for time along with random effects for word identity. We opted not to control for POS tags in the presented results, as contextual diversity is co-linear with these tags (e.g., adverbs are more contextual diverse than nouns), and the goal was to demonstrate the main effect of contextual diversity across all word types. That said, the effect of contextual diversity remained strong and significantly positive in all datasets even after controlling for POS tags.

To fit the linear mixed models, we used the Python `statsmodels` package with restricted maximum likelihood estimation (REML) (Seabold and Perktold, 2010). All mentioned significance scores were computed according to Wald's z-tests, though these results agreed with Bonferroni corrected likelihood ratio tests on the `eng-all` data.

The visualizations in Figure 2 were computed on the `eng-all` data and correspond to bootstrapped locally-linear kernel regressions with bandwidths selected via the AIC Hurvitch criteria (Li and Racine, 2007).

Beyond Plain Spatial Knowledge: Determining Where Entities Are and Are Not Located, and For How Long

Alakananda Vempala and **Eduardo Blanco**
Human Intelligence and Language Technologies Lab
University of North Texas
Denton, TX, 76203
AlakanandaVempala@my.unt.edu, eduardo.blanco@unt.edu

Abstract

This paper complements semantic role representations with spatial knowledge beyond indicating plain locations. Namely, we extract where entities are (and are not) located, and for how long (seconds, hours, days, etc.). Crowdsourced annotations show that this additional knowledge is intuitive to humans and can be annotated by non-experts. Experimental results show that the task can be automated.

1 Introduction

Extracting meaning from text is crucial for true text understanding and an important component of several natural language processing systems. Among many others, previous efforts have focused on extracting causal relations (Bethard and Martin, 2008), semantic relations between nominals (Hendrickx et al., 2010), spatial relations (Kordjamshidi et al., 2011) and temporal relations (Pustejovsky et al., 2003; Chambers et al., 2014).

In terms of corpora development and automated approaches, semantic roles are one of the most studied semantic representations (Toutanova et al., 2005; Màrquez et al., 2008). They have been proven useful for, among others, coreference resolution (Ponzetto and Strube, 2006) and question answering (Shen and Lapata, 2007). While semantic roles provide a useful semantic layer, they capture a portion of the meaning encoded in all but the simplest statements. Consider the sentence in Figure 1 and the semantic roles of *drove* (solid arrows). In addition to these roles, humans intuitively understand that (dashed arrow) (1) *John* was not located in *Berlin* before or during *drove*, (2) he was located in *Berlin* after *drove* for a short period of time (presumably, until he was done picking up the package, i.e., for a few minutes to an hour), and then left *Berlin* and thus (3) was not located there anymore. Some of this additional spatial knowledge is inherent to the motion verb *drive*: people cannot drive to the location where they are currently located, and they will be located at the destination of driving after *driving* takes place. But determining for how long the agent of *drive* remains at the destination depends on the arguments of *drive*: from *[John]*$_{\text{AGENT}}$ *[drove]*$_{v}$ *[home]*$_{\text{DESTINATION}}$ *[after an exhausting work day]*$_{\text{TIME}}$, it is reasonable to believe that *John* will be located at *home* overnight.

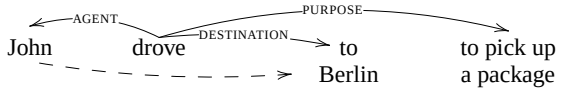


Figure 1: Semantic roles (solid arrows) and additional spatial knowledge (dashed arrow).

This paper manipulates semantic roles in order to extract temporally-anchored spatial knowledge. We extract where entities are and are *not* located, and temporally anchor this information. Temporal anchors indicate for how long something is (or is not) located somewhere, e.g., for 5 minutes before (or after) an event. We target additional spatial knowledge not only between arguments of motion verbs as exemplified above, but also between intra-sentential arguments of any verb. The main contributions are: (1) crowdsourced annotations on top of OntoNotes[1] indicating where something is and is not located (polarity), and for how long (temporal anchors); (2) detailed annotation analysis using coarse- and fine-grained labels (yes / no vs. seconds, minutes, years, etc.); and (3) experiments detailing results with several feature combinations, and using gold-standard and predicted linguistic information.

[1] Available at http://www.cse.unt.edu/~blanco/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1502–1512,
Berlin, Germany, August 7-12, 2016.

2 Definitions and Background

We use R(x, y) to denote a semantic relationship R between x and y. R(x, y) can be read "x has R y", e.g., AGENT(*drove*, *John*) can be read "*drove* has AGENT *John*." By definition, semantic roles are semantic relationships between predicates and their arguments—for all semantic roles R(x, y), x is a predicate and y is an argument of x. Generally speaking, semantic roles capture who did what to whom, how, when and where.

We use the term *additional spatial knowledge* to refer to spatial knowledge not captured with semantic roles, i.e., spatial meaning between x and y where (1) x is not a predicate or (2) x is a predicate and y is not an argument of x. As we shall see, we go beyond extracting "x has LOCATION y" with plain LOCATION(x, y) relations. We extract where entities are and are not located, and for how long they are located (and not located) somewhere.

2.1 Semantic Roles in OntoNotes

OntoNotes (Hovy et al., 2006) is large corpus (≈64K sentences) that includes verbal semantic role annotations, i.e., the first argument x of any role R(x, y) is a verb.[2] OntoNotes semantic roles follow PropBank framesets (Palmer et al., 2005). It uses a set of numbered arguments (ARG_0–ARG_5) whose meanings are verb-dependent, e.g., ARG_2 is used for "*employer*" with verb *work.01* and "*expected terminus of sleep*" with verb *sleep.01*. Additionally, it uses argument modifiers which share a common meaning across verbs (ARGM-LOC, ARGM-TMP, ARGM-PRP, ARGM-CAU, etc.). For a detailed description of OntoNotes semantic roles, we refer the reader to the LDC catalog[3] and PropBank (Palmer et al., 2005). To improve readability, we often rename numbered arguments, e.g., AGENT instead of ARG_0 in Figure 1.

3 Related Work

Approaches to extract PropBank-style semantic roles have been studied for years (Carreras and Màrquez, 2005), state-of-the-art tools sobtain F-measures of 83.5 (Lewis et al., 2015). In this paper, we complement semantic role representations with temporally-anchored spatial knowledge.

Extracting additional meaning on top of popular corpora is by no means a new problem. Gerber and Chai (2010) augmented NomBank (Meyers et al., 2004) annotations with additional numbered arguments appearing in the same or previous sentences, and Laparra and Rigau (2013) presented an improved algorithm for the same task. The SemEval-2010 Task 10 (Ruppenhofer et al., 2009) targeted cross-sentence missing arguments in FrameNet (Baker et al., 1998) and PropBank (Palmer et al., 2005). Silberer and Frank (2012) casted the SemEval task as an anaphora resolution task. We have previously proposed an unsupervised framework to compose semantic relations out of previously extracted relations (Blanco and Moldovan, 2011), and a supervised approach to infer additional argument modifiers (ARGM) for verbs in PropBank (Blanco and Moldovan, 2014). Unlike the current work, these previous efforts (1) improve the semantic representation of verbal and nominal predicates, or (2) infer relations between arguments of the same predicate.

More recently, we showed that spatial relations can be inferred from PropBank-style semantic roles (Blanco and Vempala, 2015; Vempala and Blanco, 2016). In this paper, we expand on this idea as follows. First, we not only extract whether "x has LOCATION y" before, during or after an event, but also specify for how long before and after (seconds, minutes, hours, days, weeks, months, years, etc.). Second, we release crowdsourced annotations for 1,732 potential additional spatial relations. Third, we experiment with both gold and predicted linguistic information.

Spatial semantics has received considerable attention in the last decade.

The task of spatial role labeling (Kordjamshidi et al., 2011; Kolomiyets et al., 2013) aims at representing spatial information with so-called spatial roles, e.g., trajector, landmark, spatial and motion indicators, etc. Unlike us, spatial role labeling does not aim at extracting where entities are *not* located or temporally-anchored spatial information. But doing so is intuitive to humans, as the examples and crowdsourced annotations in this paper show. Spatial knowledge is intuitively associated with motion events, e.g., *drive*, *go*, *fly*, *walk*, *run*. Hwang and Palmer (2015) presented a classifier to detect caused motion constructions triggered by non-motion verbs, e.g., *The crowd laughed the clown off the stage* (i.e., the crowd made the clown leave the stage). Our work does not target motion verbs or motion constructions,

[2] We use the CoNLL 2011 Shared Task release (Pradhan et al., 2011), `http://conll.cemantix.org/2011/`

[3] `https://catalog.ldc.upenn.edu/LDC2013T19`

as the examples in Table 3 show, non-motion constructions triggered by non-motion verbs also allow us to infer temporally-anchored spatial meaning, e.g., *played*, *honored*, *taught*, *fighting*.

4 Corpus Creation and Analysis

Our goal is to complement semantic role representations with additional spatial knowledge. Specifically, our goal is to infer temporally-anchored spatial knowledge between *x* and *y*, where semantic roles ARG$_i$(x_{verb}, *x*) and ARGM-LOC(y_{verb}, *y*) exists in the same sentence. In order to achieve this goal, we follow a two-step methodology. First, we automatically generate potential additional spatial knowledge by combining selected semantic roles. Second, we crowdsource annotations, including polarity and temporal anchors, to validate or discard the potential additional knowledge.

4.1 Generating Potential Spatial Knowledge

We generate potential additional relations LOCATION(*x*, *y*) by combining all ARG$_i$(x_{verb}, *x*) and ARGM-LOC(y_{verb}, *y*) semantic roles within a sentence (x_{verb} and y_{verb} need not be the same). Then, we enforce the following restrictions:

1. *x* and *y* must not overlap;
2. the head of *x* must be a named entity *person*, *org*, *work_of_art*, *fac*, *norp*, *product* or *event*;
3. the head of *y* must be a noun subsumed by *physical_entity.n.01* in WordNet, or a named entity *fac*, *gpe*, *loc*, or *org*;[4] and
4. the heads of *x* and *y* must be different than the heads of all previously generated pairs.

These restrictions were designed after manual analysis of randomly selected combinations of ARG$_i$ and ARGM-LOC semantic roles with two goals in mind: to (1) reduce the annotation effort and (2) generate the least amount of invalid potential additional spatial knowledge without arbitrarily discarding any predicates (e.g., focus only on motion verbs). Additional relations not satisfying restriction 1 are nonsensical, and restriction 4 simply discards potential additional relations that have already been generated. Restrictions 2 and 3 are designed to improve the likelihood that the potential additional spatial knowledge will not be discarded when crowdsourcing annotations, e.g., locations whose head is an adverb such as *here* and *there* (11% of all ARGM-LOC roles) do not yield valid additional spatial knowledge.

OntoNotes annotates 9,612 ARGM-LOC semantic roles, and the number of potential LOCATION relations generated is 1,732. Thus, our methodology aims at adding 18% of additional spatial relations on top of OntoNotes semantic roles. If we consider each temporal anchor as a different spatial relation, we aim at adding 54% additional spatial relations. As we shall see, over 69% of the additional potential relations are valid (Section 4.3).

4.2 Crowdsourcing Spatial Knowledge

Once potential spatial knowledge is generated, it must be validated or discarded. We are interested in additional spatial knowledge as intuitively understood by humans, so we avoid lengthy annotation guidelines and ask simple questions to non-experts via Amazon Mechanical Turk.

After in-house pilot annotations, it became clear that asking "Is *x* located in/at *y*" for each potential LOCATION(*x*, *y*) and forcing annotators to answer *yes* or *no* is suboptimal. For example, consider again Figure 1 and question "Is *John* located in *Berlin*?". An unabridged natural answer would be "not before or during *drove*, but certainly after *drove* for a few minutes until he was done *picking up the package*." In other words, it is intuitive to consider polarity (whether *x* is or is not located at *y*) and temporal anchors (for how long?).

We designed the interface in Figure 2 to gather annotations including polarity and temporal anchors, and accounting for granularity levels. Answers map to the following coarse-grained labels:

- Before and after: `yes`, `no`, `unk` and `inv`.
- During: `yes` (first 2 options), `no`, `unk` and `inv`.

Label `unk` stands for *unknown* and `inv` for *invalid*. Furthermore, `yes` maps to these fine-grained labels indicating specific periods of time:

- Before and after: an integer and a unit of time (`secs`, `mins`, `hours`, `days`, `weeks`, `months` or `years`)[5], or `inf` for infinity.
- During: `entire` or `some`.

[4] For a description and examples of these named entity types, refer to (Weischedel and Brunstein, 2005).

[5] The interface restricts the range of valid integers, e.g., numbers selectable with `secs` range from 1 to 59.

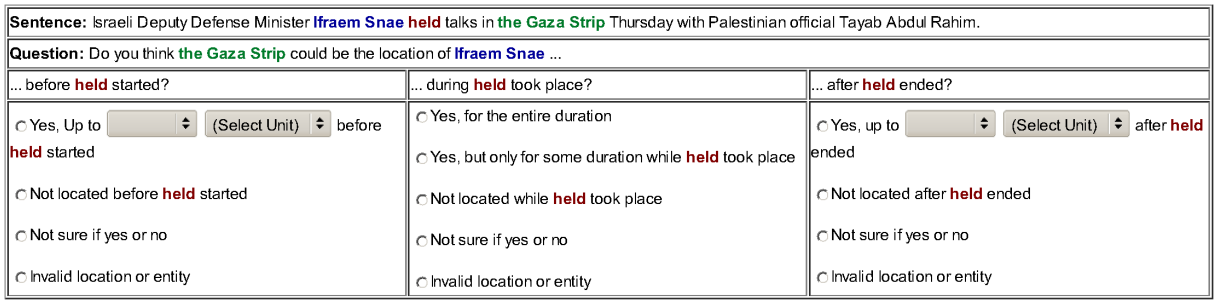


Figure 2: Amazon Mechanical Turk interface to collect temporally-anchored spatial annotations. Annotators were also provided with a description and examples of all answers (not shown).

	secs	mins	hours	days	weeks	months	years	inf	entire	some
Before	0.20	7.55	11.33	7.36	3.78	8.15	46.72	14.91	n/a	n/a
During	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	97.77	2.23
After	0.50	6.48	11.29	6.48	3.34	6.29	29.47	36.15	n/a	n/a

Table 1: Percentage of fine-grained labels for instances annotated with coarse-grained label `yes`.

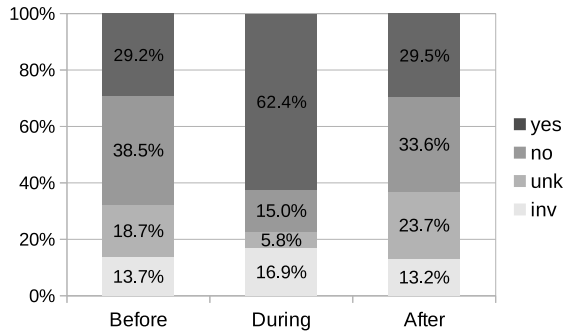


Figure 3: Percentage of coarse-grained labels per temporal anchor. Total number of annotations is $1{,}732 \times 3 = 5{,}196$.

We created one Human Intelligence Task (HIT) per potential LOCATION(x, y), and recruited annotators with previous approval rate $\geq 95\%$ and 5,000 or more previous approved HITs. A total of 74 annotators participated in the task, on average, they annotated 163.24 HITs (maximum: 1,547, minimum: 1). We rejected submissions that took unusually short time compared to other submissions, and those from annotators who always chose the same label. Overall, we only rejected 1.2% of submissions. We collected 7 annotations per HIT and paid $0.05 per HIT.

4.3 Annotation Analysis

Figure 3 shows the percentage of coarse-grained labels per temporal anchor. Labels `yes` and `no` combined account for 67.7% of labels (before), 77.4% (during) and 63.1% (after). Note that both `yes` and `no` yield valid additional spatial knowledge: whether x is (or is not) located at y. Annotators could not commit to `yes` or `no` in 16.1% of questions on average (`unk`), with a much smaller percentage for during temporal anchor (5.8%; before: 18.7%, after: 23.7%). This is not surprising, as arguments of some verbs, e.g., AGENT of *play*, must be located at the location of the event during the event, but not necessarily before or after. Finally, `inv` only accounts for 14.6% of labels (before: 13.7%, during: 16.9%, after: 13.2%), thus most potential additional knowledge automatically generated (Section 4.1) can be understood.

Percentages of fine-grained labels per temporal span, i.e., refinements of `yes` coarse-grained labels, are shown in Table 1. The vast majority of times (97.77%) annotators believe an entity is at a location during an event, the entity is there for the entire duration of the event (`entire`). Annotators barely used label `secs` (before: 0.20% and after: 0.50%), but percentages range between 3.34% and 46.72% for other units of time (uniform distribution would be $1/8 = 12.5\%$). Labels `years` and `inf`, which indicate that an entity is located somewhere for years or indefinitely before (or after) an event, are the most common fine-grained labels for *before* and *after* (14.91–46.72%).

4.3.1 Annotation Quality

Table 2 presents agreement measures. Pearson correlations are the weighted averages between each annotator and the majority label and are calculated following this mapping: (coarse labels): `yes`: 1, `unk`/`inv`: 0, `no`: -1; (fine labels): before/after: `secs`: 1, `mins`: $1 + 1/7$, `hours`:

	Coarse-grained labels						Fine-grained labels					
	Pearson	% instances s.t. a annotators agree					Pearson	% instances s.t. a annotators agree				
		$a=7$	$a\geq 6$	$a\geq 5$	$a\geq 4$	$a\geq 3$		$a=7$	$a\geq 6$	$a\geq 5$	$a\geq 4$	$a\geq 3$
Before	0.73	0.9	8.0	30.0	65.8	97.2	0.67	0.8	6.0	21.6	49.3	85.2
During	0.81	8.9	39.9	59.1	81.4	98.3	0.79	2.1	19.5	45.8	71.8	94.1
After	0.66	0.7	6.0	27.0	62.9	96.6	0.62	0.5	4.6	19.8	49.9	87.1
All	0.67	3.5	18.0	38.8	70.0	97.4	0.64	1.1	10.0	29.0	57.0	88.8

Table 2: Weighed Pearson correlations between annotators and the majority label, and percentage of instances for which at least 7, 6, 5, 4 and 3 annotators (out of 7) agree.

Statement	Before		During		After	
	C	F	C	F	C	F
Statement 1: [...] [Hsia]$_{\text{ARG}_0,\,v_1,v_2}$ [stopped]$_{v_1}$ off [in Milan]$_{\text{ARGM-LOC},v_1}$ [to [visit]$_{v_2}$ [Hsiao Chin]$_{\text{ARG}_1,v_2}$]$_{\text{ARGM-PRP},v_1}$.						
x: *Hsia*, *y*: *Milan*, y_{verb}: *stopped*	yes	mins	yes	entire	yes	hours
x: *Hsiao Chin*, *y*: *Milan*, y_{verb}: *stopped*	yes	years	yes	entire	yes	years
Statement 2: [President Clinton]$_{\text{ARG}_0,v_1}$ [played]$_{v_1}$ [a supporting role]$_{\text{ARG}_1,v_1}$ [today]$_{\text{ARGM-TMP},v_1}$ [in [New York City where]$_{\text{ARGM-LOC},v_2}$ [the first lady, Senator Clinton]$_{\text{ARG}_1,v_2}$, was [honored]$_{v_2}$ [at Madison Square Garden]$_{\text{ARGM-LOC},v_2}$]$_{\text{ARGM-LOC},v_1}$.						
x: *(President) Clinton*, *y*: *New York City*, y_{verb}: *played*	yes	hours	yes	entire	yes	hours
x: *(President) Clinton*, *y*: *Madison Square Garden*, y_{verb}: *honored*	yes	mins	yes	entire	yes	mins
x: *(Senator) Clinton*, *y*: *New York City*, y_{verb}: *played*	yes	hours	yes	entire	yes	hours
x: *(Senator) Clinton*, *y*: *Madison Square Garden*, y_{verb}: *honored*	yes	mins	yes	entire	yes	mins
Statement 3: [Before [joining]$_{v_2}$ [Maidenform]$_{\text{ARG}_1,v_2}$ [in 1972]$_{\text{ARGM-TMP},\,v_2}$]$_{\text{ARGM-TMP},\,v_1}$, [[Mr. Brawer, who]$_{\text{ARG}_0,v_3}$ [holds]$_{v_3}$ [a doctoral degree in English]$_{\text{ARG}_1,v_3}$]$_{\text{ARG}_0,\,v_1,v_2}$, [taught]$_{v_1}$ [at the University of Wisconsin]$_{\text{ARGM-LOC},\,v_1}$.						
x: *Maidenform*, *y*: *University of Wisconsin* , y_{verb}: *taught*	no	n/a	no	n/a	no	n/a
x: *Mr. Brawer*, *y*: *University of Wisconsin*, y_{verb}: *taught*	no	n/a	yes	entire	no	n/a
Statement 4: [...] [George Koskotas, self-confessed embezzler]$_{\text{ARG}_0,\,v_1}$, [now]$_{\text{ARGM-TMP},v_1}$ [residing]$_{v_1}$ [in [a jail cell in Salem, Mass., from where]$_{\text{ARGM-LOC},\,v_2}$ [he]$_{\text{ARG}_0,\,v_2}$ is [fighting]$_{v_2}$ [extradition proceedings]$_{\text{ARG}_1,v_2}$]$_{\text{ARG}_1,v_1}$.						
x: George Koskotas, *y*: *a jail cell in Salem, Mass.*, y_{verb}: *fighting*	yes	months	yes	entire	unk	n/a

Table 3: Annotation examples. For each statement, we indicate semantic roles with square brackets, all potential additional spatial knowledge (is *x* located at *y*?), and annotations with respect to y_{verb} (coarse- (C) and fine-grained (F) labels per temporal anchor: before, during and after).

$1 + 2/7$, `days`: $1 + 3/7$, `weeks`: $1 + 4/7$, `months`: $1 + 5/7$, `years`: $1 + 6/7$, `inf`: 2; during: `some`: 1 `entire`:2. Calculating the weighted average of individual Pearson correlations allows us to take into account the number of questions answered by each annotator.

Correlations range between 0.66 and 0.81 with coarse-grained labels, and are slightly lower with fine-grained labels (0.67 vs. 0.73, 0.79 vs. 0.81, and 0.62 vs. 0.66). Questions for *during* temporal anchor are easier to answer with both kinds of labels (coarse: 0.81, fine: 0.79).

Table 2 also shows how many annotators (out of 7) chose the same label (exact match). At least 4 annotators agreed with coarse-grained labels in most instances (70%), and at least 3 annotators agreed virtually always (97.4%). Percentages are lower with fine-grained labels: 57.0% and 88.8%.

4.4 Annotation Examples

Table 3 presents several annotation examples. We include all potential additional spatial knowledge (Section 4.1) and annotations per temporal anchor.

Two additional LOCATION(*x*, *y*) can be inferred from Statement (1): whether *Hsia* and *Hisao Chin* are located in *Milan* before, during and after *stopped.* Annotators understood that *Hsia* was in *Milan* temporarily: for a few minutes before *stopped*, during the full duration of *stopped* and for a few hours after *stopped.* In other words, *Hsia* was elsewhere, then went to *Milan* and left after visiting with *Hsiao* for a few hours. Regarding *Hsiao*, annotators interpreted that *Milan* is her permanent location: for years before and after *Hsia stopped* to visit her. While somehow ambiguous, these annotations are reasonably intuitive.

Statement (2) has 2 ARGM-LOC roles and 4 potential additional relations. Annotations for *during* are straightforward: both *President Clinton* and *Senator Clinton* were located in *New York City* during *played* and at *Madison Square Garden* during *honored.* Annotations for *before* and *after* are more challenging: both Clintons where located in *New York City* for hours (not days) before and after *played*, but at *Madison Square Garden* for a few minutes (not hours) before and after *honored.*

Feature		Description
basic	1–4	x_{verb}, y_{verb} and their part-of-speech tags
lexical	5–12	first and last words of *x* and *y*, and their part-of-speech tags
	13	whether *x* occurs before or after *y*
heads	14–17	heads of *x* and *y*, and their part-of-speech tags
	18–19	named entity types of the heads of *x* and *y*
semantic	20	semantic role label linking x_{verb} and *x*
	21–24	number of ARGM-TMP and ARGM-LOC roles in x_{roles} and y_{roles}
	25–26	number of ARGM-TMP and ARGM-LOC roles in the sentence to which *x* and *y* belong
	27	whether x_{verb} and y_{verb} are the same verb

Table 4: Feature set to determine whether *x* is (or is not) located at *y*, and for how long. x_{verb} (y_{verb}) denote the verbs to which *x* (y) attach, and x_{roles} (y_{roles}) denote the semantic roles of x_{verb} (y_{verb}).

In other words, they arrived to *Madison Square Garden* shortly before *honored* and left shortly after, but stayed in *New York City* for some hours.

Statement (3) exemplifies `no` label. Potential additional spatial knowledge includes whether *Maidenform* is located at *University of Wisconsin*, which is never true (`no`). Additionally, *University of Wisconsin* was a location of *Mr. Brawer* while he *taught* there (*during*), but nor *before* or *after*.

Statement (4) exemplifies contrastive coarse-grained labels and `unk` label. Annotators interpreted that *George Koskotas* was in the *jail cell* for months before and during *fighting extradition*, and that it is unknown (`unk`) *after fighting* because the outcome of the fight is unknown.

5 Inferring Temporally-Anchored Spatial Knowledge

We follow a standard machine learning approach, and use the training, development and test sets released by the organizers of the CoNLL-2011 Shared Task (Pradhan et al., 2011). We first generate additional spatial knowledge deterministically as described in Section 4.1. Then, for each additional LOCATION(*x*, *y*), we generate one instance per temporal anchor and discard those annotated `inv`. The total number of instances is $1{,}732 \times 3 - 754 = 4{,}442$. We trained SVM models with RBF kernel using scikit-learn (Pedregosa et al., 2011). The feature set and SVM parameters were tuned using 10-fold cross-validation with the train and development sets, and results are calculated using the test set. During the tuning process, we discovered that it is beneficial to train one SVM per temporal anchor instead of a single model for the 3 temporal anchors.

5.1 Feature Selection

We use a mixture of standard features from semantic role labeling, and semantic features designed for extracting temporally-anchored spatial knowledge from semantic roles. In order to determine whether *x* is (or is not) located at *y* and for how long, we extract features from *x* and *y*, the verbs to which they attach (x_{verb} and y_{verb}) and all semantic roles of x_{verb} and y_{verb} (x_{roles} and y_{roles}).

Basic, *lexical* and *heads* features are standard in role labeling (Gildea and Jurafsky, 2002). *Basic* features are the word form and part-of-speech of x_{verb} and y_{verb}. *Lexical* features capture the first and last words of *x* and *y* and their part-of-speech tags, as well as a binary flag indicating whether *x* occurs before or after *y*. *Heads* features capture the heads of *x* and *y* and their part-of-speech tags, as well as their named entity types, if any.

Semantic features include features 20–27. Feature 20 indicates the semantic role linking *x* and x_{verb} (ARG_0, ARG_1, ARG_2, etc.); recall that the semantic role between *y* and y_{verb} is always ARGM-LOC (Section 4.1). Features 21–24 are counts of ARGM-TMP and ARGM-LOC semantic roles in the verb-argument structures to which *x* and *y* attach. Features 25–26 are the same counts of roles, but taking into account all the roles in the sentence to which *x* and *y* belong. Finally, feature 27 signals whether *x* and *y* attach to the same verb.

We tried many other features, including counts of all roles, heads of all semantic roles present, semantic role ordering, VerbNet (Schuler, 2005) and Levin (Levin, 1993) verb classes, and WordNet hypernyms (Miller, 1995), but they did not yield any improvements during the tuning process.

We exemplify features with pair (*x*: *George Koskotas, self-confessed embezzler*, *y*: *a jail cell in [...], from where*) from Statement 4 in Table 3:

- Basic: features 1–4: {residing, VBG, fighting, VBG}.
- Lexical: feature 5–12: {George, NNP, Koskotas, NNP, a, DT, where, WRB}, features 13: {before}.

		Before			During			After			All		
		P	R	F	P	R	F	P	R	F	P	R	F
baseline	yes	0.00	0.00	0.00	0.77	1.00	0.87	0.00	0.00	0.00	0.77	0.55	0.64
	no	0.49	1.00	0.65	0.00	0.00	0.00	0.40	1.00	0.57	0.44	0.86	0.58
	unk	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Weighted avg.	0.24	0.49	0.32	0.60	0.77	0.67	0.16	0.40	0.23	0.51	0.55	0.50
basic	yes	0.48	0.34	0.40	0.82	0.94	0.88	0.53	0.50	0.52	0.71	0.71	0.71
	no	0.52	0.63	0.57	0.62	0.36	0.46	0.47	0.49	0.48	0.51	0.54	0.52
	unk	0.23	0.21	0.22	0.33	0.10	0.15	0.29	0.29	0.29	0.26	0.23	0.25
	Weighted avg.	0.44	0.45	0.44	0.76	0.79	0.76	0.44	0.44	0.44	0.55	0.56	**0.56**
basic + lexical + heads	yes	0.68	0.37	0.48	0.83	0.92	0.87	0.73	0.38	0.50	0.79	0.67	0.73
	no	0.59	0.80	0.68	0.47	0.32	0.38	0.53	0.66	0.59	0.56	0.68	0.61
	unk	0.36	0.29	0.32	0.20	0.10	0.13	0.32	0.39	0.35	0.33	0.32	0.32
	Weighted avg.	0.56	0.56	0.54	0.73	0.77	0.74	0.54	0.50	0.50	0.62	0.61	**0.61**
basic + lexical + heads + semantics	yes	0.86	0.44	0.58	0.85	0.94	0.90	0.79	0.38	0.51	0.84	0.70	**0.77**
	no	0.63	0.80	0.71	0.56	0.47	0.50	0.55	0.69	0.61	0.59	0.71	**0.64**
	unk	0.37	0.38	0.38	0.50	0.10	0.17	0.33	0.42	0.37	0.35	0.37	**0.36**
	Weighted avg.	0.64	0.60	**0.60**	0.78	0.81	**0.78**	0.57	0.52	**0.52**	0.66	0.64	**0.65**

Table 5: Results obtained with gold-standard linguistic annotations and coarse-grained labels using the baseline and several feature combinations (basic, lexical, heads and semantic features).

- Head: features 14–17: {Koskotas, NNP, cell, NN}, features 18–19: {person, none},
- Semantic feature 20: $\{\text{ARG}_0\}$, features 21–24: {1, 0, 0, 1}, feature 25–26: {1, 1}, feature 27: {no}.

6 Experiments and Results

We present results using gold-standard (Section 6.1) and predicted (Section 6.2) linguistic annotations. POS tags, parse trees, named entities and semantic roles are taken directly from *gold* or *auto* files in the CoNLL-2011 Shared Task release.

6.1 Gold-Standard Linguistic Annotations

Using gold-standard linguistic annotations has two advantages. First, because we have gold semantic roles and named entities, we generate the same potential additional spatial knowledge generated while creating our annotations (Section 4.1). Second, feature values are guaranteed to be correct.

6.1.1 Predicting Coarse-Grained Labels

Table 5 presents results with coarse-grained labels using a baseline and learning with several combinations of features extracted from gold-standard linguistic annotations (POS tags, parse trees, semantic roles, etc.). The baseline predicts the most frequent label per temporal anchor, i.e., yes for *during*, and no for *before* and *after* (Figure 3).

Best results for all labels and temporal anchors are obtained with all features (basic, lexical, heads and semantics). Overall F-measure is 0.65, and *during* instances obtain higher F-measure (0.78) than *before* (0.60) and *after* (0.52). Regarding labels, yes obtains best results (overall 0.77), followed by no (0.64) and unk (0.36). Not surprisingly, the most frequent label per temporal anchor obtains the best results with all features (*before*: no, 0.71; *during*: yes, 0.90; *after*: no, 0.61).

Before and *after* instances benefit the most from learning with all features with respect to the baseline (before: 0.32 vs. 0.60, after: 0.23 vs. 0.52). While *during* instances also benefit, the difference in F-measure is lower (0.67 vs. 0.78).

Feature Ablation. The bottom 3 blocks in Table 5 present results using several feature types incrementally. *Basic* features yield an overall F-measure of 0.56, and surprisingly good results for *during* instances (0.76). Indeed, the best performance obtained with *during* instances is 0.78 (all features), suggesting that the verbs to which *x* and *y* attach are very strong features.

Lexical and *heads* features are most useful for *before* (0.44 vs. 0.54, +22.7%) and *after* (0.44 vs. 0.50, +13.6%) instances, and are actually detrimental for *during* instances (0.76 vs. 0.74, -2.6%). Including *semantic* features, however, improves results with respect to *basic* features for all temporal anchors: before: 0.44 vs. 0.60, 36.4% during: 0.76 vs. 0.78, 2.6% after: 0.44 vs. 0.52, 18.2%.

Differences in overall F-measure are not statistically significant between *basic* and *basic + lexical + heads* (0.56 vs. 0.61, Z-test, two-tailed, p-value $= 0.05$), but the difference including *semantic* features is significant (0.50 vs. 0.65, Z-test, two-tailed, p-value $= 0.009$).

		Before			During			After			All		
		P	R	F	P	R	F	P	R	F	P	R	F
baseline	spurious	0.50	1.00	0.66	0.50	1.00	0.66	0.50	1.00	0.66	0.50	1.00	0.66
	other	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Weighted avg.	0.25	0.50	0.33	0.25	0.50	0.33	0.25	0.50	0.33	0.25	0.50	0.33
basic	yes	0.00	0.00	0.00	0.47	0.56	0.51	0.00	0.00	0.00	0.52	0.36	0.43
	no	0.55	0.33	0.42	0.40	0.20	0.27	0.43	0.38	0.41	0.43	0.32	0.37
	unk	0.26	0.30	0.28	0.11	0.07	0.08	0.37	0.25	0.30	0.24	0.18	0.21
	spurious	0.68	0.91	0.78	0.68	0.71	0.70	0.67	0.93	0.78	0.69	0.88	0.77
	Weighted avg.	0.51	0.58	0.53	0.53	0.55	0.54	0.49	0.58	0.52	0.54	0.58	**0.55**
basic + lexical + heads + semantics	yes	1.00	0.07	0.13	0.74	0.87	0.80	0.00	0.00	0.00	0.74	0.56	**0.64**
	no	0.64	0.48	0.55	0.67	0.20	0.31	0.43	0.46	0.44	0.53	0.49	**0.51**
	unk	0.41	0.78	0.54	0.50	0.47	0.48	0.41	0.61	0.49	0.51	0.68	0.58
	spurious	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Weighted avg.	0.82	0.75	0.73	0.84	0.84	0.83	0.66	0.71	0.68	0.80	0.79	**0.79**

Table 7: Results obtained with predicted linguistic annotations and coarse-grained labels. spurious is a new label indicating overgenerated pairs not present in the gold standard.

	Label	P	R	F
Before	mins	1.00	0.67	0.80
	days	1.00	0.50	0.67
	years	0.15	0.17	0.16
	inf	1.00	0.33	0.50
	no	0.63	0.80	0.71
	unk	0.37	0.38	0.38
	other	0.00	0.00	0.00
	Weighted avg.	0.52	0.54	**0.51**
During	entire	0.84	0.94	0.88
	some	0.00	0.00	0.00
	no	0.56	0.45	0.50
	unk	0.50	0.10	0.17
	Weighted avg.	0.77	0.80	**0.77**
After	years	0.27	0.25	0.26
	inf	0.56	0.24	0.33
	no	0.55	0.69	0.61
	unk	0.33	0.42	0.37
	other	0.00	0.00	0.00
	Weighted avg.	0.41	0.44	**0.41**
All	mins	0.50	0.50	0.50
	days	1.00	0.29	0.44
	years	0.21	0.21	0.21
	inf	0.67	0.27	0.38
	entire	0.84	0.94	0.89
	no	0.59	0.71	0.64
	unk	0.35	0.37	0.36
	other	0.00	0.00	0.00
	Weighted avg.	0.56	0.59	**0.57**

Table 6: Results obtained with gold linguistic annotations and fine-grained labels using all features.

6.1.2 Predicting Fine-Grained Labels

Table 6 presents results using fine-grained labels and all features. Overall F-measure is lower than with coarse-grained labels (0.57 vs. 0.65). Results for *during* instances barely decreases (0.78 vs. 0.77) because almost 98% of fine-grained labels are entire (Table 1).

Most fine-grained labels for *before* and *after* are infrequent (Table 1), our best model is unable to predict labels secs, hours, weeks and months for *before*, and secs, mins, hours, days, weeks and months for *after* (*other* rows). But these labels account for relatively few instances: individually, between 0.2% and 11.33%, and among all of them, 23.46% for *before* and 34.38% for *after* instances.

It is worth noting that mins, days and inf obtain relatively high F-measures for *before*: 0.80, 0.67 and 0.50 respectively. In other words, we can distinguish whether an entity is somewhere only for a few minutes or days (but not longer) before an event, or at all times before an event.

6.2 Predicted Linguistic Annotations

In order to make an honest evaluation in a realistic environment, we also experiment with predicted linguistic annotations. The major disadvantage of doing so is that predicted semantic roles and named entities are often incorrect or missing, thus we generate spurious additional spatial knowledge and miss some additional spatial knowledge because the potential relation cannot be generated.

Table 7 presents results using predicted linguistic annotations. The additional label spurious is used for instances generated from incorrect semantic roles or named entities, as these instances do not appear in the crowdsourced annotations (Section 4). Due to space constraints, we only present results using coarse-grained labels, but provide results per temporal anchor.

The baseline, which predicts the most likely label per temporal anchor, always predicts spurious since 50% of generated additional potential knowledge does not appear in the crowdsourced annotations. Using all features clearly outperforms *basic* features (overall F-measure:

0.79 vs 0.55), thus we focus on the former.

Using all features, `spurious` is always predicted correctly. While useful to discard additional spatial knowledge that should not have been generated, `spurious` does not allow us to make meaningful inferences. The labels that we are most interested in, `yes` and `no`, obtain overall F-measures of 0.64 and 0.51 (compared to 0.77 and .64 with gold linguistic annotations). Regarding labels, `yes` can only be reliably predicted for *during* instances (F-measure: 0.80), and `no` is predicted with modest F-measures for all temporal anchors: before: 0.55, during: 0.31, after: 0.44.

7 Conclusions

This paper demonstrates that semantic roles are a reliable semantic layer from which one can infer whether entities are located or not located somewhere, and for how long (seconds, minutes, days, years, etc.). Crowdsourced annotations show that this kind of inferences are intuitive to humans. Moreover, most potential additional spatial knowledge generated following a few simple deterministic rules was validated by annotators (`yes` and `no`; before: 67.7%, during: 77.4%, after: 63.1%).

Experimental results with gold-standard semantic roles and named entities show that inference can be done with standard supervised machine learning (overall F-measure: 0.65, `yes`: 0.77, `no`: 0.64). Using predicted linguistic information, results decrease substantially (`yes`: 0.64, `no`: 0.51). This is mostly due to the fact that predicted semantic roles and named entities are often wrong or missing, and this fact unequivocally makes the inference process more challenging.

We believe that combining semantic roles and other semantic representation in a similar fashion to the one used in this paper could be useful to infer knowledge beyond spatial inferences. For example, one could infer who is in POSSESSION of something over time by manipulating the events in which the object in question participates in.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational Linguistics*, Montreal, Canada.

Steven Bethard and James H. Martin. 2008. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, HLT-Short '08, pages 177–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eduardo Blanco and Dan Moldovan. 2011. Unsupervised learning of semantic relation composition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 1456–1465, Portland, Oregon.

Eduardo Blanco and Dan Moldovan. 2014. Leveraging verb-argument structures to infer semantic relations. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 145–154, Gothenburg, Sweden, April. Association for Computational Linguistics.

Eduardo Blanco and Alakananda Vempala. 2015. Inferring temporally-anchored spatial knowledge from semantic roles. In *Proceedings of the 2015 Annual Conference of the North American Chapter of the ACL*, pages 452–461.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: semantic role labeling. In *CONLL '05: Proceedings of the Ninth Conference on Computational Natural Language Learning*, pages 152–164, Morristown, NJ, USA. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense event ordering with a multi-pass architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Matthew Gerber and Joyce Chai. 2010. Beyond NomBank: A Study of Implicit Arguments for Nominal Predicates. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1583–1592, Uppsala, Sweden, July. Association for Computational Linguistics.

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

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Sebastian Ó Séaghdha, Diarmuid andPadó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38, Uppsala, Sweden, July.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. OntoNotes: the 90% Solution. In *NAACL '06: Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers on XX*, pages 57–60, Morristown, NJ, USA. Association for Computational Linguistics.

Jena D. Hwang and Martha Palmer. 2015. Identification of caused motion construction. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 51–60, Denver, Colorado, June. Association for Computational Linguistics.

Oleksandr Kolomiyets, Parisa Kordjamshidi, Marie-Francine Moens, and Steven Bethard. 2013. Semeval-2013 task 3: Spatial role labeling. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 255–262. Association for Computational Linguistics.

Parisa Kordjamshidi, Martijn Van Otterlo, and Marie-Francine Moens. 2011. Spatial role labeling: Towards extraction of spatial relations from natural language. *ACM Trans. Speech Lang. Process.*, 8(3):4:1–4:36, December.

Egoitz Laparra and German Rigau. 2013. ImpAr: A Deterministic Algorithm for Implicit Semantic Role Labelling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1180–1189, Sofia, Bulgaria, August. Association for Computational Linguistics.

Beth Levin. 1993. *English verb classes and alternations : a preliminary investigation*.

Mike Lewis, Luheng He, and Luke Zettlemoyer. 2015. Joint a* ccg parsing and semantic role labelling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1444–1454, Lisbon, Portugal, September. Association for Computational Linguistics.

Lluís Màrquez, Xavier Carreras, Kenneth C Litkowski, and Suzanne Stevenson. 2008. Semantic role labeling: an introduction to the special issue. *Computational linguistics*, 34(2):145–159.

Adam Meyers, Ruth Reeves, Catherine Macleod, Rachel Szekely, Veronika Zielinska, Brian Young, and Ralph Grishman. 2004. Annotating noun argument structure for nombank. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC-2004)*.

George A. Miller. 1995. Wordnet: A lexical database for english. In *Communications of the ACM*, volume 38, pages 39–41.

Martha Palmer, Daniel Gildea, and Paul Kingsbury. 2005. The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Simone Paolo Ponzetto and Michael Strube. 2006. Exploiting semantic role labeling, wordnet and wikipedia for coreference resolution. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 192–199, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sameer Pradhan, Lance Ramshaw, Mitchell Marcus, Martha Palmer, Ralph Weischedel, and Nianwen Xue. 2011. Conll-2011 shared task: Modeling unrestricted coreference in ontonotes. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–27, Portland, Oregon, USA, June. Association for Computational Linguistics.

James Pustejovsky, Patrick Hanks, Roser Sauri, Andrew See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, et al. 2003. The timebank corpus. In *Corpus linguistics*, volume 2003, page 40.

Josef Ruppenhofer, Caroline Sporleder, Roser Morante, Collin Baker, and Martha Palmer. 2009. SemEval-2010 Task 10: Linking Events and Their Participants in Discourse. In *Proceedings of the Workshop on Semantic Evaluations: Recent Achievements and Future Directions (SEW-2009)*, pages 106–111, Boulder, Colorado, June. Association for Computational Linguistics.

Karin Kipper Schuler. 2005. *Verbnet: A Broad-coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, Philadelphia, PA, USA. AAI3179808.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 12–21, Prague, Czech Republic, June. Association for Computational Linguistics.

Carina Silberer and Anette Frank. 2012. Casting implicit role linking as an anaphora resolution task. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, SemEval '12, pages 1–10, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kristina Toutanova, Aria Haghighi, and Christopher D Manning. 2005. Joint learning improves semantic role labeling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 589–596. Association for Computational Linguistics.

Alakananda Vempala and Eduardo Blanco. 2016. Complementing semantic roles with temporally an-

chored spatial knowledge: Crowdsourced annotations and experiments. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2652–2658.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Technical report, Linguistic Data Consortium, Philadelphia.

LEXSEMTM: A Semantic Dataset Based on All-words Unsupervised Sense Distribution Learning

Andrew Bennett,[♡] **Timothy Baldwin,**[♡] **Jey Han Lau,**[♡♢]
Diana McCarthy,[♣] and **Francis Bond**[♠]
[♡] Dept of Computing and Information Systems, The University of Melbourne
[♢] IBM Research
[♣] Dept of Theoretical and Applied Linguistics, University of Cambridge
[♠] Linguistics and Multilingual Studies, Nanyang Technological University
awbennett0@gmail.com, tb@ldwin.net, jeyhan.lau@gmail.com,
diana@dianamccarthy.co.uk, bond@ieee.org

Abstract

There has recently been a lot of interest in unsupervised methods for learning sense distributions, particularly in applications where sense distinctions are needed. This paper analyses a state-of-the-art method for sense distribution learning, and optimises it for application to the entire vocabulary of a given language. The optimised method is then used to produce LEXSEMTM: a sense frequency and semantic dataset of unprecedented size, spanning approximately 88% of polysemous, English simplex lemmas, which is released as a public resource to the community. Finally, the quality of this data is investigated, and the LEXSEMTM sense distributions are shown to be superior to those based on the WORDNET first sense for lemmas missing from SEMCOR, and at least on par with SEMCOR-based distributions otherwise.

1 Introduction

Word sense disambiguation (WSD), as well as more general problems involving word senses, have been of great interest to the NLP community for many years (for a detailed overview, see Agirre and Edmonds (2007) and Navigli (2009)). In particular, there has recently been a lot of work on unsupervised techniques for these problems. This includes unsupervised methods for performing WSD (Postma et al., 2015; Chen et al., 2014; Boyd-Graber et al., 2007; Brody et al., 2006), as well as complementary problems dealing with word senses (Jin et al., 2009; Lau et al., 2014).

One such application has been the automatic learning of sense distributions (McCarthy et al., 2004b; Lau et al., 2014). A sense distribution is a probability distribution over the senses of a given lemma. For example, if the noun *crane* had two senses, `bird` and `machine`, then a hypothetical sense distribution could indicate that the noun is expected to take the `machine` meaning 60% of the time and the `bird` meaning 40% of the time in a representative corpus. Sense distributions (or simple "first sense" information) are used widely in tasks including information extraction (Tandon et al., 2015), novel word sense detection (Lau et al., 2012; Lau et al., 2014), semi-automatic dictionary construction (Cook et al., 2013), lexical simplification (Biran et al., 2011), and textual entailment (Shnarch et al., 2011). Automatically acquired sense distributions themselves are also used to improve unsupervised WSD, for example by providing a most frequent sense heuristic (McCarthy et al., 2004b; Jin et al., 2009) or by improving unsupervised usage sampling strategies (Agirre and Martinez, 2004). Furthermore, the improvement due to the most frequent sense heuristic has been particularly strong when used with domain-specific data (Koeling et al., 2005; Chan and Ng, 2006; Lau et al., 2014).

In addition, there is great scope to use these techniques to improve existing sense frequency resources, which are currently limited by the bottleneck of requiring manual sense annotation. The most prominent example of such a resource is WORDNET (Fellbaum, 1998), where the sense

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1513–1524,
Berlin, Germany, August 7-12, 2016.

frequency data is based on SEMCOR (Miller et al., 1993), a 220,000 word corpus that has been manually tagged with WORDNET senses. This data is full of glaring irregularities due to its age and the limited size of the corpus; for example, the word *pipe* has its most frequent sense listed as `tobacco pipe`, whereas one might expect this to be `tube carrying water or gas` in modern English (McCarthy et al., 2004a). This is likely due to the more common use of the `tobacco pipe` sense in mid-20th century literature. The problem is particularly highlighted by the fact that out of the approximately 28,000 polysemous simplex lemmas in WORDNET 3.0, approximately 61% have no sense annotations at all, and less than half of the remaining lemmas have at least 5 sense annotations!

Unfortunately, there has been a lack of work investigating how to apply sense learning techniques at the scale of a full lexical resource such as WORDNET. Updating language-wide sense frequency resources would require learning sense distributions over the entire vocabularies of languages, which could be extremely computationally expensive. To make things worse, domain differences could require learning numerous distributions per word. Despite this, though, we would not want to make these techniques scalable at the expense of sense distribution quality. Therefore, we would like to understand the tradeoff between the accuracy and computation time of these techniques, and optimise this tradeoff. This could be particularly critical in applying them in an industrial setting.

The current state-of-the-art technique for unsupervised sense distribution learning is HDP-WSI (Lau et al., 2014). In order to address the above concerns, we provide a series of investigations exploring how to best optimise HDP-WSI for large-scale application. We then use our optimised technique to produce LEXSEMTM,[1] a semantic and sense frequency dataset of unprecedented size, spanning the entire vocabulary of English. Finally, we use crowdsourced data to produce a new set of gold-standard sense distributions to accompany LEXSEMTM. We use these to investigate the quality of the sense frequency data in LEXSEMTM with respect to SEMCOR.

[1] LEXSEMTM, as well as code for accessing LEXSEMTM and reproducing our experiments is available via: `https://github.com/awbennett/LexSemTm`

2 Background and Related Work

Given the difficulty and expense of obtaining large-scale and robust annotated data, unsupervised approaches to problems involving word learning and recognising word senses have long been studied in NLP. Perhaps the most famous such problem is word sense disambiguation (WSD), for which many unsupervised solutions have been proposed. Some methods are very complex, performing WSD separately for each word usage using information such as word embeddings of surrounding words (Chen et al., 2014) or POS-tags (Lapata and Brew, 2004). On the other hand, most approaches make use of the difficult-to-beat most frequent sense (MFS) heuristic (McCarthy et al., 2007), which assigns each usage of a given word-type to its most frequent sense.

Given the popularity of the MFS heuristic, much of the past work on unsupervised techniques has focused on identifying the most frequent sense. The original method of this kind was proposed by McCarthy et al. (2004b), which relied on finding distributionally similar words to the target word, and comparing these to the candidate senses. Most subsequent approaches have followed a similar approach, based on the words appearing nearby the target word across its token usages. Boyd-Graber and Blei (2007) formalise the method of McCarthy et al. (2004b) with a probabilistic model, while others take different approaches, such as adapting existing sense frequency data to specific domains (Chan and Ng, 2005; Chan and Ng, 2006), using coarse grained thesaurus-like sense inventories (Mohammad and Hirst, 2006), adapting information retrieval-based methods (Lapata and Keller, 2007), using ensemble learning (Brody et al., 2006), utilising the network structure of WORDNET (Boyd-Graber et al., 2007), or making use of word embeddings (Bhingardive et al., 2015). Alternatively, Jin et al. (2009) focus on how best to use the MFS heuristic, by identifying when best to apply it, based on sense distribution entropy. Perhaps the most promising approach is that of Lau et al. (2014), due to its state-of-the art performance, and the fact that it can easily by applied to any language and any sense repository containing sense glosses.

The task we are interested in — namely, sense distribution learning — is in principle very similar to identifying the MFS. Indeed, of these methods for identifying the MFS, some of them are

explicitly described in terms of sense distribution learning (Chan and Ng, 2005; Chan and Ng, 2006; Lau et al., 2014), while the others implicitly learn sense distributions by calculating some kind of scores used to rank senses.

The state-of-the-art technique of Lau et al. (2014) that we are building upon involves performing unsupervised word sense induction (WSI), which itself is implemented using nonparametric HDP (Teh et al., 2006) topic models, as detailed in Section 3. The WSI component, HDP-WSI, is based on the work of Lau et al. (2012), which at the time was state-of-the-art. Since then, however, other competitive WSI approaches have been developed, involving complex structures such as multi-layer topic models (Chang et al., 2014), or complex word embedding based approaches (Neelakantan et al., 2014). We have not used these approaches in this work on account of their complexity and likely computational cost, however we believe they are worth future exploration. On the other hand, because HDP-WSI is implemented using topic models, it can be customised by replacing HDP with newer, more efficient topic modelling algorithms. Recent work has produced more advanced topic modelling approaches, some of which are extensions of existing approaches using more advanced learning algorithms or expanded models (Buntine and Mishra, 2014), while others are more novel, involving variations such as neural networks (Larochelle and Murray, 2011; Cao et al., 2015), or incorporating distributional similarity of words (Xie et al., 2015). Of these approaches, we chose to experiment with that of Buntine and Mishra (2014) because a working implementation was readily available, it has previously shown very strong performance in terms of accuracy and speed, and it is similar to HDP and thus easy to incorporate into our work.

3 HDP-WSI Sense Learning

HDP-WSI (Lau et al., 2014) is a state-of-the-art unsupervised method for learning sense distributions, given a sense repository with per-sense glosses. It takes as input a collection of example usages of the target lemma[2] and the glosses for each target sense, and produces a probability distribution over the target senses.

At the heart of HDP-WSI is HDP (Teh et al., 2006), a nonparametric topic modelling technique. It is a generative probabilistic model and uses topics as a latent variable to allow statistical sharing between documents, providing a kind of soft-clustering mixture model. Each document is assumed to have a corresponding distribution over these topics, and each topic is assumed to have a corresponding distribution over words. According to the model, each word for a given document is independently generated by first sampling a topic according to that document's distribution over topics, and then sampling a word according to the topic's distribution over words. Unlike older topic modelling methods such as LDA (Blei et al., 2003), HDP is nonparametric, meaning the number of topics used by the model is automatically learnt, and does not need to be set as a hyperparameter. In other words, the model automatically learns the "right" number of topics for each lemma.

HDP-WSI follows a two-step process: word sense induction (WSI), followed by topic–sense alignment. WSI is performed using HDP based on the earlier work of Lau et al. (2012): each usage of the target lemma is treated as a document, and HDP topic modelling is run on this document collection. This gives a variable number of learnt topics, which are the senses induced by WSI. A single topic is then assigned to each document,[3] and a distribution over these topics is learnt using maximum likelihood estimation.

In the second step of HDP-WSI, we align the distribution over topics from WSI to the provided sense inventory. We first create a distribution over words for each sense, from the sense's gloss.[4] Then a prevalence score is calculated for each sense by taking a weighted sum of the similarity of that sense with every topic,[5] weighting each similarity score by the topic's probability. These prevalence scores are finally normalised to give a distribution over senses.

Despite state-of-the-art results with HDP-WSI in past work (Lau et al., 2014), there are some con-

[2]Except where stated otherwise, a lemma usage includes the sentence containing the lemma, and the two immediate neighbouring sentences (if available). It is assumed that each usage has been normalised via lemmatisation and stopword removal, and extra local-context tokens are added, as was done by Lau et al. (2012).

[3]The topic with the maximum probability is assigned.

[4]As with the lemma usages, the text is normalised via lemmatisation and stopword removal. Then a distribution is created using maximum likelihood estimation.

[5]Defined in terms of Jensen Shannon divergence between the respective distributions over words.

cerns in applying it to large-scale learning. Most importantly, in order to make HDP nonparametric, it relies on relatively inefficient MCMC sampling techniques, typically based on a hierarchical Chinese Restaurant Process ("CRP"). On the other hand, recent work has provided very efficient topic modelling techniques given a fixed number of topics. While in previous work it was assumed that performance benefits of HDP over other techniques like LDA were based on it learning the "right" number of topics (Lau et al., 2012; Lau et al., 2014), more recent work challenges this assumption. Rather, it is suggested that it is more important for topic modelling to use high-performance learning algorithms so that topics are learnt in correct proportions, in which case "junk" topics can easily be ignored (Buntine and Mishra, 2014). In other words, it is likely that the previously-found performance advantage of HDP over LDA was actually due to properties of their respective Gibbs sampling algorithms.

Furthermore, in our experience using it for sense distribution learning, HDP seems to use a very consistent number of topics. In experiments we ran on the BNC[6] — the same dataset that Lau et al. (2014) based their experiments on — the number of topics was between 5 and 10 over 80% of the time, and over 99% of the time it was below 14. Because the number of topics is so consistent, it is likely we can safely use a fixed number with little risk that it will be too low.

In addition, there are some theoretical concerns with HDP. Firstly, it models topic and word allocations using Dirichlet Processes (Teh et al., 2006). However, previous research has shown that phenomena such as word and sense frequencies follow power-law distributions according to Zipf's law (Piantadosi, 2014), and thus are better modelled using Pitman-Yor Processes (Pitman and Yor, 1997). Another weakness is that HDP does not model burstiness. This is a phenomenon where words that occur at least once in a given discourse are disproportionately more likely to occur several times, even compared with other discourses about the same topic (Church, 2000; Doyle and Elkan, 2009).

[6]The British National Corpus (Burnard, 1995), which is a balanced corpus of English.

4 HCA-WSI Sense Learning

We now present and evaluate HCA-WSI, which is an alternative to HDP-WSI that addresses the above concerns. It follows the same process as HDP-WSI, except that the HDP topic modelling is replaced with HCA[7] (Buntine and Mishra, 2014), a more advanced software suite for topic modelling.[8] HCA is based on a similar probabilistic model to HDP, except for a few differences: (1) it only has a fixed number of topics; (2) it models word frequencies using a more general Pitman-Yor Process; and (3) it incorporates an extra component to the model to model burstiness (each document can individually have an elevated probability for some words, regardless of its distribution over topics). The second and third of these differences directly answer our theoretical concerns about using HDP.

The learning algorithm for HCA is called "table indicator sampling" (Chen et al., 2011), which is a collapsed Gibbs sampling algorithm. The overall probabilistic model is interpreted as a hierarchical CRP, and some extra latent variables called table indicators are added to the model, which encode the decisions made about creating new tables during the CRP. The use of these latent variables allows for a very efficient collapsed Gibbs sampling process, which is found to converge more quickly than competing Gibbs sampling and variational Bayes techniques. The convergence is also shown to be more accurate, with topic models of lower perplexity being produced given the same underlying stochastic model.

Compared to HDP, HCA has been shown to be orders of magnitude faster, with similar memory overhead (Buntine and Mishra, 2014). Therefore, as long as the quality of the sense distributions given by HCA-WSI are no worse than those from HDP-WSI, it should be worthwhile switching in terms of scalability. This massive reduction in computation time would be of particular benefit to our intended large-scale application.

4.1 Evaluation

We evaluate HCA-WSI in comparison to HDP-WSI using one of the sense tagged datasets of

[7]Version 0.61, obtained from:
`http://www.mloss.org/software/view/527`

[8]For simplicity we use HCA to refer to both the topic modelling algorithm implemented by Buntine and Mishra (2014) as well as the corresponding software suite, whereas elsewhere HCA often only refers to the software.

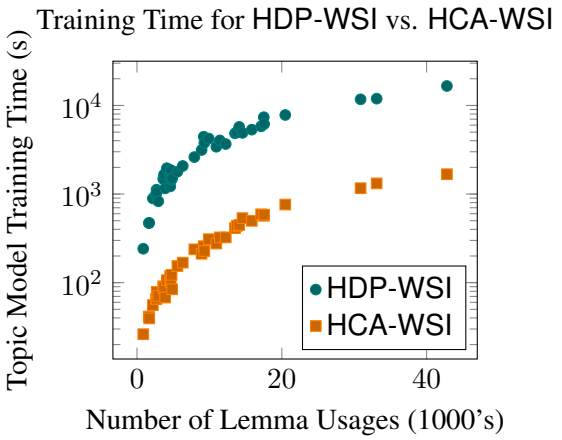


Figure 1: Comparison of the time taken to train the topic models of HDP-WSI and HCA-WSI for each lemma in the BNC dataset. For each method, one data point is plotted per lemma.

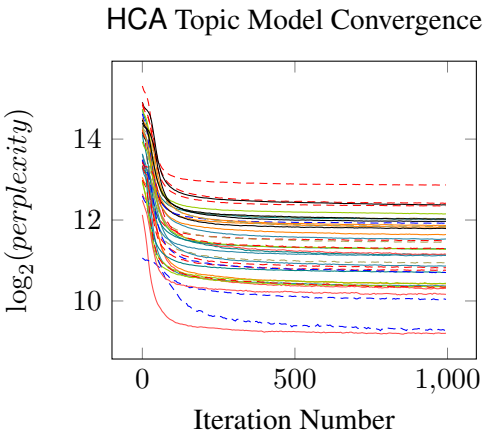


Figure 2: Convergence of log-perplexity of toic model for BNC dataset lemmas, using HCA-WSI. One line per lemma.

Koeling et al. (2005),[9] which was also used by Lau et al. (2014). This dataset consists of 40 English lemmas, and for each lemma it contains a set of usages of varying size from the BNC and a gold-standard sense distribution that was created by hand-annotating a subset of the usages with WORDNET 1.7 senses.

Using this dataset, we can calculate the quality of a candidate sense distribution by calculating its Jensen Shannon divergence (JSD) with respect to the corresponding gold-standard distribution. JSD is a measure of dissimilarity between two probability distributions, so a lower JSD score means the distribution is more similar to the gold-standard, and is therefore assumed to be of higher quality.

Given our finding on topic counts in Section 3, HCA was run using a fixed number of 10 topics. Other settings were configured as recommended in the HCA documentation, or according to the HDP settings used by Lau et al. (2014).[10] This setup is also used in subsequent experiments, except where stated otherwise.

We proceeded by calculating the JSD scores of all lemmas in this dataset, using both methods. We performed a Wilcoxon signed-rank test on the two sequences of JSD scores, in order to test the hypothesis that switching to HCA-WSI has a systematic impact on sense distribution quality. We found that the mean JSD score for HDP-WSI was 0.209 $\pm$ 0.116, slightly lower than the mean JSD score for HCA-WSI of 0.211 $\pm$ 0.117. However the two-sided p-value from the test was 0.221, which is insignificant at any reasonable decision threshold.

In addition, we compared the time taken[11] to run topic modelling for every lemma using both methods, the results of which are displayed in Figure 1. These results show that the computation time of HCA-WSI is consistently lower than that of HDP-WSI, by over an order of magnitude.

We conclude that HCA-WSI is far more computationally efficient than HDP-WSI, and there is no significant evidence that it gives worse sense distributions. Therefore, HCA-WSI is used instead of HDP-WSI for the remainder of the paper.

5 Large-Scale Learning with HCA-WSI

In order to apply HCA-WSI sense distribution learning on a language-wide scale, we need to understand how to optimise it to achieve a reasonable tradeoff between efficiency and sense distribution quality. Most pertinently, we need to know how many lemma usages and iterations of Gibbs sampling are needed for high-quality results, and whether this varies for different kinds of lemmas. To this end, we run experiments ex-

[9] Koeling et al. (2005) also produced domain-specific datasets for the same lemmas, however in order to keep our analysis focussed we only use the domain-neutral BNC dataset.

[10] Initial values for concentration and discount parameters for burstiness were set to 100 and 0.5 respectively, and the number of iterations was set to 300. Other hyperparameters were left with default values.

[11] All benchmarking experiments were run using separate cores on Intel Xeon CPU E5-4650L processors, on a Dell R820 server with 503GiB of main memory.

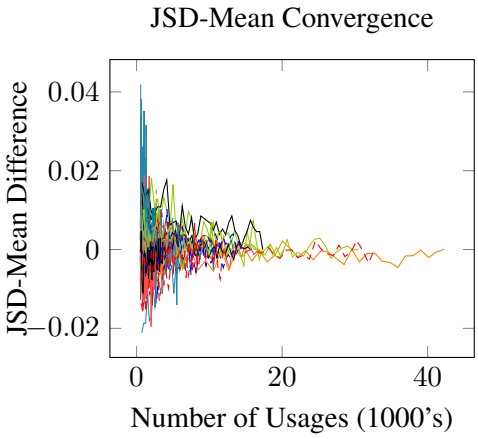


Figure 3: Convergence of mean JSD score for BNC dataset lemmas, using HCA-WSI. One line plotted per lemma, one data-point per bin. For each data-point, the difference between mean JSD within that bin and within the final bin of the lemma is plotted.

ploring how HCA-WSI converges over increasing numbers of lemma usages and topic model iterations. These experiments are all performed using the BNC dataset (see Section 4.1).

In order to explore the convergence of HCA-WSI over Gibbs sampling iterations, we trained HCA topic models for each lemma in the BNC dataset over a large number of iterations. The results of this are displayed in Figure 2, which shows the convergence of log-perplexity for each lemma. We conclude that around 300 iterations of sampling appears to be sufficient for convergence in the vast majority of cases.

Next, we explored the convergence of HCA-WSI over lemma usages by subsampling from our training data. For each lemma in the BNC dataset, we created a large number of sense distributions using random subsets of the lemma's usages.[12] Each distribution was generated by randomly selecting a number of usages between a minimum of 500 and the maximum available (uniformly), and randomly sampling that many usages without replacement. From these usages the sense distribution was created using HCA-WSI, and its JSD score relative to the gold-standard was calculated (as in Section 4.1). Finally, the results for each lemma were partitioned into 40 bins of approximately equal size, according to the number of usages sampled.

[12] Approximately 580 random sense distributions were created per lemma.

The results of our subsampling experiment are plotted in Figure 3, which shows the convergence of mean JSD score for each lemma. We conclude from this that around 5,000–10,000 usages seem to be necessary for convergent results, and that this is fairly consistent across lemmas.[13]

6 LexSemTM Dataset

We now discuss the creation of the LexSemTM ("Lexical Semantic Topic Models") dataset, which contains trained topic models for the majority of simplex English lemmas. These can be aligned to any sense repository with glosses to produce sense distributions, or used directly in other applications. In addition, the dataset contains distributions over WordNet 3.0 senses.

In order to produce domain-neutral sense distributions reflecting usage in modern English, we sampled all lemma usages from English Wikipedia.[14] Our Wikipedia corpus was tokenised and POS-tagged using OpenNLP and lemmatised using Morpha (Minnen et al., 2001).

We trained topic models for every simplex lemma in WordNet 3.0 with at least 20 usages in our processed Wikipedia corpus. This included lemmas for all POS (nouns, verbs, adjectives, and adverbs), and also nonpolysemous lemmas. In Section 5, we concluded that approximately 5,000–10,000 usages were needed for convergent results with the BNC dataset. On the other hand, given that we are working on a different corpus and with a wider range of lemmas there is uncertainty in this number, so we conservatively sampled up to 40,000 usages per lemma, if available.

These usages were sampled from the corpus by locating all sentences where either the surface or lemmatised forms of the sentence contained the target lemma, along with a matching POS-tag. Processing of lemma usages was done almost identically to Lau et al. (2014). However, because we found the usages contained substantially fewer tokens on average compared to the BNC dataset, we included two sentences rather than one on either side of the target lemma location where possible (giving 5 sentences in total), which gave a

[13] We also ran extensive experiments to test the impact of training single topic models over multiple lemmas, using a wide variety of sampling methods, but found the impact to be neutral at best in terms of both the quality of the learned sense distributions and the overall computational cost.

[14] The English Wikipedia dump is dated 2009-11-28.

better match in usage size.

Topic models were trained using HCA, using almost the same setup as described in Section 4.1. However, since some highly-polysemous lemmas may require a greater number of topics than the lemmas in the BNC dataset, we conservatively increased the number of topics used from 10 to 20. We similarly increased the number of Gibbs sampling iterations from 300 to 1,000.[15] Finally, for each polysemous lemma that we trained a topic model for, we also produced a sense distribution over WORDNET 3.0 senses, using the default topic–sense alignment method discussed in Section 3.

In total, 62,721 lemmas were processed, and 8,801 of these had the desired number of at least 5,000 usages. Counting only polysemous lemmas for which we also provide sense distributions, 25,155 were processed in total, and 6,853 of these had at least 5,000 usages. This works out to approximately 88% coverage of polysemous WORDNET 3.0 lemmas in total, or 24% coverage with at least 5,000 usages (as compared to 39% coverage by lemmas in SEMCOR, or 17% with at least 5 sense-tagged occurrences in SEMCOR).

7 Evaluation of LEXSEMTM against SEMCOR

Our final major contribution is an analysis of how our LEXSEMTM sense distributions compare with SEMCOR. We produce a new set of gold-standard sense distributions for a diverse set of simplex English lemmas tagged with WORDNET 3.0 senses, created using crowdsourced annotations of English Wikipedia usages. We use these gold-standard distributions to investigate when LEXSEMTM should be used in place of SEMCOR, and release them as a public resource, to facilitate the evaluation of future work involving LEXSEMTM.

7.1 Gold-Standard Distributions

One of our goals in creating this dataset was to determine whether there is a SEMCOR frequency cutoff,[16] below which our LEXSEMTM distributions are clearly more accurate than SEMCOR. In order to have a diverse set of lemmas and be able to address this question, we partitioned the lemmas in WORDNET 3.0 based on SEMCOR frequency.

In order to keep analysis simple and consistent with previous investigations, we first filtered out multiword lemmas, nonpolysemous lemmas, and non-nouns.[17] Next, since in Section 5 we decided that at least around 5,000 usages were needed for stable and converged sense distributions, we filtered out all lemmas without at least 5,000 usages in our English Wikipedia corpus. The remaining lemmas were then split into 5 groups of approximately equal size based on SEMCOR frequency. The SEMCOR frequencies contained in each group are summarised in Table 1.

From each of the SEMCOR frequency groups, we randomly sampled 10 lemmas, giving 50 lemmas in total. Then for each lemma, we randomly sampled 100 usages to be annotated from English Wikipedia. This was done in the same way as the sampling of lemma usages for LEXSEMTM (see Section 6).

We obtained crowdsourced sense annotations for each lemma using Amazon Mechanical Turk (AMT: Callison-Burch and Dredze (2010)). The sentences for each lemma were split into 4 batches (25 sentences per batch). In addition, two control sentences[18] were created for each lemma, and added to each corresponding batch. Each batch of 27 items was annotated separately by 10 annotators. For each item to be annotated, annotators were provided with the sentence containing the lemma, the gloss for each sense as listed in WORDNET 3.0[19] and a list of hypernyms and synonyms for each sense. Annotators were asked to assign each item to exactly one sense.

From these crowdsourced annotations, our gold-standard sense distributions were created using MACE (Hovy et al., 2013), which is a general-purpose tool for inferring item labels from multi-annotator, multi-item tasks. It provides a Bayesian framework for modelling item annotations, modelling the individual biases of each annotator, and

[15] These changes had a very minor impact on the HCA-WSI evaluation results obtained in Section 4.1, with an average increase in JSD of 0.001 ± 0.004.

[16] The number of sense annotations in SEMCOR.

[17] We chose to restrict our scope in this evaluation to nouns because much of the prior work has also focussed on nouns, and these are the words we would expect others to care the most about disambiguating, since they are more often context bearing. Also, introducing other POS would require a greater quantity of expensive annotated data.

[18] These were created manually, to be as clear and unambiguous as possible.

[19] Example sentences were removed only if they were for a different lemma within the corresponding synset.

supports semi-supervised training. MACE was run separately on the usage annotations of each lemma, with the control sentences included to guide training.

Gold-standard sense distributions were obtained from the output of MACE, which includes a list containing the mode label of each item. For each lemma, we removed the control sentence labels from this list, and constructed the gold-standard distribution from the remaining labels using maximum likelihood estimation.

7.2 Evaluation of LEXSEMTM

We now use these gold-standard distributions to evaluate the sense distributions in LEXSEMTM relative to SEMCOR. For each of the 50 lemmas that we created gold-standard distributions for, we evaluate the corresponding LEXSEMTM distribution against the gold-standard. In addition, we create benchmark sense distributions for each lemma from SEMCOR counts using maximum likelihood estimation,[20] which we also evaluate against the gold-standards. Evaluation of sense distribution quality using gold-standard distributions is done by calculating JSD, as in Section 4.1.

First, we performed this comparison of LEXSEMTM to SEMCOR JSD scores for all 50 lemmas at once. As in Section 4.1, we calculated the JSD scores for every lemma using each method individually, and compared the difference in values pairwise for statistical significance using a Wilcoxon signed-rank test. The results of this comparison are detailed in Table 1 (final row: Group = All), which shows that JSD is clearly lower for LEXSEMTM distributions compared to SEMCOR, as would be hoped. This difference is statistically significant at $p < 0.05$.

We then performed the same comparison separately within each SEMCOR frequency group (Table 1). First of all, we can see that LEXSEMTM sense distributions strongly outperform SEMCOR-based distributions in Group 1 (lemmas missing from SEMCOR). This is as would be expected, since the SEMCOR-based distributions for this group are based on which sense is listed first in WORDNET, which in the absence of SEMCOR counts is arbitrary. On the other hand, in all other groups (lemmas in SEMCOR) the difference between LEXSEMTM and SEMCOR is not statistically significant ($p > 0.1$ in all cases). This still remains true when we pool together the results from these groups (second last row of Table 1: Group = 2–5). While it appears that LEXSEMTM may still be outperforming SEMCOR on average over these groups (lower JSD on average), we do not have enough statistical power to be sure, given the high variance.

Returning to the initial question regarding a SEMCOR frequency cutoff, the only strong conclusion we can make is that LEXSEMTM is clearly superior for lemmas missing from SEMCOR. Although it appears that LEXSEMTM may outperform SEMCOR for lemmas with higher SEMCOR frequencies, the variance in our results is too high to be sure of this, let alone define a frequency cutoff. However, given that LEXSEMTM sense distributions never appear to be worse than SEMCOR-based distributions, regardless of SEMCOR frequency — and may actually be marginally superior — it seems reasonable to use our sense distributions in general in place of SEMCOR.

We can contrast this result to the findings of McCarthy et al. (2007), who found that the automatic first sense learning method of McCarthy et al. (2004b) outperformed SEMCOR for words with SEMCOR frequency less than 5. However, their analysis was based on the accuracy of the first sense heuristic, rather than the entire sense distribution, and they used very different datasets to us.[21] Furthermore, their SEMCOR frequency cutoff result was only statistically significant for some variations of their method, and they evaluated over more lemmas[22] meaning that statistical significance was easier to obtain. Given these reasons, their results likely do not contradict ours.

Given that LEXSEMTM contains sense frequencies for 88% of polysemous simplex lemmas in WORDNET, compared to only 39% for SEMCOR, the strong performance of our LEXSEMTM sense distributions for lemmas missing from SEMCOR is extremely significant. Technically these results are only relevant for lemmas where LEXSEMTM was trained on at least 5,000 us-

[20] For lemmas with no SEMCOR annotations, we assign one count to the first-listed sense in WORDNET 3.0.

[21] Their evaluation on the all words task from SENSEVAL-2, which will have more occurrences of the more frequent words, whereas ours is a lexical sample with 100 instances of each word. However, our experiment has a larger dataset ($50 \times 100 = 5000$ instances, as opposed to 786 in total in the SENSEVAL-2 dataset) which makes it more reliable.

[22] They evaluated over 63 lemmas with SEMCOR frequency between 1 and 5, whereas we only evaluated over 14 lemmas (Group 2, and part of Group 3).

Group	Lemma Count	SEMCOR Freqs.	Mean JSD		p
			LEXSEMTM	SEMCOR	
1	10	0	**.100±.080**	.615±.407	.013
2	10	1–3	**.203±.169**	.214±.250	.959
3	10	4–8	**.100±.049**	.103±.133	.878
4	10	9–20	**.148±.069**	.235±.166	.114
5	10	21+	.162±.121	**.156±.131**	.721
2–5	40	1+	**.153±.118**	.177±.184	.591
All	50	0+	**.142±.113**	.265±.301	.046

Table 1: Sense distribution quality for gold-standard dataset lemmas, comparing LEXSEMTM results to the SEMCOR benchmark.

ages, which reduces the coverage of LEXSEMTM to 24%. However, even then this gives us sense frequencies for 1,602 polysemous lemmas missing from SEMCOR, which accounts for over 5% of polysemous simplex lemmas in WORDNET. Furthermore, based on some additional ongoing analysis comparing LEXSEMTM distributions directly to SEMCOR-based distributions across all of LEXSEMTM (not presented here), it appears the decrease in sense distribution quality for lemmas trained on fewer than 5,000 usages is on average fairly small. This is corroborated by our results in Figure 3: we can observe for the lemmas in the BNC dataset that when the number of usages was reduced to 500, the mean change in JSD for each lemma was almost always less than 0.02 and never greater than 0.04, which is small compared to the difference between LEXSEMTM and SEMCOR in each SEMCOR frequency group. This strongly suggests that our conclusions can be extended to lemmas with low LEXSEMTM frequency, though more work is needed to confirm this.

8 Discussion and Future Work

The most immediate extension of our work would be to apply our sense learning method to a broader range of data. In particular, we intend to expand LEXSEMTM by applying HCA-WSI across the vocabularies of languages other than English, and also to multiword lemmas. Another obvious extension would be to further explore the alignment component of HCA-WSI. We currently use a simple approach, and we believe this process could be improved, e.g. by using word embeddings.

In addition, previous work by Lau et al. (2012) and Lau et al. (2014) also provided methods for detecting novel and unattested senses, using the topic modelling output from the WSI step of HDP-WSI. These could be applied with LEXSEMTM— which contains this WSI output as well as sense frequencies — to search for novel and unattested senses throughout the entire vocabulary of English. This could be used to expand existing sense inventories with new senses, for example using the methodology of Cook et al. (2013). Given that LEXSEMTM also contains WSI output for nonpolysemous WORDNET lemmas (37,566 in total), this could be lead to the discovery of many new polysemous lemmas.

In conclusion, we have created extensive resources for future work in NLP and related disciplines. We have produced LEXSEMTM, which was trained on English Wikipedia and spans approximately 88% of polysemous English lemmas. This dataset contains sense distributions for the majority of polysemous lemmas in WORDNET 3.0. It also contains lemma topic models, for both polysemous and nonpolysemous lemmas, which provide rich semantic information about lemma usage, and can be re-aligned to sense inventories to produce new sense distributions at trivial cost. In addition, we have produced gold-standard distributions for a subset of the lemmas in LEXSEMTM, which we have used to demonstrate that LEXSEMTM sense distributions are at least on-par with those based on SEMCOR for lemmas with a reasonable frequency in Wikipedia, and strongly superior for lemmas missing from SEMCOR. Finally, we demonstrated that HCA topic modelling is more efficient than HDP, providing guidance for others who wish to do large-scale unsupervised sense distribution learning.

Acknowledgements

This work was supported in part by a Google Cloud Platform award.

References

Eneko Agirre and Philip Edmonds. 2007. *Word Sense Disambiguation: Algorithms and Applications*. Springer, Dordrecht, Netherlands.

Eneko Agirre and David Martinez. 2004. Unsupervised WSD based on automatically retrieved examples: The importance of bias. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 25–32, Barcelona, Spain.

Sudha Bhingardive, Dhirendra Singh, V. Rudramurthy, Hanumant Redkar, and Pushpak Bhattacharyya. 2015. Unsupervised most frequent sense detection using word embeddings. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1238–1243, Denver, USA.

Or Biran, Samuel Brody, and Noemie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 496–501, Portland, USA.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022.

Jordan Boyd-Graber and David Blei. 2007. PUTOP: Turning predominant senses into a topic model for word sense disambiguation. In *Proceedings of the 4th International Workshop on Semantic Evaluation (SemEval-2007)*, pages 277–281, Prague, Czech Republic.

Jordan Boyd-Graber, David Blei, and Xiaojin Zhu. 2007. A topic model for word sense disambiguation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1024–1033, Prague, Czech Republic.

Samuel Brody, Roberto Navigli, and Mirella Lapata. 2006. Ensemble methods for unsupervised WSD. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 97–104, Sydney, Australia.

Wray L Buntine and Swapnil Mishra. 2014. Experiments with non-parametric topic models. In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2014)*, pages 881–890, New York City, USA.

Lou Burnard. 1995. User reference guide British National Corpus version 1.0. Technical report, Oxford University Computing Services, UK.

Chris Callison-Burch and Mark Dredze. 2010. Creating speech and language data with Amazon's Mechanical Turk. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Human Language Technologies 2009 (NAACL 2009): Workshop on Creating Speech and Text Language Data With Amazon's Mechanical Turk*, pages 1–12, Los Angeles, USA.

Ziqiang Cao, Sujian Li, Yang Liu, Wenjie Li, and Heng Ji. 2015. A novel neural topic model and its supervised extension. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 2210–2216, Austin, USA.

Yee Seng Chan and Hwee Tou Ng. 2005. Word sense disambiguation with distribution estimation. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 1010–1015, Edinburgh, UK.

Yee Seng Chan and Hwee Tou Ng. 2006. Estimating class priors in domain adaptation for word sense disambiguation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 89–96, Sydney, Australia.

Baobao Chang, Wenzhe Pei, and Miaohong Chen. 2014. Inducing word sense with automatically learned hidden concepts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 355–364, Dublin, Ireland.

Changyou Chen, Lan Du, and Wray Buntine. 2011. Sampling table configurations for the hierarchical Poisson-Dirichlet process. In *Machine Learning and Knowledge Discovery in Databases*, volume 6912, pages 296–311. Springer.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar.

Kenneth W. Church. 2000. Empirical estimates of adaptation: The chance of two noriegas is closer to $p/2$ than p^2. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, pages 180–186, Saarbrücken, Germany.

Paul Cook, Jey Han Lau, Michael Rundell, Diana McCarthy, and Timothy Baldwin. 2013. A lexicographic appraisal of an automatic approach for detecting new word senses. In *Proceedings of eLex 2013*, pages 49–65, Tallinn, Estonia.

Gabriel Doyle and Charles Elkan. 2009. Accounting for burstiness in topic models. In *Proceedings of the 26th Annual International Conference on Machine*

Learning (ICML 2009), pages 281–288, Montreal, Canada.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning whom to trust with MACE. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1120–1130, Atlanta, USA.

Peng Jin, Diana McCarthy, Rob Koeling, and John Carroll. 2009. Estimating and exploiting the entropy of sense distributions. In *Proceedings of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL HLT 2009): Short Papers*, pages 233–236, Boulder, USA.

Rob Koeling, Diana McCarthy, and John Carroll. 2005. Domain-specific sense distributions and predominant sense acquisition. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP 2005)*, pages 419–426, Vancouver, Canada.

Mirella Lapata and Chris Brew. 2004. Verb class disambiguation using informative priors. *Computational Linguistics*, 30(1):45–73.

Mirella Lapata and Frank Keller. 2007. An information retrieval approach to sense ranking. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 348–355, Rochester, USA.

Hugo Larochelle and Iain Murray. 2011. The neural autoregressive distribution estimator. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, pages 29–37, Fort Lauderdale, USA.

Jey Han Lau, Paul Cook, Diana McCarthy, David Newman, and Timothy Baldwin. 2012. Word sense induction for novel sense detection. In *Proceedings of the 13th Conference of the EACL (EACL 2012)*, pages 591–601, Avignon, France.

Jey Han Lau, Paul Cook, Diana McCarthy, Spandana Gella, and Timothy Baldwin. 2014. Learning word sense distributions, detecting unattested senses and identifying novel senses using topic models. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 259–270, Baltimore, USA.

Diana McCarthy, Rob Koeling, and Julie Weeds. 2004a. Ranking WordNet senses automatically. Technical Report 569, Department of Informatics, University of Sussex.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2004b. Finding predominant word senses in untagged text. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL 2004)*, pages 280–287, Barcelona, Spain.

Diana McCarthy, Rob Koeling, Julie Weeds, and John Carroll. 2007. Unsupervised acquisition of predominant word senses. *Computational Linguistics*, 33(4):553–590.

George A Miller, Claudia Leacock, and Randee Tengi. 1993. A semantic concordance. In *Proceedings of the ARPA Workshop on Human Language Technology*, pages 303–308, Plainsboro, USA.

Guido Minnen, John Carroll, and Darren Pearce. 2001. Applied morphological processing of English. *Natural Language Engineering*, 7(3):207–223.

Saif Mohammad and Graeme Hirst. 2006. Determining word sense dominance using a thesaurus. In *Proceedings of the 11th Conference of the EACL (EACL 2006)*, pages 121–128, Trento, Italy.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys*, 41:1–69.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1059–1069, Doha, Qatar.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, 21(5):1112–1130.

Jim Pitman and Marc Yor. 1997. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900.

Marten Postma, Ruben Izquierdo, and Piek Vossen. 2015. VUA-background : When to use background information to perform word sense disambiguation. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval-2015)*, pages 345–349, Denver, USA, June.

Eyal Shnarch, Jacob Goldberger, and Ido Dagan. 2011. A probabilistic modeling framework for lexical entailment. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 558–563, Portland, USA.

Niket Tandon, Gerard de Melo, Abir De, and Gerhard Weikum. 2015. Lights, camera, action: Knowledge extraction from movie scripts. In *Proceedings of the 24th International Conference on World Wide Web*, pages 127–128, Florence, Italy.

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. 2006. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101:1566–1581.

Pengtao Xie, Diyi Yang, and Eric Xing. 2015. Incorporating word correlation knowledge into topic modeling. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 725–734, Denver, USA.

The LAMBADA dataset: Word prediction requiring a broad discourse context*

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham†, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, Raquel Fernández‡
CIMeC - Center for Mind/Brain Sciences, University of Trento
{firstname.lastname}@unitn.it, †quanpn90@gmail.com
‡Institute for Logic, Language & Computation, University of Amsterdam
raquel.fernandez@uva.nl

Abstract

We introduce LAMBADA, a dataset to evaluate the capabilities of computational models for text understanding by means of a word prediction task. LAMBADA is a collection of narrative passages sharing the characteristic that human subjects are able to guess their last word if they are exposed to the whole passage, but not if they only see the last sentence preceding the target word. To succeed on LAMBADA, computational models cannot simply rely on local context, but must be able to keep track of information in the broader discourse. We show that LAMBADA exemplifies a wide range of linguistic phenomena, and that none of several state-of-the-art language models reaches accuracy above 1% on this novel benchmark. We thus propose LAMBADA as a challenging test set, meant to encourage the development of new models capable of genuine understanding of broad context in natural language text.

1 Introduction

The recent spurt of powerful end-to-end-trained neural networks for Natural Language Processing (Hermann et al., 2015; Rocktäschel et al., 2016; Weston et al., 2015, a.o.) has sparked interest in tasks to measure the progress they are bringing about in genuine language understanding. Special care must be taken in evaluating such systems, since their effectiveness at picking statistical generalizations from large corpora can lead to the illusion that they are reaching a deeper degree of understanding than they really are. For example, the end-to-end system of Vinyals and Le (2015), trained on large conversational datasets, produces dialogues such as the following:

(1) **Human:** what is your job?
Machine: i'm a lawyer
Human: what do you do?
Machine: i'm a doctor

Separately, the system responses are appropriate for the respective questions. However, when taken together, they are incoherent. The system behaviour is somewhat parrot-like. It can locally produce perfectly sensible language fragments, but it fails to take the meaning of the *broader discourse context* into account. Much research effort has consequently focused on designing systems able to keep information from the broader context into memory, and possibly even perform simple forms of reasoning about it (Hermann et al., 2015; Hochreiter and Schmidhuber, 1997; Ji et al., 2015; Mikolov et al., 2015; Sordoni et al., 2015; Sukhbaatar et al., 2015; Wang and Cho, 2015, a.o.).

In this paper, we introduce the LAMBADA dataset (**LA**nguage **M**odeling **B**roadened to **A**ccount for **D**iscourse **A**spects). LAMBADA proposes a word prediction task where the target item is *difficult to guess* (for English speakers) when only the sentence in which it appears is available, but becomes *easy* when a broader context is presented. Consider Example (1) in Figure 1. The sentence *Do you honestly think that I would want you to have a ___?* has a multitude of possible continuations, but the broad context clearly indicates that the missing word is *miscarriage*.

LAMBADA casts language understanding in the classic word prediction framework of language modeling. We can thus use it to test several existing language modeling architectures, including

*Denis and Germán share first authorship. Marco, Gemma, and Raquel share senior authorship.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1525–1534,
Berlin, Germany, August 7-12, 2016.

systems with capacity to hold longer-term contextual memories. In our preliminary experiments, none of these models came even remotely close to human performance, confirming that LAMBADA is a challenging benchmark for research on automated models of natural language understanding.

2 Related datasets

The CNN/Daily Mail (CNNDM) benchmark recently introduced by Hermann et al. (2015) is closely related to LAMBADA. CNNDM includes a large set of online articles that are published together with short summaries of their main points. The task is to guess a named entity that has been removed from one such summary. Although the data are not normed by subjects, it is unlikely that the missing named entity can be guessed from the short summary alone, and thus, like in LAMBADA, models need to look at the broader context (the article). Differences between the two datasets include text genres (news vs. novels; see Section 3.1) and the fact that missing items in CNNDM are limited to named entities. Most importantly, the two datasets require models to perform different kinds of inferences over broader passages. For CNNDM, models must be able to *summarize* the articles, in order to make sense of the sentence containing the missing word, whereas in LAMBADA the last sentence is not a summary of the broader passage, but a *continuation* of the same story. Thus, in order to succeed, models must instead understand what is a plausible development of a narrative fragment or a dialogue.

Another related benchmark, CBT, has been introduced by Hill et al. (2016). Like LAMBADA, CBT is a collection of book excerpts, with one word randomly removed from the last sentence in a sequence of 21 sentences. While there are other design differences, the crucial distinction between CBT and LAMBADA is that the CBT passages were not filtered to be human-guessable in the broader context only. Indeed, according to the post-hoc analysis of a sample of CBT passages reported by Hill and colleagues, in a large proportion of cases in which annotators could guess the missing word from the broader context, they could also guess it from the last sentence alone. At the same time, in about one fifth of the cases, the annotators could not guess the word even when the broader context was given. Thus, only a small portion of the CBT passages are really probing the model's ability to understand the broader context, which is instead the focus of LAMBADA.

The idea of a book excerpt completion task was originally introduced in the MSRCC dataset (Zweig and Burges, 2011). However, the latter limited context to single sentences, not attempting to measure broader passage understanding.

Of course, text understanding can be tested through other tasks, including entailment detection (Bowman et al., 2015), answering questions about a text (Richardson et al., 2013; Weston et al., 2015) and measuring inter-clause coherence (Yin and Schütze, 2015). While different tasks can provide complementary insights into the models' abilities, we find word prediction particularly attractive because of its naturalness (it's easy to norm the data with non-expert humans) and simplicity. Models just need to be trained to predict the most likely word given the previous context, following the classic language modeling paradigm, which is a much simpler setup than the one required, say, to determine whether two sentences entail each other. Moreover, models can have access to virtually unlimited amounts of training data, as all that is required to train a language model is raw text. On a more general methodological level, word prediction has the potential to probe almost any aspect of text understanding, including but not limited to traditional narrower tasks such as entailment, co-reference resolution or word sense disambiguation.

3 The LAMBADA dataset

3.1 Data collection[1]

LAMBADA consists of *passages* composed of a *context* (on average 4.6 sentences) and a *target sentence*. The context size is the minimum number of complete sentences before the target sentence such that they cumulatively contain at least 50 tokens (this size was chosen in a pilot study). The task is to guess the last word of the target sentence (the *target word*). The constraint that the target word be the last word of the sentence, while not necessary for our research goal, makes the task more natural for human subjects.

The LAMBADA data come from the Book Corpus (Zhu et al., 2015). The fact that it contains unpublished novels minimizes the potential

[1]Further technical details are provided in the Supplementary Material (SM): `http://clic.cimec.unitn.it/lambada/`

(1) *Context:* “Yes, I thought I was going to lose the baby.” “I was scared too,” he stated, sincerity flooding his eyes. “You were ?” “Yes, of course. Why do you even ask?” “This baby wasn’t exactly planned for.”
Target sentence: “Do you honestly think that I would want you to have a _____ ?”
Target word: miscarriage

(2) *Context:* “Why?” “I would have thought you’d find him rather dry,” she said. “I don’t know about that,” said Gabriel. “He was a great craftsman,” said Heather. “That he was,” said Flannery.
Target sentence: “And Polish, to boot,” said _____.
Target word: Gabriel

(3) *Context:* Preston had been the last person to wear those chains, and I knew what I’d see and feel if they were slipped onto my skin-the Reaper’s unending hatred of me. I’d felt enough of that emotion already in the amphitheater. I didn’t want to feel anymore. “Don’t put those on me,” I whispered. “Please.”
Target sentence: Sergei looked at me, surprised by my low, raspy please, but he put down the _____.
Target word: chains

(4) *Context:* They tuned, discussed for a moment, then struck up a lively jig. Everyone joined in, turning the courtyard into an even more chaotic scene, people now dancing in circles, swinging and spinning in circles, everyone making up their own dance steps. I felt my feet tapping, my body wanting to move.
Target sentence: Aside from writing, I ’ve always loved _____.
Target word: dancing

(5) *Context:* He shook his head, took a step back and held his hands up as he tried to smile without losing a cigarette. “Yes you can,” Julia said in a reassuring voice. “I ’ve already focused on my friend. You just have to click the shutter, on top, here.”
Target sentence: He nodded sheepishly, through his cigarette away and took the _____.
Target word: camera

(6) *Context:* In my palm is a clear stone, and inside it is a small ivory statuette. A guardian angel. “Figured if you’re going to be out at night getting hit by cars, you might as well have some backup.” I look at him, feeling stunned. Like this is some sort of sign.
Target sentence: But as I stare at Harlin, his mouth curved in a confident grin, I don’t care about _____.
Target word: signs

(7) *Context:* Both its sun-speckled shade and the cool grass beneath were a welcome respite after the stifling kitchen, and I was glad to relax against the tree’s rough, brittle bark and begin my breakfast of buttery, toasted bread and fresh fruit. Even the water was tasty, it was so clean and cold.
Target sentence: It almost made up for the lack of _____.
Target word: coffee

(8) *Context:* My wife refused to allow me to come to Hong Kong when the plague was at its height and –” “Your wife, Johanne? You are married at last ?” Johanne grinned. “Well, when a man gets to my age, he starts to need a few home comforts.
Target sentence: After my dear mother passed away ten years ago now, I became _____.
Target word: lonely

(9) *Context:* “Again, he left that up to you. However, he was adamant in his desire that it remain a private ceremony. He asked me to make sure, for instance, that no information be given to the newspaper regarding his death, not even an obituary.
Target sentence: I got the sense that he didn’t want anyone, aside from the three of us, to know that he’d even _____.
Target word: died

(10) *Context:* The battery on Logan’s radio must have been on the way out. So he told himself. There was no other explanation beyond Cygan and the staff at the White House having been overrun. Lizzie opened her eyes with a flutter. They had been on the icy road for an hour without incident.
Target sentence: Jack was happy to do all of the _____.
Target word: driving

Figure 1: Examples of LAMBADA passages. Underlined words highlight when the target word (or its lemma) occurs in the context.

usefulness of general world knowledge and external resources for the task, in contrast to other kinds of texts like news data, Wikipedia text, or famous novels. The corpus, after duplicate removal and filtering out of potentially offensive material with a stop word list, contains 5,325 novels and 465 million words. We randomly divided the novels into equally-sized training and development+testing partitions. We built the LAMBADA dataset from the latter, with the idea that models tackling LAMBADA should be trained on raw text from the training partition, composed of 2662 novels and encompassing more than 200M words. Because novels are pre-assigned to one of the two partitions only, LAMBADA passages are self-contained and cannot be solved by exploiting the knowledge in the remainder of the novels, for example background information about the characters involved or the properties of the fictional world in a given novel. The same novel-based division method is used to further split LAMBADA data between development and testing.

To reduce time and cost of dataset collection, we filtered out passages that are relatively easy for standard language models, since such cases are likely to be guessable based on local context alone. We used a combination of four language models, chosen by availability and/or ease of training: a pre-trained recurrent neural network (RNN) (Mikolov et al., 2011) and three models trained on the Book Corpus (a standard 4-gram model, a RNN and a feed-forward model; see SM for details, and note that these are different from the models we evaluated on LAMBADA as described in Section 4 below). Any passage whose target word had probability $\geq$0.00175 according to any of the language models was excluded.

A random sample of the remaining passages were then evaluated by human subjects through the CrowdFlower crowdsourcing service[2] in three steps. For a given passage,

1. one human subject guessed the target word based on the whole passage (comprising the context and the target sentence); if the guess was right,
2. a second subject guessed the target word based on the whole passage; if that guess was also right,
3. more subjects tried to guess the target word based on the target sentence only, until the word was guessed or the number of unsuccessful guesses reached 10; if no subject was able to guess the target word, the passage was added to the LAMBADA dataset.

The subjects in step 3 were allowed 3 guesses per sentence, to maximize the chances of catching cases where the target words were guessable from the sentence alone. Step 2 was added based on a pilot study that revealed that, while step 3 was enough to ensure that the data could not be guessed with the local context only, step 1 alone did not ensure that the data were easy given the discourse context (its output includes a mix of cases ranging from obvious to relatively difficult, guessed by an especially able or lucky step-1 subject). We made sure that it was not possible for the same subject to judge the same item in both passage and sentence conditions (details in SM).

In the crowdsourcing pipeline, 84–86% items were discarded at step 1, an additional 6–7% at step 2 and another 3–5% at step 3. Only about one in 25 input examples passed all the selection steps.

Subjects were paid \$0.22 per page in steps 1 and 2 (with 10 passages per page) and \$0.15 per page in step 3 (with 20 sentences per page). Overall, each item in the resulting dataset costed \$1.24 on average. Alternative designs, such as having step 3 before step 2 or before step 1, were found to be more expensive. Cost considerations also precluded us from using more subjects at stage 1, which could in principle improve the quality of filtering at this step.

Note that the criteria for passage inclusion were very strict: We required two consecutive subjects to exactly match the missing word, and we made sure that no subject (out of ten) was able to provide it based on local context only, even when given 3 guesses. An alternative to this perfect-match approach would have been to include passages where broad-context subjects provided other plausible or synonymous continuations. However, it is very challenging, both practically and methodologically, to determine which answers other than the original fit the passage well, especially when the goal is to distinguish between items that are solvable in broad-discourse context and those where the local context is enough. Theoretically, substitutability in context could be tested with manual annotation by multiple additional raters, but this would not be financially or practically feasible for a dataset of this scale (human annotators received

[2] http://www.crowdflower.com

over 200,000 passages at stage 1). For this reason we went for the strict hit-or-miss approach, keeping only items that can be unambiguously determined by human subjects.

3.2 Dataset statistics

The LAMBADA dataset consists of 10,022 passages, divided into 4,869 development and 5,153 test passages (extracted from 1,331 and 1,332 disjoint novels, respectively). The average passage consists of 4.6 sentences in the context plus 1 target sentence, for a total length of 75.4 tokens (dev) / 75 tokens (test). Examples of passages in the dataset are given in Figure 1.

The training data for language models to be tested on LAMBADA include the full text of 2,662 novels (disjoint from those in dev+test), comprising 203 million words. Note that the training data consists of text from the same domain as the dev+test passages, in large amounts but not filtered in the same way. This is partially motivated by economic considerations (recall that each data point costs $1.24 on average), but, more importantly, it is justified by the intended use of LAMBADA as a tool to evaluate general-purpose models in terms of how they fare on broad-context understanding (just like our subjects could predict the missing words using their more general text understanding abilities), not as a resource to develop *ad-hoc* models only meant to predict the final word in the sort of passages encountered in LAMBADA. The development data can be used to fine-tune models to the specifics of the LAMBADA passages.

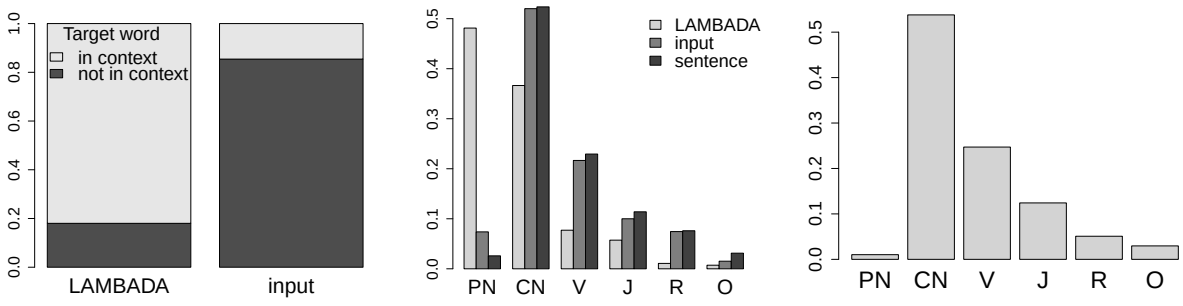
3.3 Dataset analysis

Our analysis of the LAMBADA data suggests that, in order for the target word to be predictable in a broad context only, it must be strongly cued in the broader discourse. Indeed, it is typical for LAMBADA items that the target word (or its lemma) occurs in the context. Figure 2(a) compares the LAMBADA items to a random 5000-item sample from the input data, that is, the passages that were presented to human subjects in the filtering phase (we sampled from all passages passing the automated filters described in Section 3.1 above, including those that made it to LAMBADA). The figure shows that when subjects guessed the word (only) in the broad context, often the word itself occurred in the context: More than 80% of LAMBADA passages include the target word in the context, while in the input data that was the case for less than 15% of the passages. To guess the right word, however, subjects must still put their linguistic and general cognitive skills to good use, as shown by the examples featuring the target word in the context reported in Figure 1.

Figure 2(b) shows that most target words in LAMBADA are proper nouns (48%), followed by common nouns (37%) and, at a distance, verbs (7.7%). In fact, proper nouns are hugely over-represented in LAMBADA, while the other categories are under-represented, compared to the POS distribution in the input. A variety of factors converges in making proper nouns easy for subjects in the LAMBADA task. In particular, when the context clearly demands a referential expression, the constraint that the blank be filled by a single word excludes other possibilities such as noun phrases with articles, and there are reasons to suspect that co-reference is easier than other discourse phenomena in our task (see below). However, although co-reference seems to play a big role, only 0.3% of target words are pronouns.

Common nouns are still pretty frequent in LAMBADA, constituting over one third of the data. Qualitative analysis reveals a mixture of phenomena. Co-reference is again quite common (see Example (3) in Figure 1), sometimes as "partial" co-reference facilitated by bridging mechanisms (*shutter–camera*; Example (5)) or through the presence of a near synonym (*'lose the baby'–miscarriage*; Example (1)). However, we also often find other phenomena, such as the inference of prototypical participants in an event. For instance, if the passage describes someone having breakfast together with typical food and beverages (see Example (7)), subjects can guess the target word *coffee* without it having been explicitly mentioned.

In contrast, verbs, adjectives, and adverbs are rare in LAMBADA. Many of those items can be guessed with local sentence context only, as shown in Figure 2(b), which also reports the POS distribution of the set of items that were guessed by subjects based on the target-sentence context only (step 3 in Section 3.1). Note a higher proportion of verbs, adjectives and adverbs in the latter set in Figure 2(b). While end-of-sentence context skews input distribution in favour of nouns, subject filtering does show a clear differential effect for nouns vs. other POSs. Manual inspection reveals that broad context is not necessary to guess items like



(a) (b) (c)

Figure 2: (a) Target word in or not in context; (b) Target word POS distribution in LAMBADA vs. data presented to human subjects (input) and items guessed with sentence context only (PN=proper noun, CN=common noun, V=verb, J=adjective, R=adverb, O=other); (c) Target word POS distribution of LAMBADA passages where the lemma of the target word is not in the context (categories as in (b)).

frequent verbs (*ask, answer, call*), adjectives, and closed-class adverbs (*now*, *too*, *well*), as well as time-related adverbs (*quickly*, *recently*). In these cases, the sentence context suffices, so few of them end up in LAMBADA (although of course there are exceptions, such as Example (8), where the target word is an adjective). This contrasts with other types of open-class adverbs (e.g., *innocently, confidently*), which are generally hard to guess with both local and broad context. The low proportion of these kinds of adverbs and of verbs among guessed items in general suggests that tracking event-related phenomena (such as script-like sequences of events) is harder for subjects than coreferential phenomena, at least as framed in the LAMBADA task. Further research is needed to probe this hypothesis.

Furthermore, we observe that, while explicit mention in the preceding discourse context is critical for proper nouns, the other categories can often be guessed without having been explicitly introduced. This is shown in Figure 2(c), which depicts the POS distribution of LAMBADA items for which the lemma of the target word is not in the context (corresponding to about 16% of LAMBADA in total).[3] Qualitative analysis of items with verbs and adjectives as targets suggests that the target word, although not present in the passage, is still strongly implied by the context. In about one third of the cases examined, the missing word is "almost there". For instance, the passage contains a word with the same root but a different part of speech (e.g., *death–died* in Example (6)), or a synonymous expression (as mentioned above for "miscarriage"; we find the same phenomenon for verbs, e.g., *'deprived you of water'–dehydrated*).

In other cases, correct prediction requires more complex discourse inference, including guessing prototypical participants of a scene (as in the *coffee* example above), actions or events strongly suggested by the discourse (see Examples (1) and (10), where the mention of an icy road helps in predicting the target *driving*), or qualitative properties of participants or situations (see Example (8)). Of course, the same kind of discourse reasoning takes place when the target word is already present in the context (cf. Examples (3) and (4)). The presence of the word in context does not make the reasoning unnecessary (the task remains challenging), but facilitates the inference.

As a final observation, intriguingly, the LAMBADA items contain (quoted) direct speech significantly more often than the input items overall (71% of LAMBADA items vs. 61% of items in the input sample), see, e.g., Examples (1) and (2). Further analysis is needed to investigate in what way more dialogic discourse might facilitate the prediction of the final target word.

In sum, LAMBADA contains a myriad of phenomena that, besides making it challenging from the text understanding perspective, are of great interest to the broad Computational Linguistics

[3]The apparent 1% of out-of-context proper nouns shown in Figure 2(c) is due to lemmatization mistakes (fictional characters for which the lemmatizer did not recognize a link between singular and plural forms, e.g., *Wynn – Wynns*). A manual check confirmed that all proper noun target words in LAMBADA are indeed also present in the context.

community. To return to Example (1), solving it requires a combination of linguistic skills ranging from (morpho)phonology (the plausible target word *abortion* is ruled out by the indefinite determiner *a*) through morphosyntax (the slot should be filled by a common singular noun) to pragmatics (understanding what the male participant is inferring from the female participant's words), in addition to general reasoning skills. It is not surprising, thus, that LAMBADA is so challenging for current models, as we show next.

4 Modeling experiments

Computational methods We tested several existing language models and baselines on LAMBADA. We implemented a simple **RNN** (Elman, 1990), a Long Short-Term Memory network (**LSTM**) (Hochreiter and Schmidhuber, 1997), a traditional statistical **N-Gram** language model (Stolcke, 2002) with and without **cache**, and a **Memory Network** (Sukhbaatar et al., 2015). We remark that at least LSTM, Memory Network and, to a certain extent, the cache N-Gram model have, among their supposed benefits, the ability to take broader contexts into account. Note moreover that variants of RNNs and LSTMs are at the state of the art when tested on standard language modeling benchmarks (Mikolov, 2014). Our Memory Network implementation is similar to the one with which Hill et al. (2016) reached the best results on the CBT data set (see Section 2 above). While we could not re-implement the models that performed best on CNNDM (see again Section 2), our LSTM is architecturally similar to the Deep LSTM Reader of Hermann et al. (2015), which achieved respectable performance on that data set. Most importantly, we will show below that most of our models reach impressive performance when tested on a more standard language modeling data set sourced from the same corpus used to build LAMBADA. This **control** set was constructed by randomly sampling 5K passages of the same shape and size as the ones used to build LAMBADA from the same test novels, but without filtering them in any way. Based on the control set results, to be discussed below, we can reasonably claim that the models we are testing on LAMBADA are very good at standard language modeling, and their low performance on the latter cannot be attributed to poor quality.

In order to test for strong biases in the data, we constructed **Sup-CBOW**, a baseline model weakly tailored to the task at hand, consisting of a simple neural network that takes as input a bag-of-word representation of the passage and attempts to predict the final word. The input representation comes from adding pre-trained CBOW vectors (Mikolov et al., 2013) of the words in the passage.[4] We also considered an unsupervised variant (**Unsup-CBOW**) where the target word is predicted by cosine similarity between the passage vector and the target word vector. Finally, we evaluated several variations of a random guessing baseline differing in terms of the word pool to sample from. The guessed word could be picked from: the full vocabulary, the words that appear in the current passage and a random uppercased word from the passage. The latter baseline aims at exploiting the potential bias that proper names account for a consistent portion of the LAMBADA data (see Figure 2 above).

Note that LAMBADA was designed to challenge language models with harder-than-average examples where broad context understanding is crucial. However, the average case should not be disregarded either, since we want language models to be able to handle both cases. For this reason, we trained the models entirely on unsupervised data and expect future work to follow similar principles. Concretely, we trained the models, as is standard practice, on predicting each upcoming word given the previous context, using the LAMBADA training data (see Section 3.2 above) as input corpus. The only exception to this procedure was Sup-CBOW where we extracted from the training novels similar-shaped passages to those in LAMBADA and trained the model on them (about 9M passages). Again, the goal of this model was only to test for potential biases in the data and not to provide a full account for the phenomena we are testing. We restricted the vocabulary of the models to the 60K most frequent words in the training set (covering 95% of the target words in the development set). The model hyperparameters were tuned on their accuracy in the development set. The same trained models were tested on the LAMBADA and the control sets. See SM for the tuning details.

Results Results of models and baselines are reported in Table 1. Note that the measure of interest

[4] `http://clic.cimec.unitn.it/composes/semantic-vectors.html`

Data	Method	Acc.	Ppl.	Rank
LAMBADA	*baselines*			
	Random vocabulary word	0	60000	30026
	Random word from passage	1.6	-	-
	Random capitalized word from passage	7.3	-	-
	Unsup-CBOW	0	57040	16352
	Sup-CBOW	0	47587	4660
	models			
	N-Gram	0.1	3125	993
	N-Gram w/cache	0.1	768	87
	RNN	0	14725	7831
	LSTM	0	5357	324
	Memory Network	0	16318	846
Control	*baselines*			
	Random vocabulary word	0	60000	30453
	Random word from passage	0	-	-
	Random capitalized word from passage	0	-	-
	Unsup-CBOW	0	55190	12950
	Sup-CBOW	3.5	2344	259
	models			
	N-Gram	19.1	285	17
	N-Gram w/cache	19.1	270	18
	RNN	15.4	277	24
	LSTM	21.9	149	12
	Memory Network	8.5	566	46

Table 1: Results of computational methods. Accuracy is expressed in percentage.

for LAMBADA is the average success of a model at predicting the target word, i.e., accuracy (unlike in standard language modeling, we know that the missing LAMBADA words can be precisely predicted by humans, so good models should be able to accomplish the same feat, rather than just assigning a high probability to them). However, as we observe a bottoming effect with accuracy, we also report perplexity and median rank of correct word, to better compare the models.

As anticipated above, and in line with what we expected, all our models have very good performance when called to perform a standard language modeling task on the control set. Indeed, 3 of the models (the N-Gram models and LSTM) can guess the right word in about 1/5 of the cases.

The situation drastically changes if we look at the LAMBADA results, where all models are performing very badly. Indeed, no model is even able to compete with the simple heuristics of picking a random word from the passage, and, especially, a random capitalized word (easily a proper noun). At the same time, the low performance of the latter heuristic in absolute terms (7% accuracy) shows that, despite the bias in favour of names in the passage, simply relying on this will not suffice to obtain good performance on LAMBADA, and models should rather pursue deeper forms of analysis of the broader context (the Sup-CBOW baseline, attempting to directly exploit the passage in a shallow way, performs very poorly). This confirms again that the difficulty of LAMBADA relies mainly on accounting for the information available in a broader context and not on the task of predicting the exact word missing.

In comparative terms (and focusing on perplexity and rank, given the uniformly low accuracy results) we observe a stronger performance of the traditional N-Gram models over the neural-network-based ones, possibly pointing to the difficulty of tuning the latter properly. In particular, the best relative performance on LAMBADA is achieved by N-Gram w/cache, which takes passage statistics into account. While even this model is effectively unable to guess the right word, it achieves a respectable perplexity of 768.

We recognize, of course, that the evaluation we performed is very preliminary, and it must only be taken as a proof-of-concept study of the difficulty of LAMBADA. Better results might be obtained simply by performing more extensive tuning, by adding more sophisticated mechanisms such as attention (Bahdanau et al., 2014), and so forth. Still, we would be surprised if minor modifications of the models we tested led to human-level performance on the task.

We also note that, because of the way we have constructed LAMBADA, standard language models are bound to fail on it by design: one of our first filters (see Section 3.1) was to choose passages where a number of simple language models were failing to predict the upcoming word. However, future research should find ways around this inherent difficulty. After all, humans were still able to solve this task, so a model that claims to have good language understanding ability should be able to succeed on it as well.

5 Conclusion

This paper introduced the new LAMBADA dataset, aimed at testing language models on their ability to take a broad discourse context into account when predicting a word. A number of linguistic phenomena make the target words in LAMBADA easy to guess by human subjects when they can look at the whole passages they come from, but nearly impossible if only the last sentence is considered. Our preliminary experiments suggest that even some cutting-edge neural network approaches that are in principle able to track long-distance effects are far from passing the LAMBADA challenge.

We hope the computational community will be stimulated to develop novel language models that are genuinely capturing the non-local phenomena that LAMBADA reflects. To promote research in this direction, we plan to announce a public competition based on the LAMBADA data.[5]

Our own hunch is that, despite the initially disappointing results of the "vanilla" Memory Network we tested, the ability to store information in a longer-term memory will be a crucial component of successful models, coupled with the ability to perform some kind of reasoning about what's stored in memory, in order to retrieve the right information from it.

On a more general note, we believe that leveraging human performance on word prediction is a very promising strategy to construct benchmarks for computational models that are supposed to capture various aspects of human text understanding. The influence of broad context as explored by LAMBADA is only one example of this idea.

[5]The development set of LAMBADA, along with the training corpus, can be downloaded at `http://clic.cimec.unitn.it/lambada/`. The test set will be made available at the time of the competition.

Acknowledgments

We are grateful to Aurelie Herbelot, Tal Linzen, Nghia The Pham and, especially, Roberto Zamparelli for ideas and feedback. This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 655577 (LOVe); ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES); NWO VIDI grant n. 276-89-008 (Asymmetry in Conversation). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used in our research.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. In *ICLR*.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*, pages 632–642, Lisbon, Portugal.

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

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*, Montreal, Canada. Published online: `https://papers.nips.cc/book/advances-in-neural-information-processing-systems-28-2015`.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The Goldilocks principle: Reading children's books with explicit memory representations. In *Proceedings of ICLR Conference Track*, San Juan, Puerto Rico. Published online: `http://www.iclr.cc/doku.php?id=iclr2016:main`.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–178–.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2015. Document context language models. http://arxiv.org/abs/1511.03962.

Tomas Mikolov, Stefan Kombrink, Anoop Deoras, Lukar Burget, and Jan Honza Cernocky. 2011. Rnnlm - recurrent neural network language. In *Proceedings of ASRU*. IEEE Automatic Speech Recognition and Understanding Workshop.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. http://arxiv.org/abs/1301.3781/.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. 2015. Learning longer memory in recurrent neural networks. In *Proceedings of ICLR Workshop Track*, San Diego, CA. Published online: http://www.iclr.cc/doku.php?id=iclr2015:main.

Tomas Mikolov. 2014. Using neural networks for modelling and representing natural languages. Slides presented at COLING, online at http://www.coling-2014.org/COLING\2014\Tutorial-fix\-\TomasMikolov.pdf.

Matthew Richardson, Christopher Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of EMNLP*, pages 193–203, Seattle, WA.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of ICLR Conference Track*, San Juan, Puerto Rico. Published online: http://www.iclr.cc/doku.php?id=iclr2016:main.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of NAACL*, pages 196–205, Denver, CO.

Andreas Stolcke. 2002. Srilm-an extensible language modeling toolkit. In *INTERSPEECH*, volume 2002, page 2002.

Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. http://arxiv.org/abs/1503.08895.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *Proceedings of the ICML Deep Learning Workshop*, Lille, France. Published online: https://sites.google.com/site/deeplearning2015/accepted-papers.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. http://arxiv.org/abs/1511.03729.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. http://arxiv.org/abs/1502.05698.

Wenpeng Yin and Hinrich Schütze. 2015. MultiGranCNN: An architecture for general matching of text chunks on multiple levels of granularity. In *Proceedings of ACL*, pages 63–73, Beijing, China.

Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV 2015*, pages 19–27.

Geoffrey Zweig and Christopher Burges. 2011. The Microsoft Research sentence completion challenge. Technical Report MSR-TR-2011-129, Microsoft Research.

WIKIREADING: A Novel Large-scale Language Understanding Task over Wikipedia

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey and David Berthelot
Google Research
{dhewlett,allac,llion,ipolosukhin,fto,hanjay,matkelcey,dberth}@google.com

Abstract

We present WIKIREADING, a large-scale natural language understanding task and publicly-available dataset with 18 million instances. The task is to predict textual values from the structured knowledge base Wikidata by reading the text of the corresponding Wikipedia articles. The task contains a rich variety of challenging classification and extraction sub-tasks, making it well-suited for end-to-end models such as deep neural networks (DNNs). We compare various state-of-the-art DNN-based architectures for document classification, information extraction, and question answering. We find that models supporting a rich answer space, such as word or character sequences, perform best. Our best-performing model, a word-level sequence to sequence model with a mechanism to copy out-of-vocabulary words, obtains an accuracy of 71.8%.

1 Introduction

A growing amount of research in natural language understanding (NLU) explores end-to-end deep neural network (DNN) architectures for tasks such as text classification (Zhang et al., 2015), relation extraction (Nguyen and Grishman, 2015), and question answering (Weston et al., 2015). These models offer the potential to remove the intermediate steps traditionally involved in processing natural language data by operating on increasingly raw forms of text input, even unprocessed character or byte sequences. Furthermore, while these tasks are often studied in isolation, DNNs have the potential to combine multiple forms of reasoning within a single model.

Supervised training of DNNs often requires a large amount of high-quality training data. To this end, we introduce a novel prediction task and accompanying large-scale dataset with a range of sub-tasks combining text classification and information extraction. The dataset is made publicly-available at `http://goo.gl/wikireading`. The task, which we call WIKIREADING, is to predict textual values from the open knowledge base *Wikidata* (Vrandečić and Krötzsch, 2014) given text from the corresponding articles on *Wikipedia* (Ayers et al., 2008). Example instances are shown in Table 1, illustrating the variety of subject matter and sub-tasks. The dataset contains 18.58M instances across 884 sub-tasks, split roughly evenly between classification and extraction (see Section 2 for more details).

In addition to its diversity, the WIKIREADING dataset is also at least an order of magnitude larger than related NLU datasets. Many natural language datasets for question answering (QA), such as WIKIQA (Yang et al., 2015), have only thousands of examples and are thus too small for training end-to-end models. Hermann et al. (2015) proposed a task similar to QA, predicting entities in news summaries from the text of the original news articles, and generated a NEWS dataset with 1M instances. The bAbI dataset (Weston et al., 2015) requires multiple forms of reasoning, but is composed of synthetically generated documents. WIKIQA and NEWS only involve pointing to locations within the document, and text classification datasets often have small numbers of output classes. In contrast, WIKIREADING has a rich output space of millions of answers, making it a challenging benchmark for state-of-the-art DNN architectures for QA or text classification.

We implemented a large suite of recent models, and for the first time evaluate them on common grounds, placing the complexity of the task in context and illustrating the tradeoffs inherent in each

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1535–1545,
Berlin, Germany, August 7-12, 2016.

	Categorization		Extraction	
Document	Folkart Towers are twin skyscrapers in the Bayrakli district of the Turkish city of Izmir. Reaching a structural height of 200 m (656 ft) above ground level, they are the tallest ...	Angeles blancos is a Mexican telenovela produced by Carlos Sotomayor for Televisa in 1990. Jacqueline Andere, Rogelio Guerra and Alfonso Iturralde star as the main ...	Canada is a country in the northern part of North America. Its ten provinces and three territories extend from the **Atlantic** to the **Pacific** and northward into the **Arctic Ocean**, ...	Breaking Bad is an American crime drama television series created and produced by Vince Gilligan. The show originally aired on the AMC network for five seasons, from **January 20, 2008**, to ...
Property	country	original language of work	located next to body of water	start time
Answer	Turkey	Spanish	Atlantic Ocean, Arctic Ocean, Pacific Ocean	20 January 2008

Table 1: Examples instances from WIKIREADING. The task is to predict the answer given the document and property. Answer tokens that can be extracted are shown in bold, the remaining instances require classification or another form of inference.

approach. The highest score of 71.8% is achieved by a sequence to sequence model (Kalchbrenner and Blunsom, 2013; Cho et al., 2014) operating on word-level input and output sequences, with special handing for out-of-vocabulary words.

2 WIKIREADING

We now provide background information relating to Wikidata, followed by a detailed description of the WIKIREADING prediction task and dataset.

2.1 Wikidata

Wikidata is a free collaborative knowledge base containing information about approximately 16M *items* (Vrandečić and Krötzsch, 2014). Knowledge related to each item is expressed in a set of *statements*, each consisting of a (`property, value`) tuple. For example, the item `Paris` might have associated statements asserting `(instance of, city)` or `(country, France)`. Wikidata contains over 80M such statements across 884 properties. Items may be linked to articles on Wikipedia.

2.2 Dataset

We constructed the WIKIREADING dataset from Wikidata and Wikipedia as follows: We consolidated all Wikidata statements with the same item and property into a single `(item, property, answer)` triple, where `answer` is a set of values. Replacing each `item` with the text of the linked Wikipedia article (discarding unlinked items) yields a dataset of 18.58M `(document, property, answer)` instances. Importantly, all elements in each instance are human-readable strings, making the task entirely textual. The only modification we made to these strings was to convert timestamps into a human-readable format (e.g., "4 July 1776").

The WIKIREADING task, then, is to predict the answer string for each tuple given the document and property strings. This setup can be seen as similar to information extraction, or question answering where the property acts as a "question". We assigned all instances for each document randomly to either training (12.97M instances), validation (1.88M), and test (3.73M) sets following a 70/10/20 distribution. This ensures that, during validation and testing, all documents are unseen.

2.3 Documents

The dataset contains 4.7M unique Wikipedia articles, meaning that roughly 80% of the English-language Wikipedia is represented. Multiple instances can share the same document, with a mean of 5.31 instances per article (median: 4, max: 879). The most common categories of documents are `human`, `taxon`, `film`, `album`, and `human settlement`, making up 48.8% of the documents and 9.1% of the instances. The mean and median document lengths are 489.2 and 203 words.

2.4 Properties

The dataset contains 884 unique properties, though the distribution of properties across instances is highly skewed: The top 20 properties cover 75% of the dataset, with 99% coverage achieved after 180 properties. We divide the properties broadly into two groups: *Categorical* properties, such as `instance of`, `gender` and `country`, require selecting between a relatively small number of possible answers, while *relational* properties, such as `date of birth`,

Property	Frequency	Entropy
instance of	2,574,038	0.431
sex or gender	941,200	0.189
country	803,252	0.536
date of birth	785,049	0.936
given name	767,916	0.763
occupation	716,176	0.589
country of citizenship	674,560	0.501
located in ... entity	478,372	0.802
place of birth	384,951	0.800
date of death	364,910	0.943

Table 2: Training set frequency and scaled answer entropy for the 10 most frequent properties.

`parent`, and `capital`, typically require extracting rare or totally unique answers from the document.

To quantify this difference, we compute the entropy of the answer distribution A for each property p, scaled to the $[0, 1]$ range by dividing by the entropy of a uniform distribution with the same number of values, i.e., $\hat{H}(p) = H(A_p)/\log|A_p|$. Properties that represent essentially one-to-one mappings score near 1.0, while a property with just a single answer would score 0.0. Table 2 lists entropy values for a subset of properties, showing that the dataset contains a spectrum of sub-tasks. We label properties with an entropy less than 0.7 as categorical, and those with a higher entropy as relational. Categorical properties cover 56.7% of the instances in the dataset, with the remaining 43.3% being relational.

2.5 Answers

The distribution of properties described above has implications for the answer distribution. There are a relatively small number of very high frequency "head" answers, mostly for categorical properties, and a vast number of very low frequency "tail" answers, such as names and dates. At the extremes, the most frequent answer `human` accounts for almost 7% of the dataset, while 54.7% of the answers in the dataset are unique. There are some special categories of answers which are systematically related, in particular *dates*, which comprise 8.9% of the dataset (with 7.2% being unique). This distribution means that methods focused on either head or tail answers can each perform moderately well, but only a method that handles both types of answers can achieve maximum performance. Another consequence of the long tail of answers is that many (30.0%) of the answers in the test set never appear in the training set, meaning they must be read out of the document. An answer is present verbatim in the document for 45.6% of the instances.

3 Methods

Recently, neural network architectures for NLU have been shown to meet or exceed the performance of traditional methods (Zhang et al., 2015; Dai and Le, 2015). The move to deep neural networks also allows for new ways of combining the property and document, inspired by recent research in the field of question answering (with the property serving as a question). In sequential models such as Recurrent Neural Networks (RNNs), the question could be prepended to the document, allowing the model to "read" the document differently for each question (Hermann et al., 2015). Alternatively, the question could be used to compute a form of *attention* (Bahdanau et al., 2014) over the document, to effectively focus the model on the most predictive words or phrases (Sukhbaatar et al., 2015; Hermann et al., 2015). As this is currently an ongoing field of research, we implemented a range of recent models and for the first time compare them on common grounds. We now describe these methods, grouping them into broad categories by general approach and noting necessary modifications. Later, we introduce some novel variations of these models.

3.1 Answer Classification

Perhaps the most straightforward approach to WIKIREADING is to consider it as a special case of document classification. To fit WIKIREADING into this framework, we consider each possible answer as a class label, and incorporate features based on the property so that the model can make different predictions for the same document. While the number of potential answers is too large to be practical (and unbounded in principle), a substantial portion of the dataset can be covered by a model with a tractable number of answers.

3.1.1 Baseline

The most common approach to document classification is to fit a linear model (e.g., Logistic Regression) over bag of words (BoW) features. To serve as a baseline for our task, the linear model needs to make different predictions for the same Wikipedia article depending on the property. We enable this behavior by computing two N_{w} element BoW vectors, one each for the document

and property, and concatenating them into a single $2N_{\text{w}}$ feature vector.

3.1.2 Neural Network Methods

All of the methods described in this section encode the property and document into a *joint representation* $\mathbf{y} \in \mathbb{R}^{d_{\text{out}}}$, which serves as input for a final softmax layer computing a probability distribution over the top N_{ans} answers. Namely, for each answer $i \in \{1, \ldots, N_{\text{ans}}\}$, we have:

$$P(i|\mathbf{x}) = e^{\mathbf{y}^\top \mathbf{a}_i} / \sum_{j=1}^{N_{\text{ans}}} e^{\mathbf{y}^\top \mathbf{a}_j}, \qquad (1)$$

where $\mathbf{a}_i \in \mathbb{R}^{d_{\text{out}}}$ corresponds to a learned vector associated with answer i. Thus, these models differ primarily in how they combine the property and document to produce the joint representation. For existing models from the literature, we provide a brief description and note any important differences in our implementation, but refer the reader to the original papers for further details.

Except for character-level models, documents and properties are tokenized into words. The N_{w} most frequent words are mapped to a vector in $\mathbb{R}^{d_{\text{in}}}$ using a learned embedding matrix[1]. Other words are all mapped to a special out of vocabulary (OOV) token, which also has a learned embedding. d_{in} and d_{out} are hyperparameters for these models.

Averaged Embeddings (BoW): This is the neural network version of the baseline method described in Section 3.1.1. Embeddings for words in the document and property are separately averaged. The concatenation of the resulting vectors forms the joint representation of size $2d_{\text{in}}$.

Paragraph Vector: We explore a variant of the previous model where the document is encoded as a *paragraph vector* (Le and Mikolov, 2014). We apply the PV-DBOW variant that learns an embedding for a document by optimizing the prediction of its constituent words. These unsupervised document embeddings are treated as a fixed input to the supervised classifier, with no fine-tuning.

LSTM Reader: This model is a simplified version of the Deep LSTM Reader proposed by Hermann et al. (2015). In this model, an LSTM (Hochreiter and Schmidhuber, 1997) reads the property and document sequences word-by-word and the final state is used as the joint representation. This is the simplest model that respects the order of the words in the document. In our implementation we use a single layer instead of two and a larger hidden size. More details on the architecture can be found in Section 4.1 and in Table 4.

Attentive Reader: This model, also presented in Hermann et al. (2015), uses an attention mechanism to better focus on the relevant part of the document for a given property. Specifically, Attentive Reader first generates a representation $\mathbf{u}$ of the property using the final state of an LSTM while a second LSTM is used to read the document and generate a representation $\mathbf{z}_t$ for each word. Then, conditioned on the property encoding $\mathbf{u}$, a normalized attention is computed over the document to produce a weighted average of the word representations $\mathbf{z}_t$, which is then used to generate the joint representation $\mathbf{y}$. More precisely:

$$\begin{aligned}
\mathbf{m}_t &= \tanh(W_1 \operatorname{concat}(\mathbf{z}_t, \mathbf{u})) \\
\alpha_t &= \exp(\mathbf{v}^\intercal \mathbf{m}_t) \\
\mathbf{r} &= \textstyle\sum_t \frac{\alpha_t}{\sum_\tau \alpha_\tau} \mathbf{z}_t \\
\mathbf{y} &= \tanh(W_2 \operatorname{concat}(\mathbf{r}, \mathbf{u})),
\end{aligned}$$

where W_1, W_2, and $\mathbf{v}$ are learned parameters.

Memory Network: Our implementation closely follows the End-to-End Memory Network proposed in Sukhbaatar et al. (2015). This model maps a property p and a list of sentences $\mathbf{x}_1, \ldots, \mathbf{x}_n$ to a joint representation $\mathbf{y}$ by attending over sentences in the document as follows: The input encoder I converts a sequence of words $\mathbf{x}_i = (x_{i1}, \ldots, x_{iL_i})$ into a vector using an embedding matrix (equation 2), where L_i is the length of sentence i.[2] The property is encoded with the embedding matrix U (eqn. 3). Each sentence is encoded into two vectors, a memory vector (eqn. 4) and an output vector (eqn. 5), with embedding matrices M and C, respectively. The property encoding is used to compute a normalized attention vector over the memories (eqn. 6).[3] The joint representation is the sum of the output vectors weighted

[1] Limited experimentation with initialization from publicly-available word2vec embeddings (Mikolov et al., 2013) yielded no improvement in performance.

[2] Our final results use the position encoding method proposed by Sukhbaatar et al. (2015), which incorporates positional information in addition to word embeddings.

[3] Instead of the linearization method of Sukhbaatar et al. (2015), we applied an entropy regularizer for the softmax attention as described in Kurach et al. (2015).

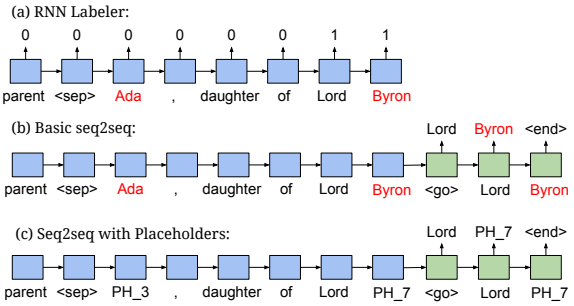


Figure 1: Illustration of RNN models. Blocks with same color share parameters. Red words are out of vocabulary and all share a common embedding.

by this attention (eqn. 7).

$$I(x_i, W) = \sum_j W x_{ij} \quad (2)$$

$$\mathbf{u} = I(p, U) \quad (3)$$

$$\mathbf{m}_i = I(x_i, M) \quad (4)$$

$$\mathbf{c}_i = I(x_i, C) \quad (5)$$

$$\mathbf{p}_i = \text{softmax}(\mathbf{q}^\intercal \mathbf{m}_i) \quad (6)$$

$$\mathbf{y} = \mathbf{u} + \sum_i \mathbf{p}_i \mathbf{c}_i \quad (7)$$

3.2 Answer Extraction

Relational properties involve mappings between arbitrary entities (e.g., `date of birth`, `mother`, and `author`) and thus are less amenable to document classification. For these, approaches from information extraction (especially relation extraction) are much more appropriate. In general, these methods seek to identify a word or phrase in the text that stands in a particular relation to a (possibly implicit) subject. Section 5 contains a discussion of prior work applying NLP techniques involving entity recognition and syntactic parsing to this problem.

RNNs provide a natural fit for extraction, as they can predict a value at every position in a sequence, conditioned on the entire previous sequence. The most straightforward application to WIKIREADING is to predict the probability that a word at a given location is part of an answer. We test this approach using an RNN that operates on the sequence of words. At each time step, we use a sigmoid activation for estimating whether the current word is part of the answer or not. We refer to this model as the *RNN Labeler* and present it graphically in Figure 1a.

For training, we label all locations where any answer appears in the document with a 1, and other positions with a 0 (similar to distant supervision (Mintz et al., 2009)). For multi-word answers, the word sequences in the document and answer must fully match[4]. Instances where no answer appears in the document are discarded for training. The cost function is the average cross-entropy for the outputs across the sequence. When performing inference on the test set, sequences of consecutive locations scoring above a threshold are chunked together as a single answer, and the top-scoring answer is recorded for submission.[5]

3.3 Sequence to Sequence

Recently, sequence to sequence learning (or *seq2seq*) has shown promise for natural language tasks, especially machine translation (Cho et al., 2014). These models combine two RNNs: an *encoder*, which transforms the input sequence into a vector representation, and a *decoder*, which converts the encoder vector into a sequence of output tokens, one token at a time. This makes them capable, in principle, of approximating any function mapping sequential inputs to sequential outputs. Importantly, they are the first model we consider that can perform any combination of answer classification and extraction.

3.3.1 Basic seq2seq

This model resembles LSTM Reader augmented with a second RNN to decode the answer as a sequence of words. The embedding matrix is shared across the two RNNs but their state to state transition matrices are different (Figure 1b). This method extends the set of possible answers to any sequence of words from the document vocabulary.

3.3.2 Placeholder seq2seq

While Basic seq2seq already expands the expressiveness of LSTM Reader, it still has a limited vocabulary and thus is unable to generate some answers. As mentioned in Section 3.2, RNN Labeler can extract any sequence of words present in the document, even if some are OOV. We extend the basic seq2seq model to handle OOV words by adding *placeholders* to our vocabulary, increasing the vocabulary size from N_w to $N_\text{w} + N_\text{doc}$. Then, when an OOV word occurs in the document, it is replaced at random (without replacement). by one of these placeholders. We also replace the corresponding OOV words in the target output se-

[4]Dates were matched semantically to increase recall.

[5]We chose an arbitrary threshold of 0.5 for chunking. The score of each chunk is obtained from the harmonic mean of the predicted probabilities of its elements.

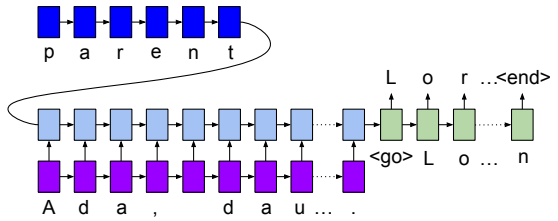


Figure 2: Character seq2seq model. Blocks with the same color share parameters. The same example as in Figure 1 is fed character by character.

quence by the same placeholder,[6] as shown in Figure 1c. Luong et al. (2015) developed a similar procedure for dealing with rare words in machine translation, copying their locations into the output sequence for further processing.

This makes the input and output sequences a mixture of known words and placeholders, and allows the model to produce any answer the RNN Labeler can produce, in addition to the ones that the basic seq2seq model could already produce. This approach is comparable to *entity anonymization* used in Hermann et al. (2015), which replaces named entities with random ids, but simpler because we use word-level placeholders without entity recognition.

3.3.3 Basic Character seq2seq

Another way of handling rare words is to process the input and output text as sequences of characters or bytes. RNNs have shown some promise working with character-level input, including state-of-the-art performance on a Wikipedia text classification benchmark (Dai and Le, 2015). A model that outputs answers character by character can in principle generate any of the answers in the test set, a major advantage for WIKIREADING.

This model, shown in Figure 2, operates only on sequences of mixed-case characters. The property encoder RNN transforms the property, as a character sequence, into a fixed-length vector. This property encoding becomes the initial hidden state for the second layer of a two-layer document encoder RNN, which reads the document, again, character by character. Finally, the answer decoder RNN uses the final state of the previous RNN to decode the character sequence for the answer.

[6]The same OOV word may occur several times in the document. Our simplified approach will attribute a different placeholder for each of these and will use the first occurrence for the target answer.

3.3.4 Character seq2seq with Pretraining

Unfortunately, at the character level the length of all sequences (documents, properties, and answers) is greatly increased. This adds more sequential steps to the RNN, requiring gradients to propagate further, and increasing the chance of an error during decoding. To address this issue in a classification context, Dai and Le (2015) showed that initializing an LSTM classifier with weights from a language model (LM) improved its accuracy. Inspired by this result, we apply this principle to the character seq2seq model with a two-phase training process: In the first phase, we train a character-level LM on the input character sequences from the WIKIREADING training set (no new data is introduced). In the second phase, the weights from this LM are used to initialize the first layer of the encoder and the decoder (purple and green blocks in Figure 2). After initialization, training proceeds as in the basic character seq2seq model.

4 Experiments

We evaluated all methods from Section 3 on the full test set with a single scoring framework. An answer is correct when there is an exact string match between the predicted answer and the gold answer. However, as describe in Section 2.2, some answers are composed from a set of values (e.g. third example in Table 1). To handle this, we define the *Mean F1* score as follows: For each instance, we compute the F1-score (harmonic mean of precision and recall) as a measure of the degree of overlap between the predicted answer set and the gold set for a given instance. The resulting per-instance F1 scores are then averaged to produce a single dataset-level score. This allows a method to obtain partial credit for an instance when it answers with at least one value from the golden set. In this paper, we only consider methods for answering with a single value, and most answers in the dataset are also composed of a single value, so this Mean F1 metric is closely related to accuracy. More precisely, a method using a single value as answer is bounded by a Mean F1 of 0.963.

4.1 Training Details

We implemented all models in a single framework based on TensorFlow (Abadi et al., 2015) with shared pre-processing and comparable hyperparameters whenever possible. All documents are

Method	Mean F1	Bound	Categorical	Relational	Date	Params
Answer Classifier						
Sparse BoW Baseline	0.438	0.831	0.725	0.063	0.004	500.5M
Averaged Embeddings	0.583		0.849	0.234	0.080	120M
Paragraph Vector	0.552		0.787	0.227	0.033	30M
LSTM Reader	0.680		0.880	0.421	0.311	45M
Attentive Reader	0.693		**0.886**	0.441	0.337	56M
Memory Network	0.612		0.861	0.288	0.055	90.1M
Answer Extraction						
RNN Labeler	0.357	0.471	0.240	0.536	0.626	41M
Sequence to Sequence						
Basic seq2seq	0.708	0.925	0.844	0.530	**0.738**	32M
Placeholder seq2seq	**0.718**	0.948	0.835	**0.565**	0.730	32M
Character seq2seq	0.677	**0.963**	0.841	0.462	0.731	4.1M
Character seq2seq (LM)	0.699	**0.963**	0.851	0.501	0.733	4.1M

Table 3: Results for all methods described in Section 3 on the test set. *F1* is the Mean F1 score described in 4. *Bound* is the upper bound on Mean F1 imposed by constraints in the method (see text for details). The remaining columns provide score breakdowns by property type and the number of model parameters.

truncated to the first 300 words except for Character seq2seq, which uses 400 characters. The embedding matrix used to encode words in the document uses $d_{\text{in}} = 300$ dimensions for the $N_{\text{w}} = 100,000$ most frequent words. Similarly, answer classification over the $N_{\text{ans}} = 50,000$ most frequent answers is performed using an answer representation of size $d_{\text{out}} = 300$.[7] The first 10 words of the properties are embedded using the document embedding matrix. Following Cho et al. (2014), RNNs in seq2seq models use a GRU cell with a hidden state size of 1024. More details on parameters are reported in Table 4.

Method	Emb. Dims	Doc. Length	Property Length	Doc. Vocab. Size
Sparse BoW Baseline	N/A	300 words	10 words	50K words
Paragraph Vector	N/A	N/A	10 words	N/A
Character seq2seq	30	400 chars	20 chars	76 chars
All others	300	300 words	10 words	100K words

Table 4: Structural model parameters. Note that the Paragraph Vector method uses the output from a separate, unsupervised model as a document encoding, which is not counted in these parameters.

Optimization was performed with the Adam stochastic optimizer[8] (Kingma and Adam, 2015) over mini-batches of 128 samples. Gradient clipping [9] (Graves, 2013) is used to prevent instability in training RNNs. We performed a search over 50 randomly-sampled hyperparameter configurations for the learning rate and gradient clip threshold, selecting the one with the highest Mean F1 on the validation set. Learning rate and clipping threshold are sampled uniformly, on a logarithmic scale, over the range $[10^{-5}, 10^{-2}]$ and $[10^{-3}, 10^{1}]$ respectively.

4.2 Results and Discussion

Results for all models on the held-out set of test instances are presented in Table 3. In addition to the overall Mean F1 scores, the model families differ significantly in Mean F1 upper bound, and their relative performance on the relational and categorical properties defined in Section 2.4. We also report scores for properties containing dates, a subset of relational properties, as a separate column since they have a distinct format and organization. For examples of model performance on individual properties, see Table 5.

As expected, all classifier models perform well for categorical properties, with more sophisticated classifiers generally outperforming simpler ones. The difference in precision reading ability between models that use broad document statistics, like Averaged Embeddings and Paragraph Vectors, and the RNN-based classifiers is revealed in the scores for relational and especially date properties. As shown in Table 5, this difference is magnified in situations that are more difficult for a classifier, such as relational properties or properties with fewer training examples, where Attentive Reader outperforms Averaged Embeddings by a wide margin. This model family also has a high

[7] For models like Averaged Embedding and Paragraph Vector, the concatenation imposes a greater d_{out}.

[8] Using $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

[9] When the norm of gradient $\mathbf{g}$ exceeds a threshold C, it is scaled down i.e. $\mathbf{g} \leftarrow \mathbf{g} \cdot \min\left(1, \frac{C}{||\mathbf{g}||}\right)$.

Property	Test Instances	Mean F1						
		Averaged Embeddings	Attentive Reader	Memory Network	Basic seq2seq	Placeholder seq2seq	Character seq2seq	Character seq2seq (LM)
Categorical Properties								
instance of	734187	0.8545	**0.8978**	0.8720	0.8877	0.8775	0.8548	0.8659
sex or gender	267896	0.9917	0.9966	0.9936	**0.9968**	0.9952	0.9943	0.9941
genre	32531	0.5320	**0.6225**	0.5625	0.5511	0.5260	0.5096	0.5283
instrument	3665	0.7621	**0.8415**	0.7886	0.8377	0.8172	0.7529	0.7832
Relational Properties								
given name	218625	0.4973	0.8486	0.7206	0.8669	**0.8868**	0.8606	0.8729
located in	137253	0.4140	0.6195	0.4832	0.5484	**0.6978**	0.5496	0.6365
parent taxon	62685	0.1990	0.3467	0.2077	0.2044	**0.7997**	0.4979	0.5748
author	9517	0.0309	0.2088	0.1050	0.6094	**0.6572**	0.1403	0.3748
Date Properties								
date of birth	223864	0.0626	0.3677	0.0016	**0.8306**	0.8259	0.8294	0.8303
date of death	103507	0.0417	0.2949	0.0506	**0.7974**	0.7874	0.7897	0.7924
publication date	31253	0.3909	0.5549	0.4851	**0.5988**	0.5902	0.5903	0.5943
date of official opening	1119	0.1510	0.3047	0.1725	**0.3333**	0.3012	0.1457	0.1635

Table 5: Property-level Mean F1 scores on the test set for selected methods and properties. For each property type, the two most frequent properties are shown followed by two less frequent properties to illustrate long-tail behavior.

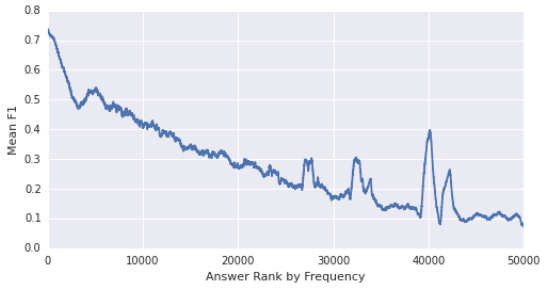


Figure 3: Per-answer Mean F1 scores for Attentive Reader (moving average of 1000), illustrating the decline in prediction quality as the number of training examples per answer decreases.

upper bound, as perfect classification across the 50,000 most frequent answers would yield a Mean F1 of 0.831. However, none of them approaches this limit. Part of the reason is that their accuracy for a given answer decreases quickly as the frequency of the answer in the training set decreases, as illustrated in Figure 3. As these models have to learn a separate weight vector for each answer as part of the softmax layer (see Section 3.1), this may suggest that they fail to generalize across answers effectively and thus require significant number of training examples per answer.

The only answer extraction model evaluated, RNN Labeler, shows a complementary set of strengths, performing better on relational properties than categorical ones. While the Mean F1 upper bound for this model is just 0.434 because it can only produce answers that are present verbatim in the document text, it manages to achieve most of this potential. The improvement on date properties over the classifier models demonstrates its ability to identify answers that are typically present in the document. We suspect that answer extraction may be simpler than answer classification because the model can learn robust patterns that indicate a location without needing to learn about each answer, as the classifier models must.

The sequence to sequence models show a greater degree of balance between relational and categorical properties, reaching performance consistent with classifiers on the categorical questions and with RNN Labeler on relational questions. Placeholder seq2seq can in principle produce any answer that RNN Labeler can, and the performance on relational properties is indeed similar. As shown in Table 5, Placeholder seq2seq performs especially well for properties where the answer typically contains rare words such as the name of a place or person. When the set of possible answer tokens is more constrained, such

as in categorical or date properties, the Basic seq2seq often performs slightly better. Character seq2seq has the highest upper bound, limited to 0.963 only because it cannot produce an answer set with multiple elements. LM pretraining consistently improves the performance of the Character seq2seq model, especially for relational properties as shown in Table 5. The performance of the Character seq2seq, especially with LM pretraining, is a surprising result: It performs comparably to the word-level seq2seq models even though it must copy long character strings when doing extraction and has access to a smaller portion of the document. We found the character based models to be particularly sensitive to hyperparameters. However, using a pretrained language model reduced this issue and significantly accelerated training while improving the final score. We believe that further research on pretraining for character based models could improve this result.

5 Related Work

The goal of automatically extracting structured information from unstructured Wikipedia text was first advanced by Wu and Weld (2007). As Wikidata did not exist at that time, the authors relied on the structured *infoboxes* included in some Wikipedia articles for a relational representation of Wikipedia content. Wikidata is a cleaner data source, as the infobox data contains many slight variations in schema related to page formatting. Partially to get around this issue, the authors restrict their prediction model Kylin to 4 specific infobox classes, and only common attributes within each class.

A substantial body of work in relation extraction (RE) follows the distant supervision paradigm (Craven and Kumlien, 1999), where sentences containing both arguments of a knowledge base (KB) triple are assumed to express the triple's relation. Broadly, these models use these distant labels to identify syntactic features relating the subject and object entities in text that are indicative of the relation. Mintz et al. (2009) apply distant supervision to extracting Freebase triples (Bollacker et al., 2008) from Wikipedia text, analogous to the relational part of WIKIREADING. Extensions to distant supervision include explicitly modelling whether the relation is actually expressed in the sentence (Riedel et al., 2010), and jointly reasoning over larger sets of sentences and relations (Surdeanu et al., 2012). Recently, Rocktäschel et al. (2015) developed methods for reducing the number of distant supervision examples required by sharing information between relations.

6 Conclusion

We have demonstrated the complexity of the WIKIREADING task and its suitability as a benchmark to guide future development of DNN models for natural language understanding. After comparing a diverse array of models spanning classification and extraction, we conclude that end-to-end sequence to sequence models are the most promising. These models simultaneously learned to classify documents and copy arbitrary strings from them. In light of this finding, we suggest some focus areas for future research.

Our character-level model improved substantially after language model pretraining, suggesting that further training optimizations may yield continued gains. Document length poses a problem for RNN-based models, which might be addressed with convolutional neural networks that are easier to parallelize. Finally, we note that these models are not intrinsically limited to English, as they rely on little or no pre-processing with traditional NLP systems. This means that they should generalize effectively to other languages, which could be demonstrated by a multilingual version of WIKIREADING.

Acknowledgments

We thank Jonathan Berant for many helpful comments on early drafts of the paper, and Catherine Finegan-Dollak for an early implementation of RNN Labeler.

References

Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2015. Tensorflow: Large-scale machine learning on heterogeneous systems. *Software available from tensorflow. org*.

Phoebe Ayers, Charles Matthews, and Ben Yates. 2008. *How Wikipedia works: And how you can be a part of it*. No Starch Press.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations (ICLR)*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

Kyunghyun Cho, Bart Van Merriënboer, Çalar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 77–86. AAAI Press.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3061–3069.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. In *Proceedings of the CVSC Workshop*, Sofia, Bulgaria. Association of Computational Linguistics.

Diederik P Kingma and Jimmy Ba Adam. 2015. A method for stochastic optimization. In *International Conference on Learning Representation*.

Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. 2015. Neural random-access machines. In *International Conference on Learning Representations (ICLR)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of The 31st International Conference on Machine Learning, pp. , 2014*, pages 1188?–1196.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thien Huu Nguyen and Ralph Grishman. 2015. Relation extraction: Perspective from convolutional neural networks. In *Proceedings of NAACL-HLT*, pages 39–48.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Machine Learning and Knowledge Discovery in Databases*, pages 148–163. Springer.

Tim Rocktäschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 455–465. Association for Computational Linguistics.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57:78–85.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. May.

Fei Wu and Daniel S Weld. 2007. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 41–50. ACM.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2013–2018.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Optimizing Spectral Learning for Parsing

Shashi Narayan and Shay B. Cohen
School of Informatics
University of Edinburgh
Edinburgh, EH8 9LE, UK
{snaraya2,scohen}@inf.ed.ac.uk

Abstract

We describe a search algorithm for optimizing the number of latent states when estimating latent-variable PCFGs with spectral methods. Our results show that contrary to the common belief that the number of latent states for each nonterminal in an L-PCFG can be decided in isolation with spectral methods, parsing results significantly improve if the number of latent states for each nonterminal is globally optimized, while taking into account interactions between the different nonterminals. In addition, we contribute an empirical analysis of spectral algorithms on eight morphologically rich languages: Basque, French, German, Hebrew, Hungarian, Korean, Polish and Swedish. Our results show that our estimation consistently performs better or close to coarse-to-fine expectation-maximization techniques for these languages.

1 Introduction

Latent-variable probabilistic context-free grammars (L-PCFGs) have been used in the natural language processing community (NLP) for syntactic parsing for over a decade. They were introduced in the NLP community by Matsuzaki et al. (2005) and Prescher (2005), with Matsuzaki et al. using the expectation-maximization (EM) algorithm to estimate them. Their performance on syntactic parsing of English at that stage lagged behind state-of-the-art parsers.

Petrov et al. (2006) showed that one of the reasons that the EM algorithm does not estimate state-of-the-art parsing models for English is that the EM algorithm does not control well for the model size used in the parser – the number of latent states associated with the various nonterminals in the grammar. As such, they introduced a coarse-to-fine technique to estimate the grammar. It splits and merges nonterminals (with latent state information) with the aim to optimize the likelihood of the training data. Together with other types of fine tuning of the parsing model, this led to state-of-the-art results for English parsing.

In more recent work, Cohen et al. (2012) described a different family of estimation algorithms for L-PCFGs. This so-called "spectral" family of learning algorithms is compelling because it offers a rigorous theoretical analysis of statistical convergence, and sidesteps local maxima issues that arise with the EM algorithm.

While spectral algorithms for L-PCFGs are compelling from a theoretical perspective, they have been lagging behind in their empirical results on the problem of parsing. In this paper we show that one of the main reasons for that is that spectral algorithms require a more careful tuning procedure for the number of latent states than that which has been advocated for until now. In a sense, the relationship between our work and the work of Cohen et al. (2013) is analogous to the relationship between the work by Petrov et al. (2006) and the work by Matsuzaki et al. (2005): we suggest a technique for optimizing the number of latent states for spectral algorithms, and test it on eight languages.

Our results show that when the number of latent states is optimized using our technique, the parsing models the spectral algorithms yield perform significantly better than the vanilla-estimated models, and for most of the languages – better than the Berkeley parser of Petrov et al. (2006).

As such, the contributions of this parser are twofold:

- We describe a search algorithm for optimiz-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1546–1556,
Berlin, Germany, August 7-12, 2016.

ing the number of latent states for spectral learning.

- We describe an analysis of spectral algorithms on eight languages (until now the results of L-PCFG estimation with spectral algorithms for parsing were known only for English). Our parsing algorithm is rather language-generic, and does not require significant linguistically-oriented adjustments.

In addition, we dispel the common wisdom that more data is needed with spectral algorithms. Our models yield high performance on treebanks of varying sizes from 5,000 sentences (Hebrew and Swedish) to 40,472 sentences (German).

The rest of the paper is organized as follows. In §2 we describe notation and background. §3 further investigates the need for an optimization of the number of latent states in spectral learning and describes our optimization algorithm, a search algorithm akin to beam search. In §4 we describe our experiments with natural language parsing for Basque, French, German, Hebrew, Hungarian, Korean, Polish and Swedish. We conclude in §5.

2 Background and Notation

We denote by $[n]$ the set of integers $\{1, \ldots, n\}$. An L-PCFG is a 5-tuple $(\mathcal{N}, \mathcal{I}, \mathcal{P}, f, n)$ where:

- $\mathcal{N}$ is the set of nonterminal symbols in the grammar. $\mathcal{I} \subset \mathcal{N}$ is a finite set of *interminals*. $\mathcal{P} \subset \mathcal{N}$ is a finite set of *preterminals*. We assume that $\mathcal{N} = \mathcal{I} \cup \mathcal{P}$, and $\mathcal{I} \cap \mathcal{P} = \emptyset$. Hence we have partitioned the set of nonterminals into two subsets.

- $f: \mathcal{N} \rightarrow \mathbb{N}$ is a function that maps each nonterminal a to the number of latent states it uses. The set $[m_a]$ includes the possible hidden states for nonterminal a.

- $[n]$ is the set of possible words.

- For all $a \in \mathcal{I}$, $b \in \mathcal{N}$, $c \in \mathcal{N}$, $h_1 \in [m_a]$, $h_2 \in [m_b]$, $h_3 \in [m_c]$, we have a binary context-free rule $a(h_1) \rightarrow b(h_2)\ \ c(h_3)$.

- For all $a \in \mathcal{P}$, $h \in [m_a]$, $x \in [n]$, we have a lexical context-free rule $a(h) \rightarrow x$.

The estimation of an L-PCFG requires an assignment of probabilities (or weights) to each of the rules $a(h_1) \rightarrow b(h_2)\ \ c(h_3)$ and $a(h) \rightarrow x$, and also an assignment of starting probabilities for each $a(h)$, where $a \in \mathcal{I}$ and $h \in [m_a]$. Estimation is usually assumed to be done from a set of parse trees (a treebank), where the latent states are not included in the data – only the "skeletal" trees which consist of nonterminals in $\mathcal{N}$.

L-PCFGs, in their symbolic form, are related to regular tree grammars, an old grammar formalism, but they were introduced as statistical models for parsing with latent heads more recently by Matsuzaki et al. (2005) and Prescher (2005). Earlier work about L-PCFGs by Matsuzaki et al. (2005) used the expectation-maximization (EM) algorithm to estimate the grammar probabilities. Indeed, given that the latent states are not observed, EM is a good fit for L-PCFG estimation, since it aims to do learning from incomplete data. This work has been further extended by Petrov et al. (2006) to use EM in a coarse-to-fine fashion: merging and splitting nonterminals using the latent states to optimize the number of latent states for each nonterminal.

Cohen et al. (2012) presented a so-called spectral algorithm to estimate L-PCFGs. This algorithm uses linear-algebraic procedures such as singular value decomposition (SVD) during learning. The spectral algorithm of Cohen et al. builds on an estimation algorithm for HMMs by Hsu et al. (2009).[1] Cohen et al. (2013) experimented with this spectral algorithm for parsing English. A different variant of a spectral learning algorithm for L-PCFGs was developed by Cohen and Collins (2014). It breaks the problem of L-PCFG estimation into multiple convex optimization problems which are solved using EM.

The family of L-PCFG spectral learning algorithms was further extended by Narayan and Cohen (2015). They presented a simplified version of the algorithm of Cohen et al. (2012) that estimates sparse grammars and assigns probabilities (instead of weights) to the rules in the grammar, and as such does not suffer from the problem of negative probabilities that arise with the original spectral algorithm (see discussion in Cohen et al., 2013). In this paper, we use the algorithms by Narayan and Cohen (2015) and Cohen

[1] A related algorithm for weighted tree automata (WTA) was developed by Bailly et al. (2010). However, the conversion from L-PCFGs to WTA is not straightforward, and information is lost in this conversion. See also (Rabusseau et al., 2016).

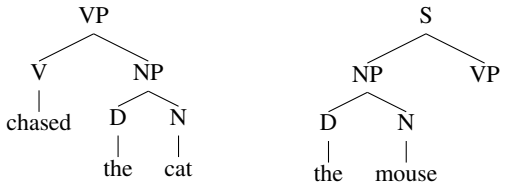


Figure 1: The inside tree (left) and outside tree (right) for the nonterminal `VP` in the parse tree `(S (NP (D the) (N mouse)) (VP (V chased) (NP (D the) (N cat))))` for the sentence *"the mouse chased the cat."*

et al. (2012), and we compare them against state-of-the-art L-PCFG parsers such as the Berkeley parser (Petrov et al., 2006). We also compare our algorithms to other state-of-the-art parsers where elaborate linguistically-motivated feature specifications (Hall et al., 2014), annotations (Crabbé, 2015) and formalism conversions (Fernández-González and Martins, 2015) are used.

3 Optimizing Spectral Estimation

In this section, we describe our optimization algorithm and its motivation.

3.1 Spectral Learning of L-PCFGs and Model Size

The family of spectral algorithms for latent-variable PCFGs rely on feature functions that are defined for *inside* and *outside* trees. Given a tree, the inside tree for a node contains the entire subtree below that node; the outside tree contains everything in the tree excluding the inside tree. Figure 1 shows an example of inside and outside trees for the nonterminal `VP` in the parse tree of the sentence *"the mouse chased the cat"*.

With L-PCFGs, the model dictates that an inside tree and an outside tree that are connected at a node are statistically conditionally independent of each other given the node label and the latent state that is associated with it. As such, one can identify the distribution over the latent states for a given nonterminal a by using the cross-covariance matrix of the inside and the outside trees, Ω^a. For more information on the definition of this cross-covariance matrix, see Cohen et al. (2012) and Narayan and Cohen (2015).

The L-PCFG spectral algorithms use singular value decomposition (SVD) on Ω^a to reduce the dimensionality of the feature functions. If Ω^a is computed from the true L-PCFG distribution then the rank of Ω^a (the number of non-zero singular values) gives the number of latent states according to the model.

In the case of estimating Ω^a from data generated from an L-PCFG, the number of latent states for each nonterminal can be exposed by capping it when the singular values of Ω^a are smaller than some threshold value. This means that spectral algorithms give a natural way for the selection of the number of latent states for each nonterminal a in the grammar.

However, when the data from which we estimate an L-PCFG model are not drawn from an L-PCFG (the model is "incorrect"), the number of non-zero singular values (or the number of singular values which are large) is no longer sufficient to determine the number of latent states for each nonterminal. This is where our algorithm comes into play: it optimizes the number of latent search for each nonterminal by applying a search algorithm akin to beam search.

3.2 Optimizing the Number of Latent States

As mentioned in the previous section, the number of non-zero singular values of Ω^a gives a criterion to determine the number of latent states m_a for a given nonterminal a. In practice, we cap m_a not to include small singular values which are close to 0, because of estimation errors of Ω^a.

This procedure does not take into account the interactions that exist between choices of latent state numbers for the various nonterminals. In principle, given the independence assumptions that L-PCFGs make, choosing the nonterminals based only on the singular values is "statistically correct." However, because in practice the modeling assumptions that we make (that natural language parse trees are drawn from an L-PCFG) do not hold, we can improve further the accuracy of the model by taking into account the nonterminal interaction. Another source of difficulty in choosing the number of latent states based the singular values of Ω^a is sampling error: in practice, we are using data to estimate Ω^a, and as such, even if the model is correct, the rank of the estimated matrix does not have to correspond to the rank of Ω^a according to the true distribution. As a matter of fact, in addition to neglecting small singular values, the spectral methods of Cohen et al. (2013) and Narayan and Cohen (2015) also cap the number of latent states for each nonterminal to an up-

Inputs: An input treebank divided into training and development set. A basic spectral estimation algorithm $\mathcal{S}$ with its default setting. An integer k denoting the size of the beam. An integer m denoting the upper bound on the number of latent states.

Algorithm:
(Step 0: Initialization)

- Set Q, a queue of size k, to be empty.
- Estimate an L-PCFG $G_{\mathcal{S}} : (\mathcal{N}, \mathcal{I}, \mathcal{P}, f_{\mathcal{S}}, n)$ using $\mathcal{S}$.
- Initialize $f = f_{\mathcal{S}}$, a function that maps each nonterminal $a \in \mathcal{N}$ to the number of latent states.
- Let L be a list of nonterminals $(a_1, \ldots, a_M)$ such that $a_i \in \mathcal{N}$ for which to optimize the number of latent states.
- Let s be the F_1 score for the above L-PCFG $G_{\mathcal{S}}$ on the development set.
- Put in Q the element $(s, 1, f, \text{coarse})$.
- The queue is ordered by s, the first element of tuples, in the queue.

(Step 1: Search, repeat until termination happens)

- Dequeue the queue into (s, j, f, t) where j is the index in the input nonterminal list L.
- If $j = (M + 1)$, return f.
- If t is coarse then for each $m_0 \in \{1, 5, 10, \ldots, m\}$:
  - Let f_0 be such that $\forall a \neq a_j \; f_0(a) = f(a)$ and $f_0(a_j) = m_0$.
  - Train an L-PCFG G_0 using $\mathcal{S}$ but with f_0.
  - Let s_0 be the F_1 score for G_0 on the development set.
  - Enqueue into Q: $(s_0, j, f_0, \text{refine})$.
- If t is refine then for each $m_0 \in \{f(a) + \ell \mid \ell \in \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}\}$:
  - Let f_0 be such that $\forall a \neq a_j \; f_0(a) = f(a)$ and $f_0(a_j) = m_0$.
  - Train an L-PCFG G_0 using $\mathcal{S}$ but with f_0.
  - Let s_0 be the F_1 score for G_0 on the development set.
  - Enqueue into Q: $(s_0, j + 1, f_0, \text{coarse})$.

Figure 2: A search algorithm for finding the optimal number of latent states.

per bound to keep the grammar size small.

Petrov et al. (2006) improves over the estimation described in Matsuzaki et al. (2005) by taking into account the interactions between the nonterminals and their latent state numbers in the training data. They use the EM algorithm to split and merge nonterminals using the latent states, and optimize the number of latent states for each nonterminal such that it maximizes the likelihood of a training treebank. Their refined grammar successfully splits nonterminals to various degrees to capture their complexity. We take the analogous step with spectral methods. We propose an algorithm where we first compute Ω^a on the training data and then we optimize the number of latent states for each nonterminal by optimizing the PARSEVAL metric (Black et al., 1991) on a development set.

Our optimization algorithm appears in Figure 2. The input to the algorithm is training and development data in the form of parse trees, a basic spectral estimation algorithm $\mathcal{S}$ in its default setting, an upper bound m on the number of latent states that can be used for the different nonterminals and a beam size k which gives a maximal queue size for the beam. The algorithm aims to learn a function f that maps each nonterminal a to the number of latent states. It initializes f by estimating a default grammar $G_{\mathcal{S}} : (\mathcal{N}, \mathcal{I}, \mathcal{P}, f_{\mathcal{S}}, n)$ using $\mathcal{S}$ and setting $f = f_{\mathcal{S}}$. It then iterates over $a \in \mathcal{N}$, improving f such that it optimizes the PARSEVAL metric on the development set.

The state of the algorithm includes a queue that consists of tuples of the form (s, j, f, t) where f is an assignment of latent state numbers to each nonterminal in the grammar, j is the index of a nonterminal to be explored in the input nonterminal list L, s is the F_1 score on the development set for a grammar that is estimated with f and t is a tag that can either be coarse or refine.

The algorithm orders these tuples by s in the queue, and iteratively dequeues elements from the queue. Then, depending on the label t, it either makes a refined search for the number of latent states for a_j, or a more coarse search. As such, the algorithm can be seen as a variant of a beam search algorithm.

The search algorithm can be used with any training algorithm for L-PCFGs, including the algorithms of Cohen et al. (2013) and Narayan and Cohen (2015). These methods, in their default setting, use a function $f_{\mathcal{S}}$ which maps each nonterminal a to a fixed number of latent states m_a it uses. In this case, $\mathcal{S}$ takes as input training data, in the form of a treebank, decomposes into inside and outside trees at each node in each tree in the training set; and reduces the dimensionality of the inside and outside feature functions by running

lang.		Basque	French	German-N	German-T	Hebrew	Hungarian	Korean	Polish	Swedish
train	sent.	7,577	14,759	18,602	40,472	5,000	8,146	23,010	6,578	5,000
	tokens	96,565	443,113	328,531	719,532	128,065	170,221	301,800	66,814	76,332
	lex. size	25,136	27,470	48,509	77,219	15,971	40,775	85,671	21,793	14,097
	#nts	112	222	208	762	375	112	352	198	148
dev	sent.	948	1,235	1,000	5,000	500	1,051	2,066	821	494
	tokens	13,893	38,820	17,542	76,704	11,305	30,010	25,729	8,391	9,339
test	sent.	946	2,541	1,000	5,000	716	1,009	2,287	822	666
	tokens	11,477	75,216	17,585	92,004	17,002	19,913	28,783	8,336	10,675

Table 1: Statistics about the different datasets used in our experiments for the training ("train"), development ("dev") and test ("test") sets. "sent." denotes the number of sentences in the dataset, "tokens" denotes the total number of words in the dataset, "lex. size" denotes the vocabulary size in the training set and "#nts" denotes the number of nonterminals in the training set after binarization.

SVD on the cross-covariance matrix Ω_a of the inside and the outside trees, for each nonterminal a. Cohen et al. (2013) estimate the parameters of the L-PCFG up to a linear transformation using $f(a)$ non-zero singular values of Ω_a, whereas Narayan and Cohen (2015) use the feature representations induced from the SVD step to cluster instances of nonterminal a in the training data into $f(a)$ clusters; these clusters are then treated as latent states that are "observed." Finally, Narayan and Cohen follow up with a simple frequency count maximum likelihood estimate to estimate the parameters in the L-PCFG with these latent states.

An important point to make is that the learning algorithms of Narayan and Cohen (2015) and Cohen et al. (2013) are relatively fast,[2] in comparison to the EM algorithm. They require only one iteration over the data. In addition, the SVD step of $\mathcal{S}$ for these learning algorithms is computed just once for a large m. The SVD of a lower rank can then be easily computed from that SVD.

4 Experiments

In this section, we describe our setup for parsing experiments on a range of languages.

4.1 Experimental Setup

Datasets We experiment with nine treebanks consisting of eight different morphologically rich languages: Basque, French, German, Hebrew, Hungarian, Korean, Polish and Swedish. Table 1 shows the statistics of 9 different treebanks with their splits into training, development and test sets. Eight out of the nine datasets (Basque, French, German-T, Hebrew, Hungarian, Korean, Polish and Swedish) are taken from the workshop on Statistical Parsing of Morphologically Rich Languages (SPMRL; Seddah et al., 2013). The German corpus in the SPMRL workshop is taken from the TiGer corpus (German-T, Brants et al., 2004). We also experiment with another German corpus, the NEGRA corpus (German-N, Skut et al., 1997), in a standard evaluation split.[3] Words in the SPMRL datasets are annotated with their morphological signatures, whereas the NEGRA corpus does not contain any morphological information.

Data preprocessing and treatment of rare words We convert all trees in the treebanks to a binary form, train and run the parser in that form, and then transform back the trees when doing evaluation using the PARSEVAL metric. In addition, we collapse unary rules into unary chains, so that our trees are fully binarized. The column "#nts" in Table 1 shows the number of nonterminals after binarization in the various treebanks. Before binarization, we also drop all functional information from the nonterminals. We use fine tags for all languages except Korean. This is in line with Björkelund et al. (2013).[4] For Korean, there are 2,825 binarized nonterminals making it impractical to use our optimization algorithm, so we use the coarse tags.

Björkelund et al. (2013) have shown that the morphological signatures for rare words are useful to improve the performance of the Berkeley parser.

[2] It has been documented in several papers that the family of spectral estimation algorithms is faster than algorithms such as EM, not just for L-PCFGs. See, for example, Parikh et al. (2012).

[3] We use the first 18,602 sentences as a training set, the next 1,000 sentences as a development set and the last 1,000 sentences as a test set. This corresponds to an 80%-10%-10% split of the treebank.

[4] In their experiments Björkelund et al. (2013) found that fine tags were not useful for Basque also; they did not find a proper explanation for that. In our experiments, however, we found that fine tags were useful for Basque. To retrieve the fine tags, we concatenate coarse tags with their refinement feature ("AZP") values.

In our preliminary experiments with naïve spectral estimation, we preprocess rare words in the training set in two ways: (i) we replace them with their corresponding POS tags, and (ii) we replace them with their corresponding POS+morphological signatures. We follow Björkelund et al. (2013) and consider a word to be rare if it occurs less than 20 times in the training data. We experimented both with a version of the parser that does not ignore and does ignore letter cases, and discovered that the parser behaves better when case is not ignored.

Spectral algorithms: subroutine choices The latent state optimization algorithm will work with either the clustering estimation algorithm of Narayan and Cohen (2015) or the spectral algorithm of Cohen et al. (2013). In our setup, we first run the latent state optimization algorithm with the clustering algorithm. We then run the spectral algorithm once with the optimized f from the clustering algorithm. We do that because the clustering algorithm is significantly faster to iteratively parse the development set, because it leads to sparse estimates.

Our optimization algorithm is sensitive to the initialization of the number of latent states assigned to each nonterminals as it sequentially goes through the list of nonterminals and chooses latent state numbers for each nonterminal, keeping latent state numbers for other nonterminals fixed. In our setup, we start our search algorithm with the best model from the clustering algorithm, controlling for all hyperparameters; we tune f, the function which maps each nonterminal to a fixed number of latent states m, by running the vanilla version with different values of m for different languages. Based on our preliminary experiments, we set m to 4 for Basque, Hebrew, Polish and Swedish; 8 for German-N; 16 for German-T, Hungarian and Korean; and 24 for French.

We use the same features for the spectral methods as in Narayan and Cohen (2015) for German-N. For the SPMRL datasets we do not use the head features. These require linguistic understanding of the datasets (because they require head rules for propagating leaf nodes in the tree), and we discovered that simple heuristics for constructing these rules did not yield an increase in performance.

We use the `kmeans` function in Matlab to do the clustering for the spectral algorithm of Narayan and Cohen (2015). We experimented with several versions of k-means, and discovered that the version that works best in a set of preliminary experiments is hard k-means.[5]

Decoding and evaluation For efficiency, we use a base PCFG without latent states to prune marginals which receive a value less than 0.00005 in the dynamic programming chart. This is just a bare-bones PCFG that is estimated using maximum likelihood estimation (with frequency count). The parser takes part-of-speech tagged sentences as input. We tag the German-N data using the Turbo Tagger (Martins et al., 2010). For the languages in the SPMRL data we use the MarMot tagger of Müeller et al. (2013) to jointly predict the POS and morphological tags.[6] The parser itself can assign different part-of-speech tags to words to avoid parse failure. This is also particularly important for constituency parsing with morphologically rich languages. It helps mitigate the problem of the taggers to assign correct tags when long-distance dependencies are present.

For all results, we report the F_1 measure of the PARSEVAL metric (Black et al., 1991). We use the `EVALB` program[7] with the parameter file `COLLINS.prm` (Collins, 1999) for the German-N data and the SPMRL parameter file, `spmrl.prm`, for the SPMRL data (Seddah et al., 2013).

In this setup, the latent state optimization algorithm terminates in few hours for all datasets except French and German-T. The German-T data has 762 nonterminals to tune over a large development set consisting of 5,000 sentences, whereas, the French data has a high average sentence length of 31.43 in the development set.[8]

Following Narayan and Cohen (2015), we further improve our results by using multiple spectral models where noise is added to the underlying features in the training set before the estimation of each model.[9] Using the optimized f, we estimate

[5] To be more precise, we use the Matlab function `kmeans` while passing it the parameter `'start'='sample'` to randomly sample the initial centroid positions. In our experiments, we found that default initialization of centroids differs in Matlab14 (random) and in Matlab15 (kmeans++). Our estimation performs better with random initialization.

[6] See Björkelund et al. (2013) for the performance of the MarMot tagger on the SPMRL datasets.

[7] `http://nlp.cs.nyu.edu/evalb/`

[8] To speed up tuning on the French data, we drop sentences with length >46 from the development set, dropping its size from 12,35 to 1,006.

[9] We only use the algorithm of Narayan and Cohen (2015) for the noisy model estimation. They have shown that decoding with noisy models performs better with their sparse

	lang.	Basque	French	German-N	German-T	Hebrew	Hungarian	Korean	Polish	Swedish
Bk	van	69.2	**79.9**	-	81.7	87.8	83.9	71.0	84.1	74.5
	rep	84.3	79.7	-	82.7	89.6	**89.1**	**82.8**	87.1	**75.5**
Cl	van (pos)	69.8	73.9	75.7	78.3	88.0	81.3	68.7	90.3	70.9
	van (rep)	78.6	73.7	-	78.8	88.1	84.7	76.5	90.4	71.4
	opt	**81.2***	76.7	77.8	81.7	90.1	87.2	79.2	92.0	75.2
Sp	van	78.1	78.0	77.6	82.0	89.2	87.7	80.6	91.7	73.4
	opt	79.0	78.1*	**79.0***	**82.9***	**90.3***	87.8*	80.9*	**91.7***	**75.5***
Bk multiple		87.4	82.5	-	85.0	90.5	91.1	84.6	88.4	79.5
Cl multiple		83.4	79.9	82.7	85.1	90.6	89.0	80.8	92.5	78.3
Hall et al. '14		83.7	79.4	-	83.3	88.1	87.4	81.9	91.1	76.0
Crabbé '15		84.0	80.9	-	84.1	90.7	88.3	83.1	92.8	77.9

Table 2: Results on the development datasets. "Bk" makes use of the Berkeley parser with its coarse-to-fine mechanism to optimize the number of latent states (Petrov et al., 2006). For Bk, "van" uses the vanilla treatment of rare words using signatures defined by Petrov et al. (2006), whereas "rep." uses the morphological signatures instead. "Cl" uses the algorithm of Narayan and Cohen (2015) and "Sp" uses the algorithm of Cohen et al. (2013). In Cl, "van (pos)" and "van (rep)" are vanilla estimations (i.e., each nonterminal is mapped to fixed number of latent states) replacing rare words by POS or POS+morphological signatures, respectively. The best of these two models is used with our optimization algorithm in "opt". For Sp, "van" uses the best setting for unknown words as Cl. Best result in each column from the first seven rows is in bold. In addition, our best performing models from rows 3-7 are marked with *. "Bk multiple" shows the best results with the multiple models using product-of-grammars procedure (Petrov, 2010) and discriminative reranking (Charniak and Johnson, 2005). "Cl multiple" gives the results with multiple models generated using the noise induction and decoded using the hierarchical decoding (Narayan and Cohen, 2015). Bk results are not available on the development dataset for German-N. For others, we report Bk results from Björkelund et al. (2013). We also include results from Hall et al. (2014) and Crabbé (2015).

	lang.	Basque	French	German-N	German-T	Hebrew	Hungarian	Korean	Polish	Swedish
Bk		74.7	**80.4**	**80.1**	**78.3**	87.0	85.2	78.6	86.8	80.6
Cl	van	79.6	74.3	76.4	74.1	86.3	86.5	76.5	90.5	76.4
	opt	**81.4***	75.6	78.0	76.0	87.2	88.4	78.4	91.2	79.4
Sp	van	79.9	78.7	78.4	78.0	87.8	89.1	**80.3**	**91.8**	78.4
	opt	80.5	79.1*	79.4*	78.2*	**89.0***	**89.2***	80.0*	**91.8***	**80.9***
Bk multiple		87.9	82.9	84.5	81.3	89.5	91.9	84.3	87.8	84.9
Cl multiple		83.4	80.4	82.7	80.4	89.2	89.9	80.3	92.4	82.8
Hall et al. '14		83.4	79.7	-	78.4	87.2	88.3	80.2	90.7	82.0
F&M '15		85.9	78.8	-	78.7	89.0	88.2	79.3	91.2	82.8
Crabbé '15		84.9	80.8	-	79.3	89.7	90.1	82.7	92.7	83.2

Table 3: Results on the test datasets. "Bk" denotes the best Berkeley parser result reported by the shared task organizers (Seddah et al., 2013). For the German-N data, Bk results are taken from Petrov (2010). "Cl van" shows the performance of the best vanilla models from Table 2 on the test set. "Cl opt" and "Sp opt" give the result of our algorithm on the test set. We also include results from Hall et al. (2014), Crabbé (2015) and Fernández-González and Martins (2015).

80 models for each of noise induction mechanisms in Narayan and Cohen: Dropout, Gaussian (additive) and Gaussian (multiplicative). To decode with multiple noisy models, we train the MaxEnt reranker of Charniak and Johnson (2005).[10] Hierarchical decoding with "maximal tree coverage" over MaxEnt models, further improves our accuracy. See Narayan and Cohen (2015) for more details on the estimation of a diverse set of models, and on decoding with them.

4.2 Results

Table 2 and Table 3 give the results for the various languages.[11] Our main focus is on comparing the coarse-to-fine Berkeley parser (Petrov et al., 2006) to our method. However, for the sake of completeness, we also present results for other parsers, such as parsers of Hall et al. (2014), Fernández-González and Martins (2015) and Crabbé (2015).

In line with Björkelund et al. (2013), our preliminary experiments with the treatment of rare words suggest that morphological features are useful for all SPMRL languages except French. Specifically, for Basque, Hungarian and Korean, improvements are significantly large.

Our results show that the optimization of the

estimates than the dense estimates of Cohen et al. (2013).

[10] Implementation: `https://github.com/BLLIP/bllip-parser`. More specifically, we used the programs `extract-spfeatures`, `cvlm-lbfgs` and `best-indices`. `extract-spfeatures` uses head features, we bypass this for the SPMRL datasets by creating a dummy `heads.cc` file. `cvlm-lbfgs` was used with the default hyperparameters from the Makefile.

[11] See more in `http://cohort.inf.ed.ac.uk/lpcfg/`.

	preterminals				interminals				all			
language	$\sum_i x_i$	$\sum_i y_i$	div.	#nts	$\sum_i x_i$	$\sum_i y_i$	div.	#nts	$\sum_i x_i$	$\sum_i y_i$	div.	#nts
Basque	311	419	196	169	91	227	152	31	402	646	348	200
French	839	715	476	108	1145	1279	906	114	1984	1994	1382	222
German-N	425	567	416	109	323	578	361	99	748	1145	777	208
German-T	1251	890	795	378	1037	1323	738	384	**2288**	**2213**	1533	762
Hebrew	434	442	182	279	169	544	393	96	603	986	575	375
Hungarian	457	415	282	87	186	261	129	25	643	676	411	112
Korean	1077	980	547	331	218	220	150	21	**1295**	**1200**	697	352
Polish	252	311	197	135	132	180	86	63	384	491	283	198
Swedish	191	284	127	106	85	345	266	42	276	629	393	148

Table 4: A comparison of the number of latent states for the different nonterminals before and after running our latent state number optimization algorithm. The index i ranges over preterminals and interminals, with x_i denoting the number of latent states for nonterminal i with the vanilla version of the estimation algorithm and y_i denoting the number of latent states for nonterminal i after running the optimization algorithm. The divergence figure ("div.") is a calculation of $\sum_i |x_i - y_i|$.

number of latent states with the clustering and spectral algorithms indeed improves these algorithms performance, and these increases generalize to the test sets as well. This was a point of concern, since the optimization algorithm goes through many points in the hypothesis space of parsing models, and identifies one that behaves optimally on the development set – and as such it could overfit to the development set. However, this did not happen, and in some cases, the increase in accuracy of the test set after running our optimization algorithm is actually larger than the one for the development set.

While the vanilla estimation algorithms (without latent state optimization) lag behind the Berkeley parser for many of the languages, once the number of latent states is optimized, our parsing models do better for Basque, Hebrew, Hungarian, Korean, Polish and Swedish. For German-T we perform close to the Berkeley parser (78.2 vs. 78.3). It is also interesting to compare the clustering algorithm of Narayan and Cohen (2015) to the spectral algorithm of Cohen et al. (2013). In the vanilla version, the spectral algorithm does better in most cases. However, these differences are narrowed, and in some cases, overcome, when the number of latent states is optimized. Decoding with multiple models further improves our accuracy. Our "Cl multiple" results lag behind "Bk multiple." We believe this is the result of the need of head features for the MaxEnt models.[12]

Our results show that spectral learning is a viable alternative to the use of expectation-maximization coarse-to-fine techniques. As we discuss later, further improvements have been introduced to state-of-the-art parsers that are orthogonal to the use of a specific estimation algorithm. Some of them can be applied to our setup.

4.3 Further Analysis

In addition to the basic set of parsing results, we also wanted to inspect the size of the parsing models when using the optimization algorithm in comparison to the vanilla models. Table 4 gives this analysis. In this table, we see that in most cases, on average, the optimization algorithm chooses to enlarge the number of latent states. However, for German-T and Korean, for example, the optimization algorithm actually chooses a smaller model than the original vanilla model.

We further inspected the behavior of the optimization algorithm for the preterminals in German-N, for which the optimal model chose (on average) a larger number of latent states. Table 5 describes this analysis. We see that in most cases, the optimization algorithm chose to decrease the number of latent states for the various preterminals, but in some cases significantly increases the number of latent states.[13]

Our experiments dispel another "common wisdom" about spectral learning and training data size. It has been believed that spectral learning do not behave very well when small amounts of data are available (when compared to maximum likelihood estimation algorithms such as EM) – however we see that our results do better than the Berkeley parser for several languages with small

[12] Björkelund et al. (2013) also use the MaxEnt raranker with multiple models of the Berkeley parser, and in their case also the performance after the raranking step is not always significantly better. See footnote 10 on how we create dummy head-features for our MaxEnt models.

[13] Interestingly, most of the punctuation symbols, such as `$*LRB*`, `$.` and `$,`, drop their latent state number to a significantly lower value indicating that their interactions with other nonterminals in the tree are minimal.

preterminal	freq.	b.	a.	preterminal	freq.	b.	a.	preterminal	freq.	b.	a.	preterminal	freq.	b.	a.
PWAT	64	2	2	TRUNC	614	8	1	PIS	1,628	8	8	KON	8,633	8	30
XY	135	3	1	VAPP	363	6	4	**$*LRB***	**13,681**	**8**	**6**	PPER	4,979	8	100
NP|NN	88	2	1	PDS	988	8	8	ADJD	6,419	8	60	**$.**	**17,699**	**8**	**3**
VMINF	177	3	5	AVP|ADV	211	4	11	KOUS	2,456	8	1	APPRART	6,217	8	15
PTKA	162	3	1	FM	578	8	3	PIAT	1,061	8	8	ADJA	18,993	8	10
VP|VVINF	409	6	2	VVIMP	76	2	1	NP|PPER	382	6	1	APPR	26,717	8	7
PRELAT	94	2	1	KOUI	339	5	2	VVPP	5,005	8	20	VVFIN	13,444	8	3
AP|ADJD	178	3	1	VAINF	1,024	8	1	PP|PROAV	174	3	1	**$,**	**16,631**	**8**	**1**
APPO	89	2	2	PRELS	2,120	8	40	VAFIN	8,814	8	1	VVINF	4,382	8	10
PWS	361	6	1	CARD	6,826	8	8	PTKNEG	1,884	8	8	ART	35,003	8	10
KOKOM	800	8	37	NE	17,489	8	6	PTKZU	1,586	8	1	ADV	15,566	8	8
VP|VVPP	844	8	5	PRF	2,158	8	1	VVIZU	479	7	1	PIDAT	1,254	8	20
PWAV	689	8	1	PDAT	1,129	8	1	PPOSAT	2,295	8	6	NN	68,056	8	12
APZR	134	3	2	PROAV	1,479	8	10	PTKVZ	1,864	8	3	VMFIN	3,177	8	1

Table 5: A comparison of the number of latent states for each preterminal for the German-N model, before ("b.") running the latent state number optimization algorithm and after running it ("a."). Note that some of the preterminals denote unary rules that were collapsed (the nonterminals in the chain are separated by |). We do not show rare preterminals with b. and a. both being 1.

training datasets, such as Basque, Hebrew, Polish and Hungarian. The source of this common wisdom is that ML estimators tend to be statistically "efficient:" they extract more information from the data than spectral learning algorithms do. Indeed, there is no reason to believe that spectral algorithms are statistically efficient. However, it is not clear that indeed for L-PCFGs with the EM algorithm, the ML estimator is statistically efficient either. MLE is statistically efficient under specific assumptions which are not clearly satisfied with L-PCFG estimation. In addition, when the model is "incorrect," (i.e. when the data is not sampled from L-PCFG, as we would expect from natural language treebank data), spectral algorithms could yield better results because they can mimic a higher order model. This can be understood through HMMs. When estimating an HMM of a low order with data which was generated from a higher order model, EM does quite poorly. However, if the number of latent states (and feature functions) is properly controlled with spectral algorithms, a spectral algorithm would learn a "product" HMM, where the states in the lower order model are the product of states of a higher order.[14]

State-of-the-art parsers for the SPMRL datasets improve the Berkeley parser in ways which are orthogonal to the use of the basic estimation algorithm and the method for optimizing the number of latent states. They include transformations of the treebanks such as with unary rules (Björkelund et al., 2013), a more careful handling of unknown words and better use of morphological information such as decorating preterminals with such information (Björkelund et al., 2014; Szántó and Farkas, 2014), with careful feature specifications (Hall et al., 2014) and head-annotations (Crabbé, 2015), and other techniques. Some of these techniques can be applied to our case.

[14]For example, a trigram HMM can be reduced to a bigram HMM where the states are products of the original trigram HMM.

5 Conclusion

We demonstrated that a careful selection of the number of latent states in a latent-variable PCFG with spectral estimation has a significant effect on the parsing accuracy of the L-PCFG. We described a search procedure to do this kind of optimization, and described parsing results for eight languages (with nine datasets). Our results demonstrate that when comparing the expectation-maximization with coarse-to-fine techniques to our spectral algorithm with latent state optimization, spectral learning performs better on six of the datasets. Our results are comparable to other state-of-the-art results for these languages. Using a diverse set of models to parse these datasets further improves the results.

Acknowledgments

The authors would like to thank David McClosky for his help with running the BLLIP parser and his comments on the paper and also the three anonymous reviewers for their helpful comments. We also thank Eugene Charniak, DK Choe and Geoff Gordon for useful discussions. Finally, thanks to Djamé Seddah for providing us with the SPMRL datasets and to Thomas Müller and Anders Björkelund for providing us the MarMot models. This research was supported by an EPSRC grant (EP/L02411X/1) and an EU H2020 grant (688139/H2020-ICT-2015; SUMMA).

References

Raphaël Bailly, Amaury Habrard, and François Denis. 2010. A spectral approach for probabilistic grammatical inference on trees. In *Proceedings of International Conference on Algorithmic Learning Theory*.

Anders Björkelund, Özlem Çetinoğlu, Richárd Farkas, Thomas Müeller, and Wolfgang Seeker. 2013. (Re)ranking meets morphosyntax: State-of-the-art results from the SPMRL 2013 shared task. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Anders Björkelund, Özlem Çetinoğlu, Agnieszka Faleńska, Richárd Farkas, Thomas Müller, Wolfgang Seeker, and Zsolt Szántó. 2014. Introducing the IMS-Wrocław-Szeged-CIS entry at the SPMRL 2014 shared task: Reranking and morphosyntax meet unlabeled data. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*.

Ezra W. Black, Steven Abney, Daniel P. Flickinger, Claudia Gdaniec, Ralph Grishman, Philip Harrison, Donald Hindle, Robert J. P. Ingria, Frederick Jelinek, Judith L. Klavans, Mark Y. Liberman, Mitchell P. Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of DARPA Workshop on Speech and Natural Language*.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of ACL*.

Shay B. Cohen and Michael Collins. 2014. A provably correct learning algorithm for latent-variable PCFGs. In *Proceedings of ACL*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean F. Foster, and Lyle Ungar. 2012. Spectral learning of latent-variable PCFGs. In *Proceedings of ACL*.

Shay B. Cohen, Karl Stratos, Michael Collins, Dean P. Foster, and Lyle Ungar. 2013. Experiments with spectral learning of latent-variable PCFGs. In *Proceedings of NAACL*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Benoit Crabbé. 2015. Multilingual discriminative lexicalized phrase structure parsing. In *Proceedings of EMNLP*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of ACL-IJCNLP*.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of ACL*.

Daniel Hsu, Sham M. Kakade, and Tong Zhang. 2009. A spectral algorithm for learning hidden Markov models. In *Proceedings of COLT*.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Mário A. T. Figueiredo, and Pedro M. Q. Aguiar. 2010. TurboParsers: Dependency parsing by approximate variational inference. In *Proceedings of EMNLP*.

Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *Proceedings of ACL*.

Thomas Müeller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of EMNLP*.

Shashi Narayan and Shay B. Cohen. 2015. Diversity in spectral learning for natural language parsing. In *Proceedings of EMNLP*.

Ankur P. Parikh, Le Song, Mariya Ishteva, Gabi Teodoru, and Eric P. Xing. 2012. A spectral algorithm for latent junction trees. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proceedings of COLING-ACL*.

Slav Petrov. 2010. Products of random latent variable grammars. In *Proceedings of HLT-NAACL*.

Detlef Prescher. 2005. Head-driven PCFGs with latent-head statistics. In *Proceedings of IWPT*.

Guillaume Rabusseau, Borja Balle, and Shay B. Cohen. 2016. Low-rank approximation of weighted tree automata. In *Proceedings of The 19th International Conference on Artificial Intelligence and Statistics*.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clérgerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of ANLP*.

Zsolt Szántó and Richárd Farkas. 2014. Special techniques for constituent parsing of morphologically rich languages. In *Proceedings of EACL*.

Stack-propagation: Improved Representation Learning for Syntax

Yuan Zhang*
CSAIL, MIT
Cambridge, MA 02139, USA
yuanzh@csail.mit.edu

David Weiss
Google Inc
New York, NY 10027, USA
djweiss@google.com

Abstract

Traditional syntax models typically leverage part-of-speech (POS) information by constructing features from hand-tuned templates. We demonstrate that a better approach is to utilize POS tags as a *regularizer* of learned representations. We propose a simple method for learning a stacked pipeline of models which we call "stack-propagation". We apply this to dependency parsing and tagging, where we use the hidden layer of the tagger network as a representation of the input tokens for the parser. At test time, our parser does not require predicted POS tags. On 19 languages from the Universal Dependencies, our method is 1.3% (absolute) more accurate than a state-of-the-art graph-based approach and 2.7% more accurate than the most comparable greedy model.

1 Introduction

In recent years, transition-based dependency parsers powered by neural network scoring functions have dramatically increased the state-of-the-art in terms of both speed and accuracy (Chen and Manning, 2014; Alberti et al., 2015; Weiss et al., 2015). Similar approaches also achieve state-of-the-art in other NLP tasks, such as constituency parsing (Durrett and Klein, 2015) or semantic role labeling (FitzGerald et al., 2015). These approaches all share a common principle: replace hand-tuned conjunctions of traditional NLP feature templates with continuous approximations learned by the hidden layer of a feed-forward network.

*Research conducted at Google.

However, state-of-the-art dependency parsers depend crucially on the use of predicted part-of-speech (POS) tags. In the pipeline or *stacking* (Wolpert, 1992) method, these are predicted from an independently trained tagger and used as features in the parser. However, there are two main disadvantages of a pipeline: (1) errors from the POS tagger cascade into parsing errors, and (2) POS taggers often make mistakes precisely because they cannot take into account the syntactic context of a parse tree. The POS tags may also contain only coarse information, such as when using the universal tagset of Petrov et al. (2011).

One approach to solve these issues has been to avoid using POS tags during parsing, e.g. either using semi-supervised clustering instead of POS tags (Koo et al., 2008) or building recurrent representations of words using neural networks (Dyer et al., 2015; Ballesteros et al., 2015). However, the best accuracy for these approaches is still achieved by running a POS tagger over the data first and combining the predicted POS tags with additional representations. As an alternative, a wide range of prior work has investigated jointly modeling both POS and parse trees (Li et al., 2011; Hatori et al., 2011; Bohnet and Nivre, 2012; Qian and Liu, 2012; Wang and Xue, 2014; Li et al., 2014; Zhang et al., 2015; Alberti et al., 2015). However, these approaches typically require sacrificing either efficiency or accuracy compared to the best pipeline model, and often they simply re-rank the predictions of a pipelined POS tagger.

In this work, we show how to improve accuracy for both POS tagging and parsing by incorporating stacking into the architecture of a feed-forward network. We propose a continuous form of stacking that allows for easy backpropagation down the pipeline across multiple tasks, a process we call

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1557–1566,
Berlin, Germany, August 7-12, 2016.

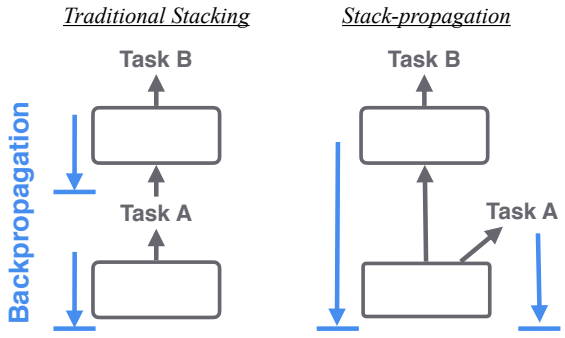


Figure 1: Traditional stacking (left) vs. Stack-propagation (right). Stacking uses the output of Task A as features in Task B, and does not allow backpropagation between tasks. Stack-propagation uses a continuous and differentiable link between Task A and Task B, allowing for backpropagation from Task B into Task A's model. Updates to Task A act as *regularization* on the model for Task B, ensuring the shared component is useful for both tasks.

"stack-propagation" (Figure 1). At the core of this idea is that we use POS tags as *regularization* instead of *features*.

Our model design for parsing is very simple: we use the hidden layer of a window-based POS tagging network as the representation of tokens in a greedy, transition-based neural network parser. Both networks are implemented with a refined version of the feed-forward network (Figure 3) from Chen and Manning (2014), as described in Weiss et al. (2015). We link the tagger network to the parser by translating traditional feature templates for parsing into feed-forward connections from the tagger to the parser (Figure 2). At training time, we unroll the parser decisions and apply stack-propagation by alternating between stochastic updates to the parsing or tagging objectives (Figure 4). The parser's representations of tokens are thus regularized to be individually predictive of POS tags, even as they are trained to be useful for parsing when concatenated and fed into the parser network. This model is similar to the multi-task network structure of Collobert et al. (2011), where Collobert et al. (2011) shares a hidden layer between multiple tagging tasks. The primary difference here is that we show how to unroll parser transitions to apply the same principle to tasks with fundamentally different structure.

The key advantage of our approach is that at test time, we do not require predicted POS tags for parsing. Instead, we run the tagger network up to the hidden layer over the entire sentence, and then dynamically connect the parser network to the tagger network based upon the discrete parser configurations as parsing unfolds. In this way, we avoid cascading POS tagging errors to the parser. As we show in Section 5, our approach can be used in conjunction with joint transition systems in the parser to improve both POS tagging as well as parsing. In addition, because the parser re-uses the representation from the tagger, we can drop all lexicalized features from the parser network, leading to a compact, faster model.

The rest of the paper is organized as follows. In Section 2, we describe the layout of our combined architecture. In Section 3, we introduce stack-propagation and show how we train our model. We evaluate our approach on 19 languages from the Universal Dependencies treebank in Section 4. We observe a $>2\%$ absolute gain in labeled accuracy compared to state-of-the-art, LSTM-based greedy parsers (Ballesteros et al., 2015) and a $>1\%$ gain compared to a state-of-the-art, graph-based method (Lei et al., 2014). We also evaluate our method on the Wall Street Journal, where we find that our architecture outperforms other greedy models, especially when only coarse POS tags from the universal tagset are provided during training. In Section 5, we systematically evaluate the different components of our approach to demonstrate the effectiveness of stack-propagation compared to traditional types of joint modeling. We also show that our approach leads to large reductions in cascaded errors from the POS tagger.

We hope that this work will motivate further research in combining traditional pipelined structured prediction models with deep neural architectures that learn intermediate representations in a task-driven manner. One important finding of this work is that, even *without* POS tags, our architecture outperforms recurrent approaches that build custom word representations using character-based LSTMs (Ballesteros et al., 2015). These results suggest that learning rich embeddings of words may not be as important as building an intermediate representation that takes multiple features of the *surrounding* context into account. Our results also suggest that deep models for dependency parsing may not discover POS classes when trained solely for parsing, even when it is fully within the capacity of the model. Designing architectures to apply stack-propagation in other coupled NLP tasks might yield significant accuracy improvements for deep learning.

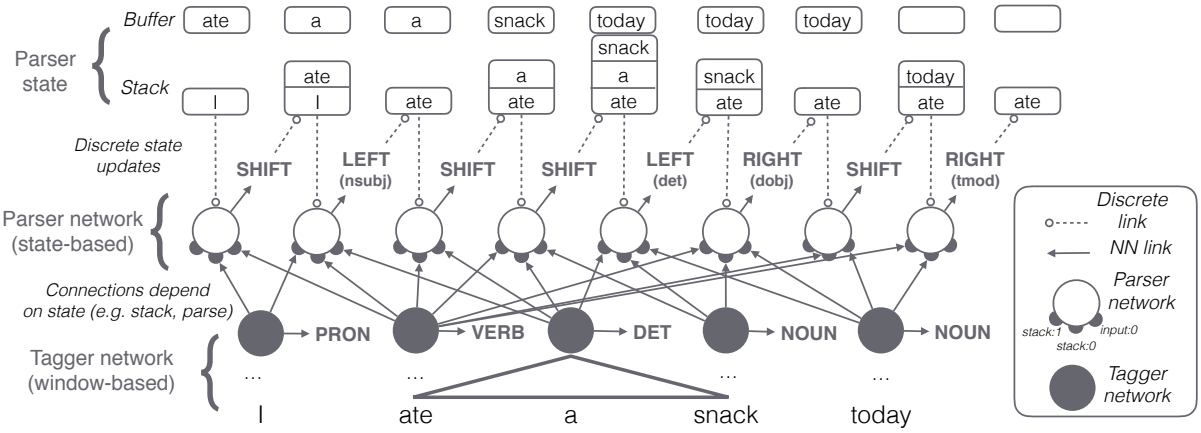


Figure 2: Detailed example of the stacked parsing model. *Top:* The discrete parser state, consisting of the stack and the buffer, is updated by the output of the parser network. In turn, the feature templates used by the parser are a function of the state. In this example, the parser has three templates, `stack:0`, `stack:1`, and `input:0`. *Bottom:* The feature templates create many-to-many connections from the hidden layer of the tagger to the input layer of the parser. For example, the predicted root of the sentence ("ate") is connected to the input of most parse decisions. At test time, the above structure is constructed dynamically as a function of the parser output. Note also that the predicted POS tags are not directly used by the parser.

2 Continuous Stacking Model

In this section, we introduce a novel neural network model for parsing and tagging that incorporates POS tags as a regularization of learned implicit representations. The basic unit of our model (Figure 3) is a simple, feed-forward network that has been shown to work very well for parsing tasks (Chen and Manning, 2014; Weiss et al., 2015). The inputs to this unit are feature matrices which are embedded and passed as input to a hidden layer. The final layer is a softmax prediction.

We use two such networks in this work: a window-based version for tagging and a transition-based version for dependency parsing. In a traditional stacking (pipeline) approach, we would use the discrete predicted POS tags from the tagger as features in the parser (Chen and Manning, 2014). In our model, we instead feed the continuous hidden layer activations of the tagger network as input to the parser. The primary strength of our approach is that the parser has access to all of the features and information used by the POS tagger during training time, but it is allowed to make its own decisions at test time.

To implement this, we show how we can reuse feature templates from Chen and Manning (2014) to specify the feed-forward connections from the tagger network to the parser network. An interesting consequence is that because this structure is a function of the derivation produced by the parser, the final feed-forward structure of the stacked model is not known until *run-time*.

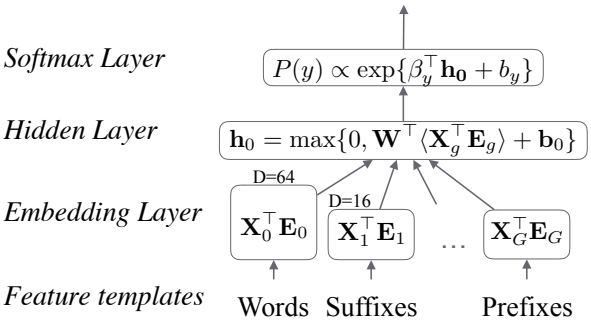
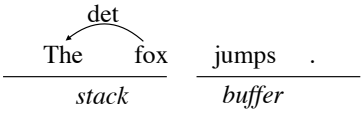


Figure 3: Elementary NN unit used in our model. Feature matrices from multiple channels are embedded, concatenated together, and fed into a rectified linear hidden layer. In the parser network, the feature inputs are continuous representations from the tagger network's hidden layer.

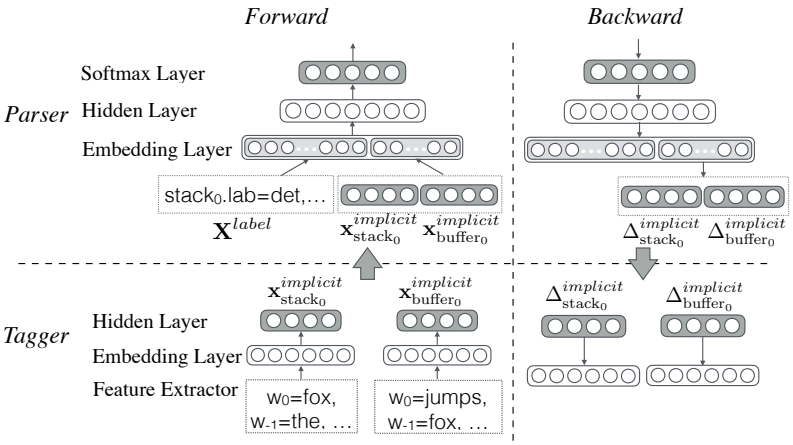
However, because the connections for any specific parsing decision are fixed given the derivation, we can still extract examples for training off-line by unrolling the network structure from gold derivations. In other words, we can utilize our approach with the same simple stochastic optimization techniques used in prior works. Figure 2 shows a fully unrolled architecture on a simple example.

2.1 The Tagger Network

As described above, our POS tagger follows the basic structure from prior work with embedding, hidden, and softmax layers. Like the "window-approach" network of Collobert et al. (2011), the tagger is evaluated per-token, with features extracted from a window of tokens surrounding the target. The input consists of a rich set of fea-



(a) Parser configuration c.



(b) Forward and backward schema for (a).

Figure 4: A schematic for the PARSER stack-propagation update. *a*: Example parser configuration c with corresponding stack and buffer. *b*: Forward and backward stages for the given single example. During the forward phase, the tagger networks compute hidden activations for each feature template (e.g. $stack_0$ and $buffer_0$), and activations are fed as features into the parser network. For the backward update, we backpropagate training signals from the parser network into each linked tagging example.

tures for POS tagging that are deterministically extracted from the training data. As in prior work, the features are divided into *groups* of different sizes that share an embedding matrix $\mathbf{E}$. Features for each group g are represented as a sparse matrix $\mathbf{X}^g$ with dimension $F^g \times V^g$, where F^g is the number of feature templates in the group, and V^g is the vocabulary size of the feature templates. Each row of $\mathbf{X}^g$ is a one-hot vector indicating the appearance of each feature.

The network first looks up the learned embedding vectors for each feature and then concatenates them to form the embedding layer. This embedding layer can be written as:

$$\mathbf{h}_0 = [\mathbf{X}^g \mathbf{E}^g \mid \forall g] \quad (1)$$

where $\mathbf{E}^g$ is a learned $V^g \times D^g$ embedding matrix for feature group. Thus, the final size $|\mathbf{h}_0| = \sum_g F^g D^g$ is the sum of all embedded feature sizes. The specific features and their dimensions used in the tagger are listed in Table 1. Note that for all features, we create additional *null* value that triggers when features are extracted outside the scope of the sentence. We use a single hidden layer in our model and apply rectified linear unit (ReLU) activation function over the hidden layer outputs. A final softmax layer reads in the activations and outputs probabilities for each possible POS tag.

2.2 The Parser Network

The parser component follows the same design as the POS tagger with the exception of the features and the output space. Instead of a window-based classifier, features are extracted from an arc-standard parser configuration[1] c consisting of the stack s, the buffer b and the so far constructed dependencies (Nivre, 2004). Prior implementations of this model used up to four groups of discrete features: words, labels (from previous decisions), POS tags, and morphological attributes (Chen and Manning, 2014; Weiss et al., 2015; Alberti et al., 2015).

Features (g)	Window	D
Symbols	1	8
Capitalization	+/- 1	4
Prefixes/Suffixes ($n = 2, 3$)	+/- 1	16
Words	+/-3	64

Table 1: Window-based tagger feature spaces. "Symbols" indicates whether the word contains a hyphen, a digit or a punctuation.

In this work, we apply the same design principle but we use an *implicitly* learned intermediate representation in the parser to replace traditional discrete features. We only retain discrete features over the labels in the incrementally constructed tree (Figure 4). Specifically, for any token of interest, we feed the hidden layer of the tagger network evaluated for that token as input to the parser. We implement this idea by re-using the feature templates from prior work as indexing functions.

We define this process formally as follows. Let $f_i(c)$ be a function mapping from parser configurations c to indices in the sentence, where i denotes each of our feature templates. For example, in Figure 4(a), when $i = \texttt{stack}_0$, $f_i(c)$ is the in-

[1]Note that the "stack" in the parse configuration is separate from the "stacking" of the POS tagging network and the parser network (Figure 1).

dex of "fox" in the sentence. Let $\mathbf{h}_1^{tagger}(j)$ be the hidden layer activation of the tagger network evaluated at token j. We define the input $\mathbf{X}^{implicit}$ by concatenating these tagger activations according to our feature templates:

$$\mathbf{x}_i^{implicit} \triangleq \mathbf{h}_1^{tagger}(f_i(c)). \quad (2)$$

Thus, the feature group $\mathbf{X}^{implicit}$ is the row-concatenation of the hidden layer activations of the tagger, as indexed by the feature templates. We have that $F^{implicit}$ is the number of feature templates, and $V^{implicit} = H^{tagger}$, the number of possible values is the number of hidden units in the tagger. Just as for other features, we learn an embedding matrix $\mathbf{E}^{implicit}$ of size $H^{implicit} \times F^{implicit}$. Note that as in the POS tagger network, we reserve an additional *null* value for out of scope feature templates. A full example of this lookup process, and the resulting feed-forward network connections created, is shown for a simple three-feature template consisting of the top two tokens on the stack and the first on the buffer in Figure 2. See Table 1 for the full list of 20 tokens that we extract for each state.

3 Learning with Stack-propagation

In this section we describe how we train our stacking architecture. At a high level, we simply apply backpropagation to our proposed continuous form of stacking (hence "stack-propagation.") There are two major issues to address: (1) how to handle the dynamic many-to-many connections between the tagger network and the parser network, and (2) how to incorporate the POS tag labels during training.

Addressing the first point turns out to be fairly easy in practice: we simply unroll the gold trees into a derivation of (state, action) pairs that produce the tree. The key property of our parsing model is that the *connections* of the feed-forward network are constructed incrementally as the parser state is updated. This is different than a generic recurrent model such as an LSTM, which passes activation vectors from one step to the next. The important implication at training time is that, unlike a recurrent network, the parser decisions are conditionally independent given a fixed history. In other words, if we unroll the network structure ahead of time given the gold derivation, we do not need to perform inference when training with respect to these examples. Thus, the overall training procedure is similar to that introduced in Chen and Manning (2014).

To incorporate the POS tags as a regularization during learning, we take a fairly standard approach from multi-task learning. The objective of learning is to find parameters Θ that maximize the data log-likelihood with a regularization on Θ for both parsing and tagging:

$$\max_{\Theta} \lambda \sum_{\mathbf{x},y \in \mathcal{T}} \log(P_\Theta(y \mid \mathbf{x})) + \sum_{c,a \in \mathcal{P}} \log\left(P_\Theta(a \mid c)\right), \quad (3)$$

where $\{\mathbf{x}, y\}$ are POS tagging examples extracted from individual tokens and $\{c, a\}$ are parser (configuration, action) pairs extracted from the unrolled gold parse tree derivations, and λ is a trade-off parameter.

We optimize this objective stochastically by alternating between two updates:

- TAGGER: Pick a POS tagging example and update the tagger network with backpropagation.
- PARSER: (Figure 4) Given a parser configuration c from the set of gold contexts, compute both tagger and parser activations. Backpropagate the parsing loss through the stacked architecture to update both parser and tagger, ignoring the tagger's softmax layer parameters.

While the learning procedure is inspired from multi-task learning—we only update each step with regards one of the two likelihoods—there are subtle differences that are important. While a traditional multi-task learning approach would use the final layer of the parser network to predict both POS tags and parse trees, we predict POS tags from the first hidden layer of our model (the "tagger" network) only. We treat the POS labels as regularization of our parser and simply discard the softmax layer of the tagger network at test time. As we will show in Section 4, this regularization leads to dramatic gains in parsing accuracy. Note that in Section 5, we also show experimentally that stack-propagation is more powerful than the traditional multi-task approach, and by combining them together, we can achieve better accuracy on both POS and parsing tasks.

Method	ar	bg	da	de	en	es	eu	fa	fi	fr	hi	id	it	iw	nl	no	pl	pt	sl	AVG
NO TAGS																				
B'15 LSTM	75.6	**83.1**	69.6	**72.4**	77.9	78.5	67.5	74.7	**73.2**	77.4	85.9	**72.3**	**84.1**	73.1	69.5	82.4	**78.0**	**79.9**	**80.1**	76.6
Ours (window)	**76.1**	82.9	**70.9**	71.7	**79.2**	**79.3**	**69.1**	**77.5**	72.5	**78.2**	**87.1**	71.8	83.6	**76.2**	**72.3**	**83.2**	77.8	79.0	79.8	**77.3**
UNIVERSAL TAGSET																				
B'15 LSTM	74.6	82.4	68.1	73.0	77.9	77.8	66.0	75.0	73.6	78.0	86.8	72.2	84.2	74.5	68.4	83.3	74.5	80.4	78.1	76.2
Pipeline P_{tag}	73.7	83.6	72.0	73.0	79.3	79.5	63.0	78.0	66.9	78.5	87.8	73.5	84.2	75.4	70.3	83.6	73.4	79.5	79.4	76.6
RBGParser	75.8	83.6	**73.9**	73.5	79.9	79.6	68.0	**78.5**	65.4	78.9	87.7	**74.2**	84.7	**77.6**	72.4	83.9	75.4	**81.3**	80.7	77.6
Stackprop	**77.0**	**84.3**	73.8	**74.2**	**80.7**	**80.7**	**70.1**	**78.5**	**74.5**	**80.0**	**88.9**	74.1	**85.8**	77.5	**73.6**	**84.7**	**79.2**	80.4	**81.8**	**78.9**

Table 2: Labeled Attachment Score (LAS) on Universal Dependencies Treebank. *Top:* Results without any POS tag observations. "B'15 LSTM" is the character-based LSTM model (Ballesteros et al., 2015), while "Ours (window)" is our window-based architecture variant without stackprop. *Bottom:* Comparison against state-of-the-art baselines utilizing the POS tags. Paired t-tests show that the gain of Stackprop over all other approaches is significant ($p < 10^{-5}$ for all but RBGParser, which is $p < 0.02$).

3.1 Implementation details

Following Weiss et al. (2015), we use mini-batched averaged stochastic gradient descent (ASGD) (Bottou, 2010) with momentum (Hinton, 2012) to learn the parameters Θ of the network. We use a separate learning rate, moving average, and velocity for the tagger network and the parser; the PARSER updates all averages, velocities, and learning rates, while the TAGGER updates only the tagging factors. We tuned the hyperparameters of momentum rate μ, the initial learning rate η_0 and the learning rate decay step γ using held-out data. The training data for parsing and tagging can be extracted from either the same corpus or different corpora; in our experiments they were always the same.

To trade-off the two objectives, we used a random sampling scheme to perform 10 epochs of PARSER updates and 5 epochs of TAGGER updates. In our experiments, we found that pre-training with TAGGER updates for one epoch before interleaving PARSER updates yielded faster training with better results. We also experimented using the TAGGER updates solely for initializing the parser and found that interleaving updates was crucial to obtain improvements over the baseline.

4 Experiments

In this section, we evaluate our approach on several dependency parsing tasks across a wide variety of languages.

4.1 Experimental Setup

We first investigated our model on 19 languages from the Universal Dependencies Treebanks v1.2.[2] We selected the 19 largest currently spoken languages for which the full data was freely available. We used the coarse universal tagset in our experiments with no explicit morphological annotations. To measure parsing accuracy, we report unlabeled attachment score (UAS) and labeled attachment score (LAS) computed on all tokens (including punctuation), as is standard for non-English datasets.

For simplicity, we use the arc-standard (Nivre, 2004) transition system with greedy decoding. Because this transition system only produces projective trees, we first apply a projectivization step to all treebanks before unrolling the gold derivations during training. We make an exception for Dutch, where we observed a significant gain on development data by introducing the SWAP action (Nivre, 2009) and allowing non-projective trees.

For models that required predicted POS tags, we trained a window-based tagger using the same features as the tagger component of our stacking model. We used 5-fold jackknifing to produce predicted tags on the training set. We found that the window-based tagger was comparable to a state-of-the-art CRF tagger for most languages. For every network we trained, we used the development data to evaluate a small range of hyperparameters, stopping training early when UAS no longer improved on the held-out data. We use $H = 1024$ hidden units in the parser, and $H = 128$ hidden units in the tagger. The parser embeds the tagger activations with $D = 64$. Note that following Ballesteros et al. (2015), we did not use any auxiliary data beyond that in the treebanks, such as pre-trained word embeddings.

For a final set of experiments, we evaluated on the standard Wall Street Journal (WSJ) part of the Penn Treebank (Marcus et al., 1993)), dependencies generated from version 3.3.0 of the Stanford

[2] http://universaldependencies.org

Method	UAS	LAS
NO TAGS		
Dyer et al. (2015)	92.70	90.30
Ours (window-based)	**92.85**	**90.77**
UNIVERSAL TAGSET		
Pipeline (P_{tag})	92.52	90.50
Stackprop	**93.23**	**91.30**
FINE TAGSET		
Chen & Manning (2014)	91.80	89.60
Dyer et al. (2015)	93.10	90.90
Pipeline (P_{tag})	93.10	91.16
Stackprop	**93.43**	**91.41**
Weiss et al. (2015)	93.99	92.05
Alberti et al. (2015)	94.23	92.36

Table 3: WSJ Test set results for greedy and state-of-the-art methods. For reference, we show the most accurate models from Alberti et al. (2015) and Weiss et al. (2015), which use a deeper model and beam search for inference.

converter (De Marneffe et al., 2006). We followed standard practice and used sections 2-21 for training, section 22 for development, and section 23 for testing. Following Weiss et al. (2015), we used section 24 to tune any hyperparameters of the model to avoid overfitting to the development set. As is common practice, we use pretrained word embeddings from the `word2vec` package when training on this dataset.

4.2 Results

We present our main results on the Universal Treebanks in Table 2. We directly compare our approach to other baselines in two primary ways. First, we compare the effectiveness of our learned continuous representations with those of Alberti et al. (2015), who use the predicted distribution over POS tags concatenated with word embeddings as input to the parser. Because they also incorporate beam search into training, we re-implement a greedy version of their method to allow for direct comparisons of token representations. We refer to this as the "Pipeline (P_{tag})" baseline. Second, we also compare our architecture trained without POS tags as regularization, which we refer to as "Ours (window-based)". This model has the same architecture as our full model but with no POS supervision and updates. Since this model never observes POS tags in any way, we compare against a recurrent character-based parser (Ballesteros et al., 2015) which is state-of-the-art when no POS tags are provided.[3] Finally, we compare to RGBParser (Lei et al., 2014), a state-of-the art graph-based (non-greedy) approach.

Model Variant	UAS	LAS	POS
Arc-standard transition system			
Pipeline (P_{tag})	81.56	76.55	95.14
Ours (window-based)	82.08	77.08	-
Ours (Stackprop)	**83.38**	**78.78**	-
Joint parsing & tagging transition system			
Pipeline (P_{tag})	81.61	76.57	95.30
Ours (window-based)	82.58	77.76	94.92
Ours (Stackprop)	**83.21**	**78.64**	**95.43**

Table 4: Averaged parsing and POS tagging results on the UD treebanks for joint variants of stackprop. Given the window-based architecture, stackprop leads to higher parsing accuracies than joint modeling (83.38% vs. 82.58%).

Our greedy stackprop model outperforms all other methods, including the graph-based RBGParser, by a significant margin on the test set (78.9% vs 77.6%). This is despite the limitations of greedy parsing. Stackprop also yields a 2.3% absolute improvement in accuracy compared to using POS tag confidences as features (Pipeline P_{tag}). Finally, we also note that adding stackprop to our window-based model improves accuracy in *every* language, while incorporating predicted POS tags into the LSTM baseline leads to occasional drops in accuracy (most likely due to cascaded errors.)

5 Discussion

Stackprop vs. other representations. One unexpected result was that, even *without* the POS tag labels at training time, our stackprop architecture achieves better accuracy than either the character-based LSTM or the pipelined baselines (Table 2). This suggests that adding window-based representations–which aggregate over many features of the word and surrounding context–is more effective than increasing the expressiveness of individual word representations by using character-based recurrent models. In future work we will explore combining these two complementary approaches.

We hypothesized that stackprop might provide larger gains over the pipelined model when the

[3]We thank Ballesteros et al. (2015) for their assistance running their code on the treebanks.

Token	married by a **judge**.	Don't **judge** a book by	and **walked** away satisfied	when I **walk** in the door
Neighbors	mesmerizing as a *rat*. A *staple!* day at a *bar*, then go	doesn't *change* the company's won't *charge* your phone don't *waste* your money	tried, and *tried* hard and *incorporated* into and *belonged* to the	upset when I *went* to I *mean* besides me I *felt* as if I
Pattern	a [noun]	'nt [verb]	and [verb]ed	I [verb]

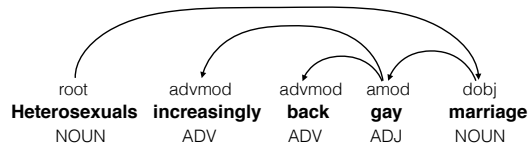
Table 5: Four of examples of tokens in context, along with the three most similar tokens according to the tagger network's activations, and the simple pattern exhibited. Note that this model was trained with the Universal tagset which does not distinguish verb tense.

POS tags are very coarse. We tested this latter hypothesis on the WSJ corpus by training our model using the coarse universal tagsets instead of the fine tagset (Table 3). We found that stackprop achieves similar accuracy using coarse tagsets as the fine tagset, while the pipelined baseline's performance drops dramatically. And while stackprop doesn't achieve the highest reported accuracies on the WSJ, it does achieve competitive accuracies and outperforms prior state-of-the-art for greedy methods (Dyer et al., 2015).

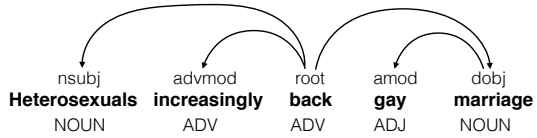
Stackprop vs. joint modeling. An alternative to stackprop would be to train the *final* layer of our architecture to predict both POS tags and dependency arcs. To evaluate this, we trained our window-based architecture with the integrated transition system of Bohnet and Nivre (2012), which augments the SHIFT transition to predict POS tags. Note that if we also apply stackprop, the network learns from POS annotations twice: once in the TAGGER updates, and again the PARSER updates. We therefore evaluated our window-based model both with and without stack-propagation, and with and without the joint transition system.

We compare these variants along with our reimplementation of the pipelined model of Alberti et al. (2015) in Table 4. We find that stackprop is always better, even when it leads to "double counting" the POS annotations; in this case, the result is a model that is significantly better at POS tagging while marginally worse at parsing than stackprop alone.

Reducing cascaded errors. As expected, we observe a significant reduction in cascaded POS tagging errors. An example from the English UD treebank is given in Figure 5. Across the 19 languages in our test set, we observed a 10.9% gain (34.1% vs. 45.0%) in LAS on tokens where the pipelined POS tagger makes a mistake, compared to a 1.8% gain on the rest of the corpora.



(a) Tree by a pipeline model.



(b) Tree by Stackprop model.

Figure 5: Example comparison between predictions by a pipeline model and a joint model. While both models predict a wrong POS tag for the word "back" (ADV rather than VERB), the joint model is robust to this POS error and predict the correct parse tree.

Decreased model size. Previous neural parsers that use POS tags require learning embeddings for words and other features on top of the parameters used in the POS tagger (Chen and Manning, 2014; Weiss et al., 2015). In contrast, the number of total parameters for the combined parser and tagger in the Stackprop model is reduced almost by half compared to the Pipeline model, because the parser and tagger share parameters. Furthermore, compared to our implementation of the pipeline model, we observed that this more compact parser model was also roughly twice as fast.

Contextual embeddings. Finally, we also explored the significance of the representations learned by the tagger. Unlike word embedding models, the representations used in our parser are constructed for each token based on its surrounding context. We demonstrate a few interesting trends we observed in Table 5, where we show the nearest neighbors to sample tokens in this contextual embedding space. These representations tend to represent syntactic patterns rather than individual words, distinguishing between the form (e.g. "judge" as a noun vs. a verb') and context of tokens (e.g. preceded by a personal pronoun).

6 Conclusions

We present a stacking neural network model for dependency parsing and tagging. Through a simple learning method we call "stack-propagation," our model learns effective intermediate representations for parsing by using POS tags as regularization of implicit representations. Our model outperforms all state-of-the-art parsers when evaluated on 19 languages of the Universal Dependencies treebank and outperforms other greedy models on the Wall Street Journal.

We observe that the ideas presented in this work can also be as a principled way to optimize upstream NLP components for down-stream applications. In future work, we will extend this idea beyond sequence modeling to improve models in NLP that utilize parse trees as features. The basic tenet of stack-propagation is that the hidden layers of neural models used to generate annotations can be used instead of the annotations themselves. This suggests a new methodology to building deep neural models for NLP: we can design them from the ground up to incorporate multiple sources of annotation and learn far more effective intermediate representations.

Acknowledgments

We would like to thank Ryan McDonald, Emily Pitler, Chris Alberti, Michael Collins, and Slav Petrov for their repeated discussions, suggestions, and feedback, as well all members of the Google NLP Parsing Team. We would also like to thank Miguel Ballesteros for assistance running the character-based LSTM.

References

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of EMNLP 2015*.

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

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465. Association for Computational Linguistics.

Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, volume 1, pages 740–750.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of Fifth International Conference on Language Resources and Evaluation*, pages 449–454.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the Association for Computational Linguistics*. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *ACL*.

Nicholas FitzGerald, Oscar Tckstrm, Kuzman Ganchev, and Dipanjan Das. 2015. Semantic role labeling with neural network factors. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP '15)*.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *IJCNLP*, pages 1216–1224. Citeseer.

Geoffrey E Hinton. 2012. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade*, pages 599–619. Springer.

Terry Koo, Xavier Carreras Pérez, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *46th Annual Meeting of the Association for Computational Linguistics*, pages 595–603.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1381–1391.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, Wenliang Chen, and Haizhou Li. 2011. Joint models for chinese pos tagging and dependency parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1180–1191. Association for Computational Linguistics.

Zhenghua Li, Min Zhang, Wanxiang Che, Ting Liu, and Wenliang Chen. 2014. Joint optimization for chinese POS tagging and dependency parsing. *IEEE/ACM Transactions on Audio, Speech & Language Processing*, pages 274–286.

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

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Proceedings of the Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, IncrementParsing '04, pages 50–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2011. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*.

Xian Qian and Yang Liu. 2012. Joint chinese word segmentation, pos tagging and parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 501–511. Association for Computational Linguistics.

Zhiguo Wang and Nianwen Xue. 2014. Joint pos tagging and transition-based constituent parsing in chinese with non-local features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 733–742.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of ACL 2015*, pages 323–333.

David H Wolpert. 1992. Stacked generalization. *Neural networks*.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 42–52.

Inferring Perceived Demographics from User Emotional Tone and User-Environment Emotional Contrast

Svitlana Volkova
Johns Hopkins University
(now at Pacific Northwest National Laboratory)
Baltimore, MD, 21218, USA
svitlana@jhu.edu

Yoram Bachrach
Microsoft Research
Cambridge, UK CB1 2FB
yobach@microsoft.com

Abstract

We examine communications in a social network to study user emotional contrast – the propensity of users to express different emotions than those expressed by their neighbors. Our analysis is based on a large Twitter dataset, consisting of the tweets of 123,513 users from the USA and Canada. Focusing on Ekman's basic emotions, we analyze differences between the emotional tone expressed by these users and their neighbors of different types, and correlate these differences with perceived user demographics. We demonstrate that many perceived demographic traits correlate with the emotional contrast between users and their neighbors. Unlike other approaches on inferring user attributes that rely solely on user communications, we explore the network structure and show that it is possible to accurately predict a range of perceived demographic traits based solely on the emotions emanating from users and their neighbors.

1 Introduction

The explosion of social media services like Twitter, Google+ and Facebook have led to a growing application potential for personalization in human computer systems such as personalized intelligent user interfaces, recommendation systems, and targeted advertising. Researchers have started mining these massive volumes of personalized and diverse data produced in public social media with the goal of learning about their demographics (Burger et al., 2011; Zamal et al., 2012; Volkova et al., 2015) and personality (Golbeck et al., 2011; Kosinski et al., 2013),[1] language variation (Eisenstein et al., 2014; Kern et al., 2014; Bamman et al., 2014),[2] likes and interests (Bachrach et al., 2012; Lewenberg et al., 2015), emotions and opinions they express (Bollen et al., 2011b; Volkova and Bachrach, 2015), their well-being (Schwartz et al., 2013) and their interactions with online environment (Bachrach, 2015; Kalaitzis et al., 2016). The recent study has shown that the environment in a social network has a huge influence on user behavior and the tone of the messages users generate (Coviello et al., 2014; Ferrara and Yang, 2015a).

People vary in the ways they respond to the emotional tone of their environment in a social network. Some people tend to send out messages with a positive emotional tone, while others tend to express more negative emotions such as sadness or fear. Some of us are likely to share peer messages that are angry, whereas others filter out such messages. In this work we focus on the problem of predicting user *perceived demographics* by examining the emotions expressed by users and their immediate neighbors. We first define the user emotional tone, the environment emotional tone, and the user-environment emotional contrast.

Definition 1 Environment emotional tone *is the proportion of tweets with a specific emotion produced by the user's neighbors. For example, if the majority of tweets sent by the user's neighbors express joy, that user has a positive environment. In contrast, a user is in a negative environment if most of his or her neighbors express anger.*

Definition 2 User emotional tone *is the proportion of tweets with a specific emotion produced by a user. If a user mostly sends sad messages, he generates a sad emotional tone, while a user who mostly sends joyful messages has a joyful tone.*

[1] https://apps.facebook.com/snpredictionapp/

[2] http://demographicvis.uncc.edu/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1567–1578,
Berlin, Germany, August 7-12, 2016.

Definition 3 User-environment emotional contrast *is a degree to which user emotions differ from the emotions expressed by user neighbors. We say that users express more of an emotion when they express it more frequently than their neighbors, and say they express less of an emotion when they express it less frequently than their environment.*

There are two research questions we address in this work. First, we analyze how user demographic traits are predictive of the way they respond to the emotional tone of their environment in a social network. One hypothesis stipulates that the emotional response is a *universal* human trait, regardless of the specific demographic background (Wierzbicka, 1986; Cuddy et al., 2009). For example, men and women or young and old people should not be different in the way they respond to their emotional environment. An opposite hypothesis is a *demographic dependent emotional contrast hypothesis*, stipulating that user demographic background is predictive of the emotional contrast with the environment. For example, one might expect users with lower income to express negative emotion even when their environment expresses mostly positive emotions (high degree of emotional contrast), while users with higher income are more likely to express joy even if their environment expresses negative emotions (Kahneman and Deaton, 2010).

We provide an empirical analysis based on a large dataset sampled from a Twitter network, supporting the *demographic dependent emotional contrast hypothesis*. We show that users predicted to be younger, without kids and with lower income tend to express more sadness compared to their neighbors but older users, with kids and higher income express less; users satisfied with life express less anger whereas users dissatisfied with life express more anger compared to their neighbors; optimists express more joy compared to their environment whereas pessimists express less.

Furthermore, we investigate whether user demographic traits can be predicted from user emotions and user-environment emotional contrast. Earlier work on inferring user demographics has examined methods that use lexical features in social networks to predict demographic traits of the author (Burger et al., 2011; Van Durme, 2012; Conover et al., 2011; Bergsma et al., 2013; Bamman et al., 2014; Ruths et al., 2014; Sap et al., 2014). However, these are simply features of the text a user produces, and make limited use of the social embedding of the user in the network. Only limited amount of work briefly explored the network structure for user profiling (Pennacchiotti and Popescu, 2011a; Filippova, 2012; Zamal et al., 2012; Volkova et al., 2014; Culotta et al., 2015). In contrast, we investigate the predictive value of features that are completely dependent on the network: the emotional contrast between users and their neighbors. We also combine network (context) and text (content) features to further boost the performance of our models.

Our results show that the emotional contrast of users is very informative regarding their demographic traits. Even a very small set of features consisting of the emotional contrast between users and their environment for each of Ekman's six basic emotions and three sentiment types is sufficient to obtain high quality predictions for a range of user attributes.

Carrying out such an analysis requires using a large dataset consisting of many users annotated with a variety of properties, and a large pool of their communications annotated with emotions and sentiments. Creating such a large dataset with the ground truth annotations is extremely costly; user sensitive demographics e.g., income, age is not available for the majority of social media including Twitter. Therefore, we rely our analysis on a large Twitter dataset annotated with demographics and affects using predictive models that can accurately infer user attributes, emotions and sentiments as discussed in Section 3.

2 Data

User-Neighbor Dataset For the main analysis we collected a sample of $U = 10,741$ Twitter users and randomly sampled their neighbors $n \in N^{(u)}$ of different types including friends – u follows $n^{(u)}$, mentions – u mentions $n^{(u)}$ in his or her tweets *e.g., @modollar1*, and retweets – u retweets $n^{(u)}$ tweets *e.g., RT @GYPSY*. In total we sampled N= $141,034$ neighbors for U=$10,741$

Relation	$\subseteq U$	N_{uniq}	N_{all}	T_{total}
Retweet R	9,751	32,197	48,262	6,345,722
Mention M	9,251	37,199	41,456	7,634,961
Friend F	10,381	43,376	51,316	8,973,783
TOTAL	**10,741**	**112,772**	**141,034**	**24,919,528**

Table 1: Twitter ego-network sample stats: U=$123,513$ unique users with T=$24,919,528$ tweets, and E=$141,034$ edges that represent social relations between Twitter users.

users; on average 15 neighbors per user, 5 neighbors of each type with their 200 tweets; in total T=24,919,528 tweets as reported in Table 1. We also report the number of users with at least one neighbor of each type $\subseteq U$ and the number of unique neighbors N_{uniq}.[3]

Dataset Annotated with Demographics Unlike Facebook (Bachrach et al., 2012; Kosinski et al., 2013), Twitter profiles do not have personal information attached to the profile e.g., gender, age, education. Collecting self-reports (Burger et al., 2011; Zamal et al., 2012) brings data sampling biases which makes the models trained on self-reported data unusable for predictions of random Twitter users (Cohen and Ruths, 2013; Volkova et al., 2014). Asking social media users to fill personality questionnaires (Kosinski et al., 2013; Schwartz et al., 2013) is time consuming. An alternative way to collect attribute annotations is through crowdsourcing as has been effectively done recently (Flekova et al., 2015; Sloan et al., 2015; Preoiuc-Pietro et al., 2015).

Thus, to infer sociodemographic traits for a large set of random Twitter users in our dataset we relied on pre-trained models learned from 5,000 user profiles annotated via crowdsourcing[4] released by Volkova and Bachrach (2015). We annotated 125,513 user and neighbor profiles with eight sociodemographic traits. We only used a subset of sociodemographic traits from their original study to rely our analysis on models trained on annotations with high or moderate inter-annotator agreement. Additionally, we validated the models learned from the crowdsourced annotations on several public datasets labeled with gender as described in Section 2. Table 2 reports attribute class distributions and the number of profiles annotated.

Validating Crowdsourced Annotations To validate the quality of perceived annotations we applied 4,998 user profiles to classify users from the existing datasets annotated with gender using approaches other than crowdsourcing. We ran experiments across three datasets (including perceived annotations): Burger et al.'s data (Burger et al., 2011) – 71,312 users, gender labels were obtained via URL following users' personal blogs; Zamal et al.'s data (Zamal et al., 2012) – 383 users, gender labels were collected via user names. Table 3 presents a cross-dataset comparison results.

Attribute	Class Distribution	Profiles
Age	$\leq$ 25 y.o. (65%), $>$ 25 y.o.	3,883
Children	No (84%), Yes	5,000
Education	High School (68%), Degree	4,998
Ethnicity	Caucasian (59%), Afr. Amer.	4,114
Gender	Female (58%), Male	4,998
Income	$\leq$ \$35K (66%), $>$ \$35K	4,999
Life Satisf.	Satisfied (78%), Dissatisfied	3,789
Optimism	Optimist (75%), Pessimist	3,562

Table 2: Annotation statistics of perceived user properties from Volkova and Bachrach (2015).

We consistently used logistic regression with L2 regularization and relied on word ngram features similar to Volkova and Bachrach (2015). Accuracies on a diagonal are obtained using 10-fold cross-validation. These results show that textual classifiers trained on perceived annotations have a reasonable agreement with the alternative prediction approaches. This provides another indication that the quality of crowdsourced annotations, at least for gender, is acceptable. There are no publicly available datasets annotated with other attributes from Table 2, so we cannot provide a similar comparison for other traits.

Train\Test	Users	Burger	Zamal	Perceived
Burger	71,312	**0.71**	0.71	0.83
Zamal	383	0.47	**0.79**	0.53
Perceived	4,998	0.58	0.66	**0.84**

Table 3: Cross-dataset accuracy for gender prediction on Twitter.

Sentiment Dataset Our sentiment analysis dataset consists of seven publicly available Twitter sentiment datasets described in detail by Hassan Saif, Miriam Fernandez and Alani (2013). It includes $T_S^L = 19,555$ tweets total (35% positive, 30% negative and 35% neutral) from Stanford,[5] Sanders,[6] SemEval-2013,[7] JHU CLSP,[8] SentiStrength,[9] Obama-McCain Debate and Health Care.[10]

Emotion Dataset We collected our emotion dataset by bootstrapping noisy hashtag annota-

[3] Despite the fact that we randomly sample user neighbors, there still might be an overlap between user neighborhoods dictated by the Twitter network design. Users can be reweeted or mentioned if they are in the friend neighborhood $R \subset F, M \subset F$.

[4] Data collection and perceived attribute annotation details are discussed in (Volkova and Bachrach, 2015) and (Preoiuc-Pietro et al., 2015).

[5] http://help.sentiment140.com

[6] http://www.sananalytics.com/lab/twitter-sentiment/

[7] http://www.cs.york.ac.uk/semeval-2013/task2/

[8] http://www.cs.jhu.edu/∼svitlana/

[9] http://sentistrength.wlv.ac.uk/

[10] https://bitbucket.org/speriosu/updown/

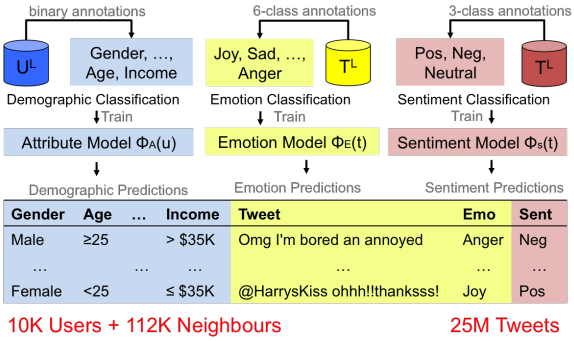


Figure 1: Our approach for predicting user perceived sociodemographics and affects on Twitter.

tions for six basic emotions argued by Ekman[11] as have been successfully done before (De Choudhury et al., 2012; Mohammad and Kiritchenko, 2014). Despite the existing approaches do not disambiguate sarcastic hashtags e.g., *It's Monday #joy* vs. *It's Friday #joy*, they still demonstrate that a hashtag is a reasonable representation of real feelings (González-Ibáñez et al., 2011). Moreover, in this work we relied on emotion hashtag synonyms collected from WordNet-Affect (Valitutti, 2004), GoogleSyns and Roget's thesaurus to overweight the sarcasm factor. Overall, we collected $T_E^L = 52,925$ tweets annotated with anger (9.4%), joy (29.3%), fear (17.1%), sadness (7.9%), disgust (24.5%) and surprise (15.6%).

3 Methodology

Annotating User-Neighbor Data with Sociodemographics and Affects As shown in Figure 1, to perform our analysis we developed three machine learning components. The first component is a user-level demographic classifier $\Phi_A(u)$, which can examine a set of tweets produced by any Twitter user and output a set of predicted demographic traits for that user, including age, education etc. Each demographic classifier relies on features extracted from user content. The second and third components are tweet-level emotion and sentiment classifiers $\Phi_E(t)$ and $\Phi_S(t)$, which can examine any tweet to predict the emotion and sentiment expressed in the tweet.

For inferring user demographics, emotions and sentiments we trained log-linear models with L2 regularization using scikit-learn.[12] Our models rely on word ngram features extracted from user or neighbor tweets and affect-specific features described below.

Perceived Attribute Classification Quality In Section 2 we compared attribute prediction models trained on crowdsourced data vs. other datasets. We showed that models learned from perceived annotations yield higher or comparable performance using the same features and learning algorithms. Given Twitter data sharing restriction,[13] we could only make an indirect comparison with other existing approaches. We found that our models report higher accuracy compared to the existing approaches for gender: +0.12 (Rao et al., 2010), +0.04 (Zamal et al., 2012); and ethnicity: +0.08 (Bergsma et al., 2013), +0.15 (Pennacchiotti and Popescu, 2011b).[14] For previously unexplored attributes we present the ROC AUC numbers obtained using our log-linear models trained on lexical features estimated using 10-fold c.v. in Table 6.

Affect Classification Quality For emotion and opinion classification we trained tweet-level classifiers using lexical features extracted from tweets annotated with sentiments and six basic emotions. In addition to lexical features we extracted a set of *stylistic features including emoticons, elongated words, capitalization, repeated punctuation, number of hashtags and took into account the clause-level negation* (Pang et al., 2002). Unlike other approaches (Wang and Manning, 2012), we observed that adding other linguistic features e.g., higher order ngrams, part-of-speech tags or lexicons did not improve classification performance. We demonstrate our emotion model prediction quality using 10-fold c.v. on our hashtag emotion dataset and compare it to other existing datasets in Table 4. Our results significantly outperform the existing approaches and are comparable with the state-of-the-art system for Twitter sentiment classification (Mohammad et al., 2013; Zhu et al., 2014) (evaluated on the official SemEval-2013 test set our system yields F1 as high as 0.66).

Correlating User-Environment Emotional Contract and Demographics We performed

[11] We prefer Ekman's emotion classification over others e.g., Plutchik's because we would like to compare the performance of our predictive models to other systems.

[12] Scikit-learn toolkit: http://scikit-learn.org/stable/ Email svitlana.volkova@pnnl.gov to get access to pre-trained scikit-learn models and the data.

[13] Twitter policy restricts to sharing only tweet IDs or user IDs rather than complete tweets or user profiles. Thus, some profiles may become private or get deleted over time.

[14] Other existing work on inferring user attributes rely on classification with different categories or use regression e.g., age (Nguyen et al., 2011), income (Preoiuc-Pietro et al., 2015), and education (Li et al., 2014).

#Emotion	Wang (2012)		Roberts (2012)		Qadir (2013)		Mohammad (2014)		This work	
#anger	457,972	0.72	583	0.64	400	0.44	1,555	0.28	4,963	0.80
#disgust	–	–	922	0.67	–	–	761	0.19	12,948	0.92
#fear	11,156	0.44	222	0.74	592	0.54	2,816	0.51	9,097	0.77
#joy	567,487	0.72	716	0.68	1,005	0.59	8,240	0.62	15,559	0.79
#sadness	489,831	0.65	493	0.69	560	0.46	3,830	0.39	4,232	0.62
#surprise	1,991	0.14	324	0.61	–	–	3849	0.45	8,244	0.64
ALL:	**1,991,184**	–	**3,777**	**0.67**	**4,500**	**0.53**	**21,051**	**0.49**	**52,925**	**0.78**

Table 4: Emotion classification results (one vs. all for each emotion and 6 way for ALL) using our models compared to others.

our *user-environment emotional contrast analysis* on a set of users U and neighbors N, where $N^{(u)}$ are the neighbors of u. For each user we defined a set of incoming T^{in} and outgoing T^{out} tweets. We then classified T^{in} and T^{out} tweets containing a sentiment $s \in S$ or emotion $e \in E$, e.g. T^{in}_e, T^{out}_e and T^{in}_s, T^{out}_s where $E \rightarrow$*{anger, joy, fear, surprise, disgust, sad}* and $S \rightarrow$ *{positive, negative, neutral}*.

We measured the proportion of user's incoming and outgoing tweets containing a certain emotion or sentiment e.g., $p^{in}_{sad} = |T^{in}_{sad}|/|T^{in}|$. Then, for every user we estimated *user-environment emotional contrast* using the normalized difference between the incoming p^{in}_e and outgoing p^{out}_e emotion and sentiment proportions:

$$\Delta e = \frac{p^{out}_e - p^{in}_e}{p^{out}_e + p^{in}_e}, \forall e \in E. \quad (1)$$

We estimated *user environment emotional tone* and *user emotional tone* from the distributions over the incoming and outgoing affects e.g., $D^{in}_s = \{p^{in}_{pos}, \ldots, p^{in}_{neut}\}$ and $D^{in}_e = \{p^{in}_{joy}, \ldots, p^{in}_{fear}\}$. We evaluated *user environment emotional tone* – proportions of incoming emotions D^{in}_e and sentiments D^{in}_s on a combined set of friend, mentioned and retweeted users; and *user emotional tone* – proportions of outgoing emotions D^{out}_e and sentiment, D^{out}_s from user tweets. We measure similarity between *user emotional tone* and *environment emotional tone* via Jensen Shannon Divergence (JSD). It is a symmetric and finite KL divergence that measures the difference between two probability distributions.

$$\text{JSD}(D^{in}||D^{out}) = \frac{1}{2}I(D^{in}||D) + \frac{1}{2}I(D^{out}||D), \quad (2)$$

$$\text{where } D = \frac{1}{2}I(D^{in}||D^{out}), I = \sum_e D^{in} \ln \frac{D^{in}}{D^{out}}.$$

Next, we compared emotion and sentiment differences for the groups of users with different demographics $A = \{a_0; a_1\}$ e.g., a_0 = Male and a_1 = Female using a non-parametric Mann-Whitney U test. For example, we measured the means $\mu^{Male}_{\Delta e=joy}$ and $\mu^{Female}_{\Delta e=joy}$ within the group of users predicted to be Males or Females, and estimated whether these means are statistically significantly different. Finally, we used logistic regression to infer a variety of attributes for $U = 10,741$ users using different features below:

- outgoing emotional tone p^{out}_e, p^{out}_s – the overall emotional profile of a user (regardless the emotions projected in his environment);
- user-environment emotional contrast $\Delta e, \Delta s$ – show whether a certain emotion Δe or sentiment Δs is being expressed more or less by the user given the emotions he has been exposed to within his social environment;
- lexical features extracted from user content – represent the distribution of word unigrams over the vocabulary.

4 Experimental Results

For sake of brevity we will refer to a user *predicted* to be male as a male, and a tweet predicted to contain surprise as a simply containing surprise. Despite this needed shorthand it is important to recall that a major contribution of this work is that these results are based on *automatically predicted* properties, as compared to ground truth. We argue here that while such automatically predicted annotations may be less than perfect at the individual user or tweet level, they provide for meaningful analysis when done on the aggregate.

4.1 Similarity between User and Environment Emotional Tones

We report similarities between *user emotional tone* and *environment emotional tone* for different groups of Twitter users using Jensen Shannon Divergence defined in the Eq. 2. We present the mean JSD values estimated over users with two contrasting attributes e.g., predicted to be a_0=Male vs. a_1=Female in Table 5.

	Sentiment Similarities						Emotion Similarities					
Attribute $[a_0, a_1]$	**Retweet**		**Friend**		**All**		**Retweet**		**Friend**		**All**	
Income [$\geq$ \$35K, $<$ \$35K]	22.1	19.4	23.7	21.1	18.6	15.1	18.7	17.8	33.6	33.3	20.0	17.6
Age [$<$ 25 y.o, $\geq$ 25 y.o.]	19.0	22.7	20.2	25.3	14.3	19.7	17.2	19.9	32.8	34.7	17.0	21.1
Education [School, Degree]	19.4	22.1	21.1	23.8	15.2	18.5	18.0	18.1	33.9	32.1	18.1	18.9
Children [Yes, No]	24.2	19.9	28.4	21.4	23.2	15.6	20.9	17.8	35.6	33.2	22.6	18.0
Gender [Male, Female]	19.7	20.5	22.0	21.9	16.5	15.9	18.3	17.9	31.6	34.6	18.2	18.5
Ethnicity [Caucas., Afr. American]	20.5	19.4	21.7	22.5	15.8	16.9	17.2	19.8	32.5	35.2	17.5	20.1
Optimism [Pessimist, Optimist]	19.9	20.3	23.1	21.7	16.8	16.0	18.9	17.9	33.6	33.3	18.6	18.3
Life Satisfaction [Dissatis., Satisfied]	19.4	20.3	21.6	22.0	15.3	16.3	18.6	18.0	33.1	33.4	18.5	16.5

Table 5: Mean Jensen Shannon Divergences (displayed as percentages) between the incoming D^{in} and outgoing D^{out} affects for contrastive attribute values a_0 and a_1. MannWhitney test results for differences between a_0 and a_1 JSD values are shown in blue (p-value ≤ 0.01), green (p-value ≤ 0.05), and gray (p-value ≤ 0.1).

In Table 5 *user environment emotional tones* are estimated over different user-neighbor environments e.g., retweet, friend, and all neighborhoods including user mentions. We found that if *user environment emotional tones* are estimated from mentioned or retweeted neighbors the JSD values are lower compared to the friend neighbors. It means that users are more emotionally similar to the users they mention or retweet than to their friends (users they follow).

We show that *user incoming and outgoing sentiment tones* D_s^{in} and D_s^{out} estimated over all neighbors are significantly different for the majority of attributes except ethnicity. The divergences are consistently pronounced across all neighborhoods for income, age, education, optimism and children attributes (p-value ≤ 0.01). When the *incoming and outgoing emotional tones* D_e^{in} and D_e^{out} are estimated over all neighbors, they are significantly different for all attributes except education and life satisfaction.

4.2 User-Environment Affect Contrast

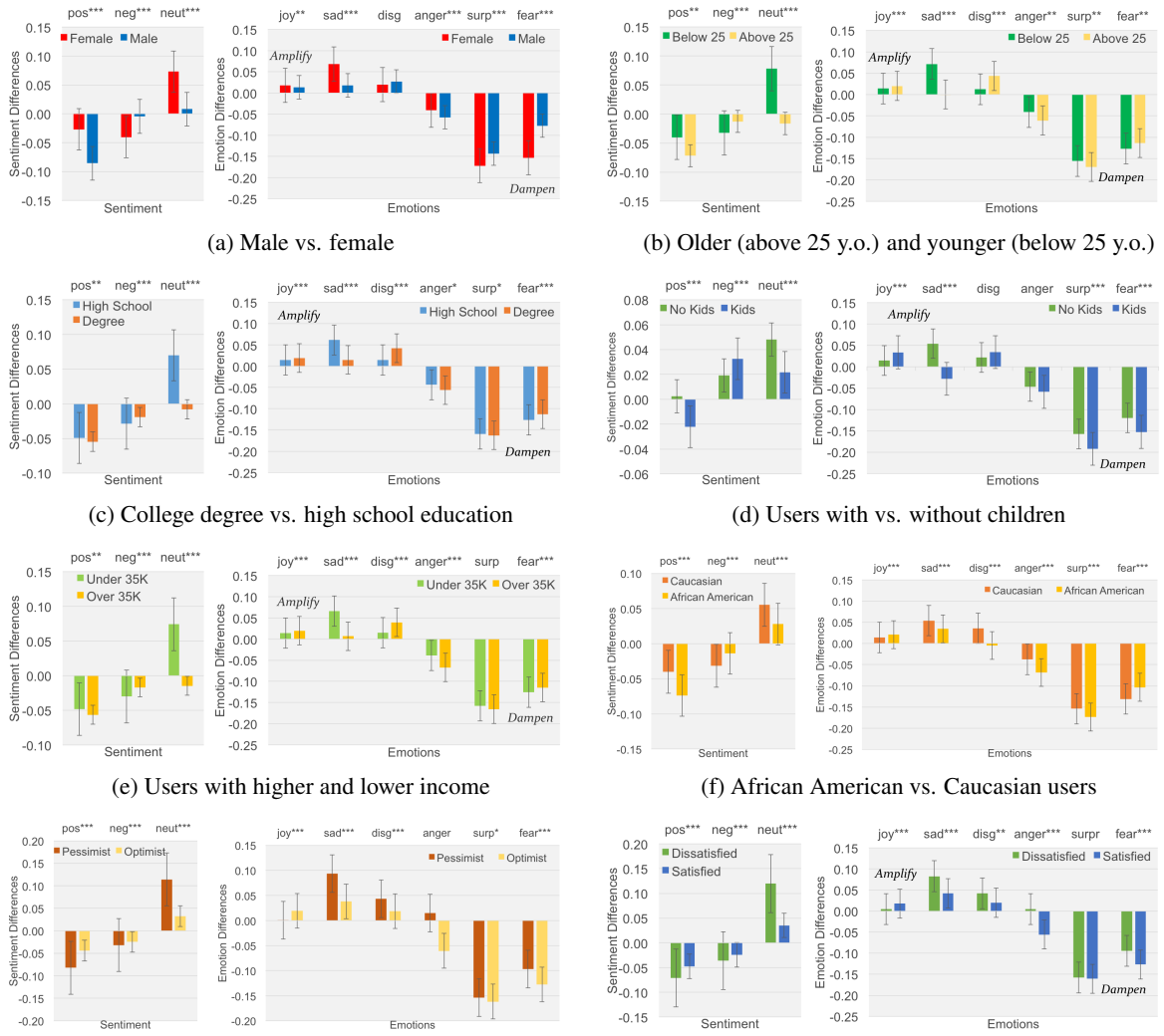
Our key findings discussed below confirm the *demographic dependent emotional contrast hypothesis*. We found that regardless demographics Twitter users tend to express more $(U > N)$ sadness$\uparrow$, disgust$\uparrow$, joy$\uparrow$ and neutral$\uparrow$ opinions and express less $(U < N)$ surprise$\downarrow$, fear$\downarrow$, anger$\downarrow$, positive$\downarrow$ and negative$\downarrow$ opinions compared to their neighbors except some exclusions below.

Users predicted to be older and having kids express less sadness whereas younger users and user without kids express more. It is also known as the *aging positivity effect* recently picked up in social media (Kern et al., 2014). It states that older people are happier than younger people (Carstensen and Mikels, 2005). Users predicted to be pessimists express less joy compared to their neighbors whereas optimists express more. Users predicted to be dissatisfied with life express more anger compared to their environment whereas users predicted to be satisfied with life produce less. Users predicted to be older, with a degree and higher income express neutral opinions compared to their environment whereas users predicted to be younger, with lower income and high school education express more neutral opinions. Users predicted to be male and having kids express more positive opinions compared to their neighbors whereas female users and users without kids express less. We present more detailed analysis on user-environment emotional contrast for different attribute-affect combinations in Figure 2.

Gender Female users have a stronger tendency to express more surprise and fear compared to their environment. They express less sadness compared to male users, supporting the claim that female users are more emotionally driven than male users in social media (Volkova et al., 2013). Male users have a stronger tendency to express more anger compared to female users. Female users tend to express less negative opinions compared to their environment.

Age Younger users express more sadness but older users express similar level of sadness compared to their environment. It is also known as the *aging positivity effect* recently picked up in social media (Kern et al., 2014). It states that older people are happier than younger people (Carstensen and Mikels, 2005). They have a stronger tendency to express less anger but more disgust compared to younger users. Younger users have a stronger tendency to express less fear and negative sentiment compared to older users.

Education Users with a college degree have a weaker tendency to express less sadness but stronger tendency to express more disgust from



(a) Male vs. female

(b) Older (above 25 y.o.) and younger (below 25 y.o.)

(c) College degree vs. high school education

(d) Users with vs. without children

(e) Users with higher and lower income

(f) African American vs. Caucasian users

(g) Optimists vs. pessimists

(h) Satisfied vs. dissatisfied with life

Figure 2: Mean differences in affect proportions between users with contrasting demographics. Error bars show standard deviation for every e and s; p-values are shown as $\leq 0.01^{***}$, $\leq 0.05^{**}$ and $\leq 0.1^{*}$.

their environment compared to users with high school education. They have a stronger tendency to express less anger but weaker tendency to express less fear. Users with high school education are likely to express more neutral opinions whereas users with a college degree express less.

Children Users with children have a stronger tendency to express more joy, less surprise and fear from their environment compared to users without children. Users with children express less sadness and less positive opinions whereas users without children express more.

Income Users with higher annual income have a weaker tendency to express more sadness and have a stronger tendency to express more disgust, less anger and fear from their environment. They tend to express less neutral opinions whereas users with lower income express more.

Ethnicity Caucasian users have a stronger tendency to express more sadness and disgust from their environment whereas African American users have a stronger tendency to express more joy and less disgust. African American users have a stronger tendency to express less anger and surprise, but a weaker tendency to express less fear.

Optimism Optimists express more joy from their environment whereas pessimists do not. Instead, pessimists have a stronger tendency to express more sadness and disgust compared to optimists. Optimists tend to express less fear. Pessimists tend to express less positive but more neutral opinions.

Life Satisfaction User-environment emotional contrast for the life satisfaction attribute highly correlates with the optimism attribute. Users dissatisfied with life have a weaker tendency to ex-

press more joy but a stronger tendency to express more sadness and disgust. They express more anger whereas users satisfied with life express less anger. Users satisfied with life have a stronger tendency to express less fear but weaker tendency to express less positive and negative opinions.

In addition to our analysis on user-environment emotional contrast and demographics, we discovered which users are more "opinionated" relative to their environment on Twitter. In other words, users in which demographic group amplify less neutral but more subjective tweets e.g., positive, negative. As shown in Figure 2 male users are significantly more opinionated $\gg$ than female users, users with kids $>$ users without kids, users with a college degree $\gg$ users with high school education, older users $\gg$ younger users, users with higher income $\gg$ users with lower income, optimists $\gg$ pessimists, satisfied $\gg$ dissatisfied with life, and African American $>$ Caucasian users.

4.3 Inferring User Demographics From User-Environment Emotional Contrast

Our findings in previous sections indicate that predicted demographics correlate with the emotional contrast between users and their environment in social media. We now show that by using user emotional tone and user-environment emotional contrast we can quite accurately predict many demographic properties of the user.

Table 6 presents the quality of demographic predictions in terms of the area and the ROC curve based on different feature sets. These results indicate that most user traits can be quite accurately predicted using solely the emotional tone and emotional contrast features of the users. That is, given the emotions expressed by a user, and contrasting these with the emotions expressed by user environment, one can accurately infer many interesting properties of the user without using any additional information. We note that the emotional features have a strong influence on the prediction quality, resulting in significant absolute ROC AUC improvements over the lexical only feature set.

Furthermore, we analyze correlations between users' emotional-contrast features and their demographic traits. We found that differences between users and their environment in sadness, joy, anger and disgust could be used for predicting whether these users have children or not. Similarly, negative and neutral opinions, as opposed to joy, fear and surprise emotions can be predictive of users with higher education.

Attribute	Lexical	EmoSent	All	Δ
Age	0.63	0.74 (+0.11)	0.83	+0.20
Children	0.72	0.67 (–0.05)	0.80	+0.08
Education	0.77	0.78 (+0.01)	0.88	+0.11
Ethnicity	0.93	0.75 (–0.18)	0.97	+0.04
Gender	0.90	0.77 (–0.13)	0.95	+0.05
Income	0.73	0.77 (+0.04)	0.85	+0.12
Life Satisf.	0.72	0.77 (+0.05)	0.84	+0.12
Optimism	0.72	0.77 (+0.05)	0.83	+0.11

Table 6: Sociodemographic attribute prediction results in ROC AUC using Lexical, EmoSent (user emotional tone + user-environment emotional contrast), and All (EmoSent + Lexical) features extracted from user content.

5 Discussion

We examined the expression of emotions in social media, an issue that has also been the focus of recent work which analyzed emotion contagion using a controlled experiment on Facebook (Coviello et al., 2014). That study had important ethical implications, as it involved manipulating the emotional messages users viewed in a controlled way. It is not feasible for an arbitrary researcher to reproduce that experiment, as it was carried on the proprietary Facebook network. Further, the significant criticism of the ethical implications of the experimental design of that study (McNeal, 2014) indicates how problematic it is to carry out research on emotions in social networks using a controlled/interventional technique.

Our methodology for studying emotions in social media thus uses an *observational* method, focusing on Twitter. We collected subjective judgments on a range of previously unexplored user properties, and trained machine learning models to predict those properties for a large sample of Twitter users. We proposed a concrete quantitative definition of the emotional contrast between users and their network environment, based on the emotions emanating from the users versus their neighbors.

We showed that various demographic traits correlate with the emotional contrast between users and their environment, supporting the *demographic-dependent emotional contrast hypothesis*. We also demonstrated that it is possible to accurately predict many perceived demographic traits of Twitter users based solely on the emotional contrast between them and their neighbors. This suggests that the way in which the emotions we radiate differ from those expressed in our environment reveals a lot about our identity.

We note that our analysis and methodology have several limitations. First, we only study *correlations* between emotional contrast and demographics. As such we do not make any *causal* inference regarding these parameters. Second, our labels regarding demographic traits of Twitter users were the result of subjective reports obtained using human annotations – subjective impressions (Flekova et al., 2016) of people rather than the true traits. Finally, we crawled both user and neighbor tweets within a short time frame (less than a week) and made sure that user and neighbor tweets were produced at the same time. Despite these limitations, our results do indicate higher performance compared to earlier work. Due to the large size of our dataset, we believe our findings are correct.

6 Related Work

Personal Analytics in Social Media Earlier work on predicting latent user attributes based on Twitter data uses supervised models with lexical features for classifying four main attributes including gender (Rao et al., 2010; Burger et al., 2011; Zamal et al., 2012), age (Zamal et al., 2012; Kosinski et al., 2013; Nguyen et al., 2013), political preferences (Volkova and Van Durme, 2015) and ethnicity (Rao et al., 2010; Bergsma et al., 2013).

Similar work characterizes Twitter users by using network structure information (Conover et al., 2011; Zamal et al., 2012; Volkova et al., 2014; Li et al., 2015), user interests and likes (Kosinski et al., 2013; Volkova et al., 2016), profile pictures (Bachrach et al., 2012; Leqi et al., 2016).

Unlike the existing work, we not only focus on previously unexplored attributes e.g., having children, optimism and life satisfaction but also demonstrate that user attributes can be effectively predicted using emotion and sentiment features in addition to commonly used text features.

Emotion and Opinion Mining in Microblogs Emotion analysis[15] has been successfully applied to many kinds of informal and short texts including emails, blogs (Kosinski et al., 2013), and news headlines (Strapparava and Mihalcea, 2007), but *emotions* in social media, including Twitter and Facebook, have only been investigated recently. Researchers have used supervised learning models trained on lexical word ngram features, synsets, emoticons, topics, and lexicon frameworks to determine which emotions are expressed on Twitter (Wang et al., 2012; Roberts et al., 2012; Qadir and Riloff, 2013; Mohammad and Kiritchenko, 2014). In contrast, *sentiment* classification in social media has been extensively studied (Pang et al., 2002; Pang and Lee, 2008; Pak and Paroubek, 2010; Hassan Saif, Miriam Fernandez and Alani, 2013; Nakov et al., 2013; Zhu et al., 2014).

Emotion Contagion in Social Networks Emotional contagion theory states that emotions and sentiments of two messages posted by friends are more likely to be similar than those of two randomly selected messages (Hatfield and Cacioppo, 1994). There have been recent studies about emotion contagion in massively large social networks (Fan et al., 2013; Ferrara and Yang, 2015b; Bollen et al., 2011a; Ferrara and Yang, 2015a).

Unlike these papers, we do not aim to model the spread of emotions or opinions in a social network. Instead, given both homophilic and assortative properties of a Twitter social network, we study how emotions expressed by user neighbors correlate with user emotions, and whether these correlations depend on user demographic traits.

7 Summary

We examined a large-scale Twitter dataset to analyze the relation between perceived user demographics and the emotional contrast between users and their neighbors. Our results indicated that many sociodemographic traits correlate with user-environment emotional contrast. Further, we showed that one can accurately predict a wide range of perceived demographics of a user based solely on the emotions expressed by that user and user's social environment.

Our findings may advance the current understanding of social media population, their online behavior and well-being (Nguyen et al., 2015). Our observations can effectively improve personalized intelligent user interfaces in a way that reflects and adapts to user-specific characteristics and emotions. Moreover, our models for predicting user demographics can be effectively used for a variety of downstream NLP tasks e.g., text classification (Hovy, 2015), sentiment analysis (Volkova et al., 2013), paraphrasing (Preotiuc-Pietro et al., 2016), part-of-speech tagging (Hovy and Søgaard, 2015; Johannsen et al., 2015) and visual analytics (Dou et al., 2015).

[15]EmoTag: http://nil.fdi.ucm.es/index.php?q=node/186

References

Yoram Bachrach, Michal Kosinski, Thore Graepel, Pushmeet Kohli, and David Stillwell. 2012. Personality and patterns of Facebook usage. In *Proceedings of ACM WebSci*, pages 24–32.

Yoram Bachrach. 2015. Human judgments in hiring decisions based on online social network profiles. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE.

David Bamman, Jacob Eisenstein, and Tyler Schnoebelen. 2014. Gender identity and lexical variation in social media. *Journal of Sociolinguistics*, 18(2):135–160.

Shane Bergsma, Mark Dredze, Benjamin Van Durme, Theresa Wilson, and David Yarowsky. 2013. Broadly improving user classification via communication-based name and location clustering on Twitter. In *Proceedings of NAACL-HLT*, pages 1010–1019.

Johan Bollen, Bruno Gonçalves, Guangchen Ruan, and Huina Mao. 2011a. Happiness is assortative in online social networks. *Artificial life*, 17(3):237–251.

Johan Bollen, Huina Mao, and Xiaojun Zeng. 2011b. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

John D. Burger, John Henderson, George Kim, and Guido Zarrella. 2011. Discriminating gender on Twitter. In *Proceedings of EMNLP*, pages 1301–1309.

Laura L Carstensen and Joseph A Mikels. 2005. At the intersection of emotion and cognition aging and the positivity effect. *Current Directions in Psychological Science*, 14(3):117–121.

Raviv Cohen and Derek Ruths. 2013. Classifying political orientation on Twitter: It's not easy! In *Proceedings of ICWSM*.

Michael D. Conover, Bruno Gonçalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. 2011. Predicting the political alignment of Twitter users. In *Proceedings of Social Computing*.

Lorenzo Coviello, Yunkyu Sohn, Adam DI Kramer, Cameron Marlow, Massimo Franceschetti, Nicholas A Christakis, and James H Fowler. 2014. Detecting emotional contagion in massive social networks. *PloS one*, 9(3):e90315.

Amy JC Cuddy, Susan T Fiske, Virginia SY Kwan, Peter Glick, Stéphanie Demoulin, Jacques-Philippe Leyens, Michael Harris Bond, Jean-Claude Croizet, Naomi Ellemers, Ed Sleebos, et al. 2009. Stereotype content model across cultures: Towards universal similarities and some differences. *British Journal of Social Psychology*, 48(1):1–33.

Aron Culotta, Nirmal Kumar Ravi, and Jennifer Cutler. 2015. Predicting the demographics of Twitter users from website traffic data. In *Proceedings of AAAI*.

Munmun De Choudhury, Michael Gamon, and Scott Counts. 2012. Happy, nervous or surprised? Classification of human affective states in social media. In *Proceedings of ICWSM*.

Wenwen Dou, Isaac Cho, Omar ElTayeby, Jaegul Choo, Xiaoyu Wang, and William Ribarsky. 2015. Demographicvis: Analyzing demographic information based on user generated content. In *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*, pages 57–64. IEEE.

Jacob Eisenstein, Brendan O'Connor, Noah A Smith, and Eric P Xing. 2014. Diffusion of lexical change in social media. *PloS one*, 9(11):e113114.

Rui Fan, Jichang Zhao, Yan Chen, and Ke Xu. 2013. Anger is more influential than joy: sentiment correlation in Weibo. *arXiv preprint arXiv:1309.2402*.

Emilio Ferrara and Zeyao Yang. 2015a. Measuring emotional contagion in social media. *PloS one*, 10(11):e0142390.

Emilio Ferrara and Zeyao Yang. 2015b. Quantifying the effect of sentiment on information diffusion in social media. *PeerJ Computer Science*, 1:e26.

Katja Filippova. 2012. User demographics and language in an implicit social network. In *Proceedings of EMNLP-CoNLL*.

Lucie Flekova, Salvatore Giorgi, Jordan Carpenter, Lyle Ungar, and Daniel Preotiuc-Pietro. 2015. Analyzing crowdsourced assessment of user traits through Twitter posts. *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing*.

Lucie Flekova, Jordan Carpenter, Salvatore Giorgi, Lyle Ungar, and Daniel Preotiuc-Pietro. 2016. Analyzing biases in human perception of user age and gender from text. In *Proceedings of the Association for Computational Linguistics*.

Jennifer Golbeck, Cristina Robles, Michon Edmondson, and Karen Turner. 2011. Predicting personality from Twitter. In *Proceedings of SocialCom/PASSAT*.

Roberto González-Ibáñez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in Twitter: A closer look. In *Proceedings of ACL*, pages 581–586.

Yulan He Hassan Saif, Miriam Fernandez and Harith Alani. 2013. Evaluation datasets for Twitter sentiment analysis: A survey and a new dataset, the sts-gold. *First ESSEM workshop*.

Elaine Hatfield and John T Cacioppo. 1994. *Emotional contagion*. Cambridge university press.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 483–488.

Dirk Hovy. 2015. Demographic factors improve classification performance. *Proceedings of ACL*.

Anders Johannsen, Dirk Hovy, and Anders Søgaard. 2015. Cross-lingual syntactic variation over age and gender. In *Proceedings of CoNLL*.

Daniel Kahneman and Angus Deaton. 2010. High income improves evaluation of life but not emotional well-being. *Proceedings of the National Academy of Sciences*, 107(38):16489–16493.

Alfredo Kalaitzis, Maria Ivanova Gorinova, Yoad Lewenberg, Yoram Bachrach, Michael Fagan, Dean Carignan, and Nitin Gautam. 2016. Predicting gaming related properties from twitter profiles. In *2016 IEEE Second International Conference on Big Data Computing Service and Applications (BigDataService)*, pages 28–35. IEEE.

Margaret L Kern, Johannes C Eichstaedt, H Andrew Schwartz, Gregory Park, Lyle H Ungar, David J Stillwell, Michal Kosinski, Lukasz Dziurzynski, and Martin EP Seligman. 2014. From sooo excited!!! to so proud: Using language to study development. *Developmental psychology*, 50(1):178.

Michal Kosinski, David Stillwell, and Thore Graepel. 2013. Private traits and attributes are predictable from digital records of human behavior. *National Academy of Sciences*.

Liu Leqi, Daniel Preoţiuc-Pietro, Zahra Riahi, Mohsen E. Moghaddam, and Lyle Ungar. 2016. Analyzing personality through social media profile picture choice. *ICWSM*.

Yoad Lewenberg, Yoram Bachrach, and Svitlana Volkova. 2015. Using emotions to predict user interest areas in online social networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*, pages 1–10. IEEE.

Jiwei Li, Alan Ritter, and Eduard Hovy. 2014. Weakly supervised user profile extraction from Twitter. *Proceedings of ACL*.

Jiwei Li, Alan Ritter, and Dan Jurafsky. 2015. Learning multi-faceted representations of individuals from heterogeneous evidence using neural networks. *arXiv preprint arXiv:1510.05198*.

Gregory McNeal. 2014. Facebook manipulated user news feeds to create emotional responses. *Forbes*.

Saif M. Mohammad and Svetlana Kiritchenko. 2014. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence*.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of SemEval*, June.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in Twitter. In *Proceedings of SemEval*, pages 312–320.

Dong Nguyen, Noah A. Smith, and Carolyn P. Rosé. 2011. Author age prediction from text using linear regression. In *Proceedings of LaTeCH*, pages 115–123.

Dong Nguyen, Rilana Gravel, Dolf Trieschnigg, and Theo Meder. 2013. "How old do you think I am?" A study of language and age in Twitter. In *Proceedings of ICWSM*, pages 439–448.

Dong Nguyen, A Seza Doğruöz, Carolyn P Rosé, and Franciska de Jong. 2015. Computational sociolinguistics: A survey. *arXiv preprint arXiv:1508.07544*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations of Trends in IR*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86.

Marco Pennacchiotti and Ana-Maria Popescu. 2011a. Democrats, republicans and starbucks afficionados: user classification in Twitter. In *Proceedings of KDD*, pages 430–438.

Marco Pennacchiotti and Ana Maria Popescu. 2011b. A machine learning approach to Twitter user classification. In *Proceedings of ICWSM*, pages 281–288.

Daniel Preoiuc-Pietro, Svitlana Volkova, Vasileios Lampos, Yoram Bachrach, and Nikolaos Aletras. 2015. Studying user income through language, behaviour and affect in social media. *PLoS ONE*, 10(9):e0138717, 09.

Daniel Preotiuc-Pietro, Wei Xu, and Lyle Ungar. 2016. Discovering user attribute stylistic differences via paraphrasing.

Ashequl Qadir and Ellen Riloff. 2013. Bootstrapped learning of emotion hashtags #hashtags4you. *WASSA 2013*.

Delip Rao, David Yarowsky, Abhishek Shreevats, and Manaswi Gupta. 2010. Classifying latent user attributes in Twitter. In *Proceedings of SMUC*, pages 37–44.

Kirk Roberts, Michael A Roach, Joseph Johnson, Josh Guthrie, and Sanda M Harabagiu. 2012. Empatweet: Annotating and detecting emotions on Twitter. In *Proceedings of LREC*.

Derek Ruths, Jürgen Pfeffer, et al. 2014. Social media for large studies of behavior. *Science*, 346(6213):1063–1064.

Maarten Sap, Gregory Park, Johannes Eichstaedt, Margaret Kern, David Stillwell, Michal Kosinski, Lyle Ungar, and Hansen Andrew Schwartz. 2014. Developing age and gender predictive lexica over social media. In *Proceedings of EMNLP*.

Hansen Andrew Schwartz, Johannes C Eichstaedt, Margaret L Kern, Lukasz Dziurzynski, Richard E Lucas, Megha Agrawal, Gregory J Park, Shrinidhi K Lakshmikanth, Sneha Jha, Martin EP Seligman, et al. 2013. Characterizing geographic variation in well-being using tweets. In *ICWSM*.

Luke Sloan, Jeffrey Morgan, Pete Burnap, and Matthew Williams. 2015. Who tweets? deriving the demographic characteristics of age, occupation and social class from twitter user meta-data. *PloS one*, 10(3):e0115545.

Carlo Strapparava and Rada Mihalcea. 2007. Semeval-2007 task 14: Affective text. In *Proceedings of SemEval*, pages 70–74.

Ro Valitutti. 2004. Wordnet-affect: an affective extension of wordnet. In *Proceedings of LREC*, pages 1083–1086.

Benjamin Van Durme. 2012. Streaming analysis of discourse participants. In *Proceedings of EMNLP*, pages 48–58.

Svitlana Volkova and Yoram Bachrach. 2015. On predicting sociodemographic traits and emotions from communications in social networks and their implications to online self-disclosure. *Cyberpsychology, Behavior, and Social Networking*, 18(12):726–736.

Svitlana Volkova and Benjamin Van Durme. 2015. Online bayesian models for personal analytics in social media. In *Proceedings of AAAI*.

Svitlana Volkova, Theresa Wilson, and David Yarowsky. 2013. Exploring demographic language variations to improve multilingual sentiment analysis in social media. In *Proceedings of EMNLP*.

Svitlana Volkova, Glen Coppersmith, and Benjamin Van Durme. 2014. Inferring user political preferences from streaming communications. In *Proceedings of ACL*, pages 186–196.

Svitlana Volkova, Yoram Bachrach, Michael Armstrong, and Vijay Sharma. 2015. Inferring latent user properties from texts published in social media (demo). In *Proceedings of AAAI*.

Svitlana Volkova, Yoram Bachrach, and Benjamin Van Durme. 2016. Mining user interests to predict perceived psycho-demographic traits on Twitter.

Sida Wang and Christopher D Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94.

Wenbo Wang, Lu Chen, Krishnaprasad Thirunarayan, and Amit P Sheth. 2012. Harnessing Twitter "big data" for automatic emotion identification. In *Proceedings of SocialCom*, pages 587–592.

Anna Wierzbicka. 1986. Human emotions: universal or culture-specific? *American Anthropologist*, 88(3):584–594.

Faiyaz Al Zamal, Wendy Liu, and Derek Ruths. 2012. Homophily and latent attribute inference: Inferring latent attributes of Twitter users from neighbors. In *Proceedings of ICWSM*.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif M Mohammad. 2014. NRC-Canada-2014: Recent improvements in the sentiment analysis of tweets. *SemEval*.

Prototype Synthesis for Model Laws

Matthew Burgess
University of Michigan
mattburg@umich.edu

Eugenia Giraudy
UC Berkeley & YouGov
eugenia.giraudy@gmail.com

Eytan Adar
University of Michigan
eadar@umich.edu

Abstract

State legislatures often rely on existing text when drafting new bills. Resource and expertise constraints, which often drive this copying behavior, can be taken advantage of by lobbyists and special interest groups. These groups provide *model bills*, which encode policy agendas, with the intent that the models become actual law. Unfortunately, model legislation is often opaque to the public–both in source and content. In this paper we present LOBBYBACK, a system that reverse engineers model legislation from observed text. LOBBYBACK identifies clusters of bills which have text reuse and generates "prototypes" that represent a canonical version of the text shared between the documents. We demonstrate that LOBBYBACK accurately reconstructs model legislation and apply it to a dataset of over 550k bills.

1 Introduction

Beginning in 2005, a number of states began passing "Stand Your Ground" laws–legal protections for the use of deadly force in self-defense. Within a few years, at least two dozen states implemented a version of the this legislation (Garrett and Jansa, 2015). Though each state passed its own variant, there is striking similarity in the text of the legislation. While seemingly "viral" the expedient adoption of these laws was not the result of an organic diffusion process, but rather more centralized efforts. An interest group, the American Legislative Exchange Council (ALEC), drafted model legislation (in this case modeled on Florida's law) and lobbied to have the model law enacted in other states. While the influence of the lobbyists through model laws grows, the hidden nature of their original text (and source) creates a troubling opacity.

Reconstructing such hidden text through analysis of observed, potentially highly mutated, copies poses an interesting and challenging NLP problem. We refer to this as the *Dark Corpora* problem. Since legislatures are not required to cite the source of the text that goes into a drafted bill, the bills that share text are unknown beforehand. The first problem therein lies in identifying clusters of bills with reused text. Once a cluster is identified, a second challenge is the reconstruction of the original or prototype bill that corresponds to the observed text. The usual circumstances under which a model law is adopted by individual states involves "mutation." This may be as simple as modifying parameters to the existing policy (*e.g.*, changing the legal limit allowed of medical marijuana possession to 3.0 ounces from 2.5) or can be more substantial, with significant additions or deletions of different conditions of a policy. Interestingly, the need to maintain the objectives of the law creates a pressure to retain a legally meaningful structure and precise language–thus changes need to satisfy the existing laws of the state but carry out the intent of the model. Both subtle changes of this type, and more dramatic ones, are of great interest to political scientists. A specific application, for example, may be predicting likely spots for future modifications as additional states adopt the law. Our challenge is to identify and represent "prototype" sentences that capture the *similarity* of observed sentences while also capturing the *variation*.

In this paper we propose LOBBYBACK, a system that automatically identifies clusters of documents that exhibit text reuse, and generates "prototypes" that represent a canonical version of text shared between the documents. In order to syn-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1579–1588,
Berlin, Germany, August 7-12, 2016.

thesize the prototypes, LOBBYBACK first extracts clusters of sentences, where each sentence pertains to the same policy but can exhibit variation. LOBBYBACK then uses a greedy multi-sequence alignment algorithm to identify an approximation of the optimal alignment between the sentences. Prototype sentences are synthesized by computing a consensus sentence from the multi-sentence alignment. As sentence variants are critical in understanding the effect of the model legislation, we can not simply generate a single "common" summary sentence as a prototype. Rather, LOBBYBACK creates a data structure that captures this variability in text for display to the end-user.

With LOBBYBACK, end-users can quickly identify clusters of text reuse to better understand what type of policies are diffused across states. In other applications, sentence prototypes can be used by journalists and researchers to discover previously unknown model legislation and the involvement of lobbying organizations. For example, prototype text can be compared to the language or policy content of interest groups documents and accompanied with qualitative research it can help discover which lobbyists have drafted this legislation.

We evaluated LOBBYBACK on the task of reconstructing 122 known model legislation documents. Our system was able to achieve an average of 0.6 F1 score based on the number of prototype sentences that had high similarity with sentences from the model legislation. We have also run LOBBYBACK on the entire corpus of state legislation (571,000 documents) from openstates.org as an open task. The system identified 4,446 clusters for which we generated prototype documents. We have released the resulting data set and code at http://github.com/mattburg/LobbyBack. LOBBYBACK is novel in fully automating and scaling the pipeline of model-legislation reconstruction. The output of this pipeline captures both the likely "source sentences" but also the variations of those sentences.

2 Related Work

While no specific system or technique has focused on the problem of legislative document reconstruction, we find related work in a number of domains. Multi-document summarization (MDS), for example, can be used to partially model the underlying problem–generating a representative document from multiple sources. Extractive MDS, in particular, is promising in that representative sentences are identified.

Early work in extractive summarization include greedy approaches such as that proposed by Carbonell and Goldstein (1998). The algorithm uses an objective function which trades off between relevance and redundancy. Global optimization techniques attempt to generate "summaries" (selected sets of sentences or utterances) that maximize an objective based on informativeness, redundancy and/or length of summary. These have shown superior performance to greedy algorithms (Yih et al., 2007; Gillick et al., 2009). Approaches based on neural networks have recently been proposed for ranking candidate sentences (Cao et al., 2015). Graph based methods, such as LexRank (Erkan and Radev, 2004), have also proven effective for MDS. Extensions to this approach combine sentence ranking with clustering in order to minimize redundancy (Qazvinian and Radev, 2008; Wan and Yang, 2008; Cai and Li, 2013). The C-LexRank algorithm (Qazvinian and Radev, 2008), in particular, uses this combination and inspired our high level design. Similar to our approach, Wang and Cardie (2013) propose a method for generating templates of meeting summaries using multi-sequence alignment.

Though related, it is important to note that the objectives of summarization (informativeness, reduced redundancy, etc.) are not entirely consistent with our task. For example, using the n-gram co-occurrence based ROUGE score would not be sufficient at evaluating LOBBYBACK. Our goal is to accurately reconstruct entire sentences of a hidden document given observed mutations of that document. Additionally, our goal is not simply to find a representative sentence that reflects the original document, but to capture the similarity *and* variability of the text within a given "sentence cluster."

Within the political science and legal studies communities research has focused on *manual* approaches to both understanding how model legislation impacts law and how policy ideas diffuse between bill text. As these studies are time consuming, there is no large-scale or broad analysis of legislative materials. Rather, researchers have limited their workload by focusing on a single topic (*e.g.*, abortion (Patton, 2003) and crime (Kent and Carmichael, 2015)) or a single lobbying group (*e.g.*, ALEC (Jackman, 2013). Similarly, those

studying policy diffusion across US states have also limited their analysis to a few topics (*e.g.*, same-sex marriage (Haider-Markel, 2001)).

Recent attempts to automate the analysis of model legislation has had similar problems, as most researchers have limited their analysis to one interest group or a few relevant topics (Hertel-Fernandez and Kashin, 2015; Jansa et al., 2015). Hertel-Fernandez and Kashin proposed a supervised model in which they train on hand labeled examples of state bills that borrow text and/or concepts from ALEC bills. The problem they focus on is different from ours. The motivation behind LOBBYBACK is that there exists many model legislation which we don't have access to and the goal is to try and reconstruct these documents without labeled training data. Jansa *et al.* propose a technique for inferring a network of policy diffusion for manually labeled clusters of bills. Both Jansa *et al.* and Hertel-Fernandaz and Kashin propose techniques that only look at the problem of inferring whether two bills exhibit text reuse but unlike LOBBYBACK they do not attempt to infer whether specific policies (sentences) in the documents are similar/different.

A related "dark corpora" problem, though at a far smaller scale, is the biblical "Q source" where hidden oral sources are reconstructed through textual analysis (Mournet, 2005).

3 Problem Definition

Policy diffusion is a common phenomenon in state bills (Gray, 1973; Shipan and Volden, 2008; Berry and Berry, 1990; Haider-Markel, 2001). Unlike members of the (Federal) Congress, few state legislators have the expertise, time, and staff to draft legislation. It is far easier for a legislator to adapt existing legislative text than to write a bill from scratch. As a consequence, state legislatures have an increased willingness to adopt legislation drafted by interest groups or by legislators in other states (Jansa et al., 2015). In addition to states borrowing text from other legislators and lobbyists, another reason why bills can exhibit text reuse is when a new federal law passes and each state needs to modify its existing policy to conform with the new federal law.

The result of legislative copying, whether caused by diffusion between states, influence from a lobby or the passing of a new federal law, is similar: a cluster of bills will share very similar text, often varying only by implementation details of a given policy. The goal in constructing a prototype document–a representation of the "original" text–is to synthesize this document from the modified copies. In the case when the bill cluster was influenced by one external source, such as lobby or passage of a federal bill, the ideal prototype document would capture the language that each bill borrowed from the source document. In the case when their is not one single document that influenced a cluster of bills, the prototype will still give a summary of a concise description of the diffused text between bills, providing fast insight into what text was shared and changed within a bill cluster.

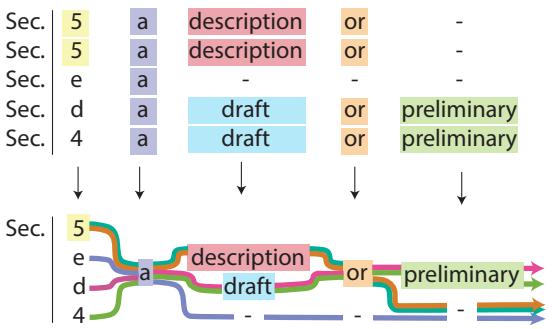


Figure 1: Visualization of a multi-sentence alignment (fragment) and resulting prototype sentence.

3.1 State Legislation Corpus

We obtained the entire openstates.org corpus of state legislation, which includes 550,000 bills and 200,000 resolutions for all 50 states. While for some states this corpus includes data since 2007, for the majority of states we have data from 2010 on. We do not include data from Puerto Rico, where the text is in Spanish, and from Washington DC, which includes many idiosyncrasies (*e.g.*, correspondence from city commissions introduced as bills). On average, each state introduced 10,524 bills, with an average length of 1205 words.

4 LOBBYBACK Architecture

LOBBYBACK consists of 3 major components. The first component identifies clusters of bills that have text reuse. Then for each of these bill clusters, LOBBYBACK extracts and clusters the sentences from all documents. For each of the sentence clusters, LOBBYBACK synthesizes prototype sentences in order to capture the similarity and variability of the sentences in the cluster.

4.1 Clustering Bills

Groups of bills with significant text reuse represent candidates for analysis as they have may have all copied from the same model legislation. There are a number of ways one could identify such clusters through text mining. In our implementation, we have opted to generate a network representation of the bills and then use a network clustering (i.e., "community-finding") algorithm to generate the bill clusters. In our network representation each node represents a state bill and weighted edges represent the degree to which two bills exhibit substantive text reuse. Since most pairs of bills do not have text reuse, we chose to use a network model because community finding algorithms work well on sparse data and do not require any parameter choice for the number of clusters. In the context of this paper, text reuse occurs when two state bills share:

1. Long passages of text, *e.g.* (sections of bills) that *can* differ in details.
2. These passages contain text of substantive nature to the topic of the bill (*i.e.*, text that is *not* boilerplate).

In addition to text that describes policy, state bills also contain boilerplate text that is common to all bills from a particular state or to a particular topic. Examples of legislative boilerplate include: "Read first time 01/29/16. Referred to Committee on Higher Education" (meta-data describing where the bill is in the legislative process); and "Safety clause. The general assembly hereby finds, determines, and declares ..." (a standard clause included in nearly all legislation from Colorado, stating the eligibility of a bill to be petitioned with a referendum).

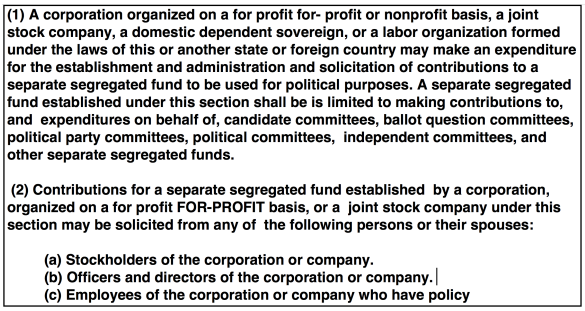
In order to identify pairs of bills that exhibit text reuse, we created an inverted index that contained n-grams ranging from size 4-8. We use ElasticSearch to implement the inverted index and computed the similarity between bills using the "More like This" (MLT) query (Elastic, 2016). The MLT query first selects the 100 highest scoring TF*IDF n-grams from a given document and uses those to form a search query. The MLT query is able to quickly compute the similarity between documents and since it ranks the query terms by using TF*IDF the query text is more likely to be substantive rather then boilerplate. The MLT query we used was configured to only return documents that matched at least 50% of the query's shingles and returned at most 100 documents per query. By implementing the similarity search using a TF*IDF cutoff we were able to scale the similarity computation while still maintaining our desire to identify reuse of substantive text.

The edges of the bill similarity network are computed by calculating pairwise similarity. Each bill is submitted as input for an MLT query and scored matches are returned by the search engine. Since the MLT query extracts n-grams only for the query document, the similarity function between two documents d_i and d_j is not symmetric. We construct a symmetric bill similarity network by taking the average score of each (d_i, d_j) and its reciprocal (d_j, d_i). A non-existent edge is represented as an edge with score 0. We further reduce the occurrence of false-positive edges by removing all edges with a score lower than 0.1. The resulting network is very sparse, consisting of 35,346 bills that have 1 or more neighbors, 125,401 edges, and 3534 connected components that contain an average of 10 bills.

A specific connected component may contain more than one bill cluster. To isolate these clusters in the bill network we use the InfoMap community detection algorithm (Rosvall and Bergstrom, 2008). We use the InfoMap algorithm because it has been to shown to be one of the best community detection algorithms and it is able to detect clusters at a finer granularity than other methods. Our corpus contains both bills that have passed and those that have not. Bills can often be reintroduced in their entirety after failing the previous year. As we do not want to bias the clusters towards bills that are re-introduced more than others, we filter the clusters such that they only include the earliest bill from each state.

4.2 Prototype Synthesis

Once we have identified a cluster of bills that have likely emerged from a single "source" we would like to construct a plausible representation of that source. The prototype synthesizer achieves this by constructing a canonical document that captures the similarity and variability of the content in a given bill cluster. The two main steps in prototype synthesis consists of clustering bill sentences and generating prototype sentences from the clusters.


(1) A corporation organized on a for profit for- profit or nonprofit basis, a joint stock company, a domestic dependent sovereign, or a labor organization formed under the laws of this or another state or foreign country may make an expenditure for the establishment and administration and solicitation of contributions to a separate segregated fund to be used for political purposes. A separate segregated fund established under this section shall be is limited to making contributions to, and expenditures on behalf of, candidate committees, ballot question committees, political party committees, political committees, independent committees, and other separate segregated funds.

(2) Contributions for a separate segregated fund established by a corporation, organized on a for profit FOR-PROFIT basis, or a joint stock company under this section may be solicited from any of the following persons or their spouses:

(a) Stockholders of the corporation or company.
(b) Officers and directors of the corporation or company.
(c) Employees of the corporation or company who have policy

Figure 2: An sample state bill segment from Michigan Senate Bill 571

4.2.1 Sentence Clustering

Most state bills have a common structure, consisting of an introduction that describes the intent of the bill followed by sections that contain the law to be implemented. Each section of a bill is comprised of self-contained policy, usually consisting of a long sentence that describes the policy and the implementation details of that policy. Each document is segmented into these policy sentences using the standard Python NLTK sentence extractor. Sentences are cleaned by removing spurious white space characters, surrounding punctuation and lower-casing each word. Once we have extracted all of the sentences for a given bill cluster, we compute the cosine similarity between all pairs of sentences which are represented using a unigram bag-of-words model. We used a simple unweighted bag of words model because in legal text stop words can be import[1] In this case, we are generating a similarity "matrix" capturing sentence–sentence similarity.

Given the similarity matrix, our next goal is to isolate clusters of variant sentences that likely came from the same source sentence. We elected to use the DBSCAN (Ester et al., 1996) algorithm to generate these clusters. The DBSCAN algorithm provides us with tunable parameters that can isolate better clusters. Specifically, the parameter ϵ controls the maximum distance between any two points in the same neighborhood. By varying ϵ we are able to control both the number of clusters and the amount of sentence variation within a cluster. A second reason for selecting DBSCAN is that the algorithm automatically deals with noisy data points, placing all points that are not close enough to other points in a separate cluster labeled "noise." Since many sentences in a given bill cluster do not contribute to the reused text between bills, the noise cluster is useful for grouping those sentences together rather than having them be outsiders in "good" clusters.

[1]The difference between the words "shall" and "may" for instance is important, the former requires that a specific action be put on a states budget while the later does not

4.2.2 Multi-Sequence Alignment

Once we have sentence clusters we then synthesize a "prototype" sentence from all of the sentences in a given cluster. An ideal prototype "sentence" is one that simultaneously captures the similarity between each sentence in the cluster (the common sentence structures) and the variation between the sentences in a cluster. For a simple pair of (partial) sentences, "The Department of Motor Vehicles retains the right to . . . " and "The Department of Transportation retains the right to . . . ", a prototype might be of the form, "The Department of { Motor Vehicles, Transportation } retains the rights to . . . " Our "sentence" is not strictly a single linear piece of text. Rather, we have a data structure that describes alternative sub-strings and captures variant text.

To generate this structure we propose an algorithm that computes an approximation of the optimal multi-sentence alignment (MSA) in the cluster and then generates a prototype sentence representing a 'consensus' for sentences in the MSA.

We generate an MSA using a modified version of the iterative pairwise alignment algorithm described in (Gusfield, 1997). The greedy algorithm builds a multi-alignment by iteratively applying the Needleman-Wunsch pairwise global alignment algorithm. Needleman-Wunsch computes the optimal pairwise alignment by maximizing the alignment score between two sentences. An alignment score is calculated based on three parameters: word matches, word mismatches (when a word appears in one sentence but not the other), and gaps (when the algorithm inserts a space in one of the sentences).

An MSA is generated by the following steps:

1. Construct an edit-distance matrix for all pairs of sentences
2. Construct an initial alignment between the two sentences with the smallest edit distance
3. Repeat this step k times:
   (a) Select the sentence with the smallest average edit distance to the current MSA.
   (b) Add the chosen sentence to the MSA by aligning it to the existing MSA.

The algorithm stops after the alignment has reached a size that is determined as a free parameter k (user configured, but can be chosen to be the number of sentences in the cluster). As the algorithm execution is ordered on the edit distance between the current MSA and the next sentence to be added, the larger the MSA, the more variation we are allowing in the prototype sentence.

4.2.3 Synthesizing Prototype Sentences

We synthesize a prototype sentence by finding a *consensus* sentence from all of the aligned sentences in the MSA for a given cluster. We achieve this by going through each "column" of the MSA and using the following rules to decide which token will be used in the prototype. A token can be either a word in one of the sentences or a "space" that was inserted during the alignment process.

1. If there is a token that occurs in the majority of alignments ($> 50\%$) then that token is chosen.
2. If no token appears in a majority, then a special variable token is constructed that displays all of the possible tokens in each sentence. For example the 1^{st} and 3^{rd} columns in Figure 1.
3. If a space is the majority token chosen then it is shown as a variable token with the second most common token.

5 Evaluation

We provide experiments that evaluate all three components of LOBBYBACK.

5.1 Model Legislation Corpus

To test the effectiveness of LOBBYBACK in recreating a model bill, we first identified a set of known model bills. Our model legislation corpus consists of 1846 model bills that we found by searching on Google using the keywords "model law", "model policy" and "model legislation." Most of the corpus is comprised of bills from the conservative lobby group, American Legislative Exchange Council (ALEC, 708 documents), its liberal counterpart, the State Innovative Exchange (SIX, 269 documents) and the non-partisan Council of State Governments (CSG, 470 documents), and the remainder (399) from smaller interest groups that focus on specific issues.

Using the clusters we previously described (Section 4.1), we found the most similar cluster to each model bill. This was done by first computing the set of neighbors (ego-network) for a model bill using the same procedure used in creating the bill similarity network. We then matched a bill cluster to the model legislation by finding the bill cluster that had the highest Jaccard similarity (based on the overlapping bills in each cluster) with the neighbor set of a model bill. Each test example in our evaluation data set consists of model bill and its corresponding bill cluster. The total number of model legislation documents that had matches in the state bill corpus was 360 documents.

Once we have an evaluation data set comprised of model bill/cluster pairs our goal is to compare the prototype sentences we infer for a cluster to the model bill that matches that cluster. Since we do not have ground truth on which sentences from the model bill match sentences in the documents that comprise the cluster we need to infer such labels. In order to identify which sentences from the model legislation actually get re-used in the bill cluster, we take the following steps:

1. Extract all sentences from each of the bills in a cluster and the sentences in the corresponding model legislation.
2. Compute the pairwise cosine similarity between bill sentences and each of the model bill sentences using the same unigram bag-of-words model described in Section 4.2.1
3. Compute the "oracle" matching M^* using the Munkres algorithm (Munkres, 1957)

The Munkres algorithm gives the best possible one-to-one matching between the sentences in the model legislation and the sentences in the bill clusters. There are some sentences in the model bill that are never used in actual state legislation (e.g., sentences that describe the intent of a law or instructions of how to implement a model policy). Therefore we label model bill sentences in M^* that match a bill sentence with a score greater than 0.85 as true matches (S^*)[2]. The final set of 122 evaluation examples consists of all model legislation/bill cluster pairs where more than 50% of model bill sentences have true matches.

[2]A threshold of 0.85 was found effective in prior work (Garrett and Jansa, 2015) and we observed good separation between matching sentences and non-matches in our data-set.

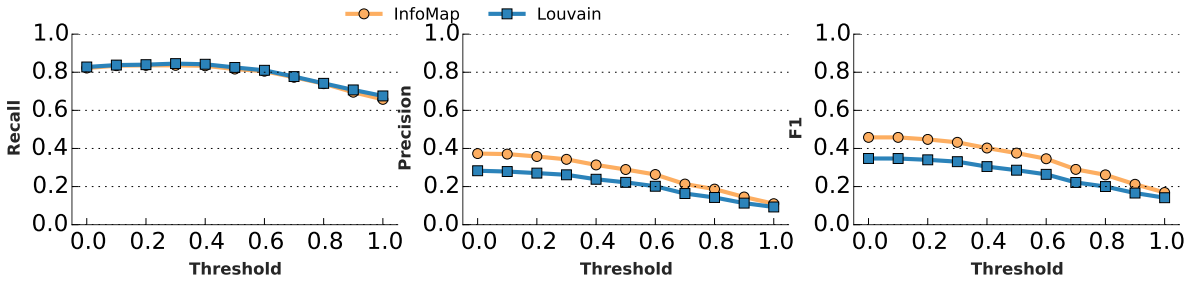


Figure 3: Precision, Recall, and F1 scores for the bill clustering component of LOBBYBACK configured with both the Louvain and InfoMap clustering algorithms.

5.2 Baselines

While no specific baseline exists for our problem, we implemented two alternatives to test against. The first, Random-Baseline, was implemented simply to show the performance of randomly constructing prototype documents. The second, LexRank-Baseline, implements a popular extractive summarization method.

Random-Baseline – The *random-baseline* randomly samples sentences from a given bill cluster. The number of sentences it samples is equal to the number of sentences in the optimal matching $|M^*|$

LexRank-Baseline – The LexRank baseline uses the exact same clustering algorithm as LOBBYBACK except instead of synthesizing prototype sentences, it uses the LexRank algorithm (Erkan and Radev, 2004) to pick the most salient sentence from each of the sentence clusters.

5.3 Evaluating Bill Clusters

As described in Section 5.1, each model bill is associated with a set of bills that comprise its neighbors in the bill network. In order to evaluate how LOBBYBACK clusters bills we compare the inferred clusters to the corresponding neighbor set for each model bill.

The inferred cluster for a given model bill can exhibit false positive and false negative errors. False negatives, in which a bill that exists in the neighbor set of a model bill but is not contained in the inferred cluster, are easy to identify (allowing us to calculate recall). Precision, on the other hand, is more difficult as any "extra" bills in the inferred cluster are not necessarily incorrect. It is possible that there are bills which do exhibit text re-use with the model legislation but did not match via the ElasticSearch query used to construct the network. We have found that in practice the second component of LOBBYBACK, which clusters the sentences extracted from a bill cluster, is robust to false-positives due to DBSCAN's treatment of "noisy" data points. Because of this, we are more concerned with recall in the bill clustering module as any data lost in this step propagates through the rest of the system.

Figure 3 shows the precision, recall, and F1 scores for LOBBYBACK coupled with both the Louvain (Blondel et al., 2008) and InfoMap (Rosvall and Bergstrom, 2008) network clustering algorithms. Because we do not know with absolute certainty which state bills are derived from model legislation, we would like to test our approach at different levels of confidence. To do this we vary the threshold on the similarity scores (edge weights) of the ego-network determined for each model legislation. A normalized weight is computed by taking the score provided by ElasticSearch and dividing that edge weight weight by the maximum edge weight in the neighbor set (to control for the unbounded scores provided by ElasticSearch). We vary the threshold for this weight (ranging from 0 to 1) and calculate the precision, recall, and F1 on a neighborhood set that is comprised of all bills that have a weight greater than the threshold. Higher threshold values means fewer state bills are included in the set, but they are increasingly similar to the model bill.

As shown in Figure 3, recall stays high for the majority of threshold values. This indicates that both clustering algorithms do a good job at recovering the bills that are the most similar to the model legislation. However, the two clustering algorithms differ somewhat in precision. Louvain, in particular performs worse, as it suffer from "resolution" problems. While effective for networks with large clusters, Louvain can not isolate small groups, of which we have many. For this reason,

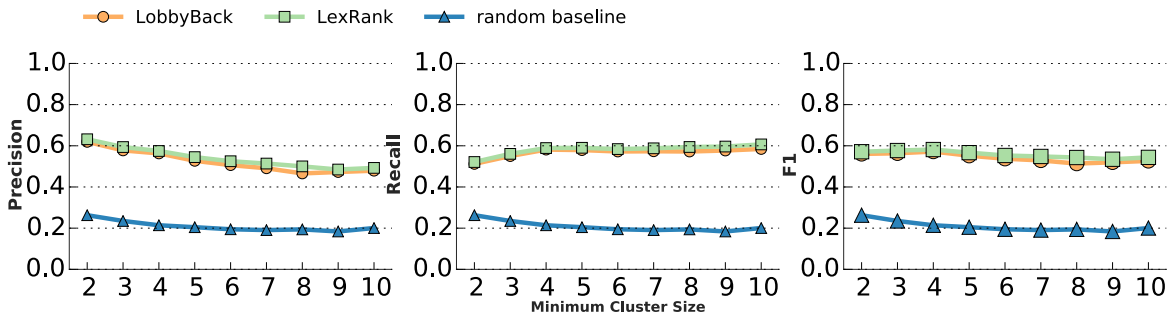


Figure 4: Precision, Recall, and F1 scores of LOBBYBACK and baselines.

we chose InfoMap as the method to use for the final implementation of LOBBYBACK.

5.4 Evaluating Sentence Clusters

We first evaluate the quality of the sentence clusters using the optimal matching M^* described above. For each test example in the evaluation set we generate a prototype document using LOBBYBACK and each of the baselines described above. We then compute a matching M between the prototype sentences and the model bill sentences (using the same procedure described in 5.1), where S is the set of sentences in the prototype and $S_{0.85}$ is the set of sentences that match with a score greater than 0.85. We compute precision, P, as $P = |S_{0.85}|/|S|$, and recall, R, as $R = |S|/|S^*|$.

Figure 4 shows precision, recall and F1 scores for both baselines and LOBBYBACK. Each curve is generated by averaging the precision/recall/F1 scores computed for each of the examples in the test set. The x-axis represents the minimum bill cluster size of the test examples for which the score is computed. For example, a minimum cluster size would average over all test examples with at least 2 bills in a cluster. LOBBYBACK relies on the fact that if text was borrowed from a model bill, then it would have been borrowed by many of the bills in the cluster. By analyzing how LOBBYBACK performs with respect to the minimum cluster size, we can determine how much evidence LOBBYBACK needs in order to construct clusters that correspond to sentences in the model bills. While the performance of LOBBYBACK and the LexRank baseline substantially improves over the random baseline, the different between LOBBYBACK and LexRank for this task is negligible. Since our cut-off similarity is 0.85, all sentences above the threshold are treated as true positives, making the distinction between the LexRank baseline and system small. LOBBYBACK performs a little worse than LexRank for large cluster sizes because it is penalized for having space and variable tokens which don't occur in model bills. Space and variable tokens occur more frequently in prototype sentences in larger clusters because there is more variation in the sentence clusters.

5.5 Evaluating Sentence Synthesis

The experiment in the previous section evaluated the quality of the sentence clusters by treating all matching sentences with a similarity greater than 0.85 as true positives. Here we provide an evaluation of the synthesized sentences that LOBBYBACK generates and compare them to the LexRank baseline, which chooses the most salient sentence from each cluster. We evaluate the quality of the synthesized prototype sentences by computing the word-based edit-distance between the prototype sentence with its corresponding model bill sentence in S for each test example.

Since the prototypes contain variable and space tokens which do not occur in the model bill sentences we modify the standard edit distance algorithm by *not* penalizing space tokens and allowing for any of the tokens that comprise a variable to be positive matches. In addition, we remove punctuation and lowercase all words in all of the sentences, regardless of method. We generate the results in Table 1 by averaging the edit distance for a configuration of LOBBYBACK or LexRank over sentence clusters produced for each test example. LOBBYBACK was configured to run with the number of iterations set to the size of the sentence cluster.

We compared both the performance of LOBBYBACK and LexRank for DBSCAN ϵ values of 0.1 and 0.15 as well as computing the average edit distance for different minimum sizes of cluster values. As the table shows, LOBBYBACK obtains a lower edit distance than LexRank in every config-

uration and as the size of the clusters increase the gap between the two increases. The goal of LOBBYBACK is not to be a better summarization algorithm than LexRank. By comparing to LexRank and showing that the edit distances are smaller on average, we can conclude that the prototype sentences created by LOBBYBACK are capturing the text that is "central" or similar within a given cluster. In addition, the prototype sentences produced by LOBBYBACK are superior because they also capture and describe in a succinct way, the variability of the sentences within a cluster.

6 Discussion

One assumption that we made about the nature of state adoption of model legislation is that the legislatures make modifications that largely preserve the model language in an effort to preserve policy. However, we currently do not consider cases in which a legislature has intentionally obscured the text while still retaining the same meaning. While not as frequent as common text reuse, Hertel-Fernandez and Kashin (2015) observed that some legislatures almost completely changed the text while reusing the concepts. One area of future work would be to try and extend LOBBYBACK to be more robust to these cases. One strategy would be to allow for a more flexible representation of text, such as word vector embeddings. The embeddings might even be used to extend the multi-sentence alignment to include a penalty based on word distance in embedding space.

LOBBYBACK performs well on reconstructing model legislation from automatically generated bill clusters. However, there are a number of improvements that can refine part of the pipeline. A potential change, but one that is more computationally costly, would be to use a deeper parsing of the sentences that we extract from the documents. We used a simple unigram model when computing sentence similarities because we wanted to ensure that stop words were included–due to their importance in legal text. We suspect that by using a parser we could, for example, weight the similarity of noun-phrases, yielding a better similarity matrix and potentially higher precision/recall. Currently LOBBYBACK does not consider the order of the sentences when attempting to construct a prototype document. We envision a future version of LOBBYBACK that tries to reconstruct the original ordering of the prototype sentences.

		Min Cluster Size			
Method	ϵ	2	4	6	8
LOBBYBACK	0.1	**24.4**	**20.4**	**18.2**	**17.2**
LexRank	0.1	25.4	22.5	20.3	19.4
LOBBYBACK	0.15	**25.5**	**21.6**	**19.4**	**17.9**
LexRank	0.15	27.3	25.6	25.0	24.1

Table 1: Mean edit distance scores for LOBBYBACK and LexRank.

7 Conclusion

In this paper we present LOBBYBACK, a system to reconstruct the "dark corpora" that is comprised of model bills which are copied (and modified) by resource constrained state legislatures. LOBBYBACK first identifies clusters of text reuse in a large corpora of state legislation and then generates prototype sentences that summarizes the similarity and variation of the copied text in a bill cluster. We believe that by open-sourcing LOBBYBACK and releasing our data of prototype bills to the public, journalists and legal scholars can use our findings to better understand the origination of U.S state laws.

Acknowledgements

The authors would like to thank Mike Cafarella, Joe Walsh and Rayid Ghani for helpful discussions, the Sunlight Foundation for providing access to data and the anonymous reviewers for their helpful comments.

References

Frances Stokes Berry and William D. Berry. 1990. State lottery adoptions as policy innovations: An event history analysis. *American Political Science Review*, 84:395–415, 6.

Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008.

Xiaoyan Cai and Wenjie Li. 2013. Ranking through clustering: An integrated approach to multi-document summarization. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(7):1424–1433.

Ziqiang Cao, Furu Wei, Li Dong, Sujian Li, and Ming Zhou. 2015. Ranking with recursive neural networks and its application to multi-document summarization. In *AAAI Conference on Artificial Intelligence*, AAAI'15, pages 2153–2159. AAAI Press.

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 335–336.

Elastic. 2016. Elasticsearch reference. *https://www.elastic.co/guide/en*.

Günes Erkan and Dragomir R. Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 22(1):457–479, December.

Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press.

Kristin N. Garrett and Joshua M. Jansa. 2015. Interest group influence in policy diffusion networks. *State Politics & Policy Quarterly*, 15(3):387–417.

Dan Gillick, Korbinian Riedhammer, Benoit Favre, and Dilek Hakkani-Tur. 2009. A global optimization framework for meeting summarization. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, ICASSP '09, pages 4769–4772, Washington, DC, USA. IEEE Computer Society.

Virginia Gray. 1973. Innovation in the states: A diffusion study. *American Political Science Review*, 67(04):1174–1185.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, New York, NY, USA.

Donald P Haider-Markel. 2001. Policy diffusion as a geographical expansion of the scope of political conflict: Same-sex marriage bans in the 1990s. *State Politics & Policy Quarterly*, 1(1):5–26.

Alexander Hertel-Fernandez and Konstantin Kashin. 2015. Capturing business power across the states with text reuse. In *Annual Conference of the Midwest Political Science Association*, pages 16–19.

Molly Jackman. 2013. Alecs influence over lawmaking in state legislatures. *Brookings Institute Blog*, December 6.

Joshua M Jansa, Eric R Hansen, and Virginia H Gray. 2015. Copy and paste lawmaking: The diffusion of policy language across american state legislatures. *Working Paper*.

Stephanie L Kent and Jason T Carmichael. 2015. Legislative responses to wrongful conviction: Do partisan principals and advocacy efforts influence state-level criminal justice policy? *Social science research*, 52:147–160.

Terence C Mournet. 2005. *Oral tradition and literary dependency: Variability and stability in the synoptic tradition and Q*, volume 195. Mohr Siebeck.

J. Munkres. 1957. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March.

Dana Jill Patton. 2003. *The Effect of U.S. Supreme Court Intervention on the Innovation and Diffusion of Post-Roe Abortion Policies in the American States, 1973-2000*. Ph.D. thesis, University of Kentucky.

Vahed Qazvinian and Dragomir R. Radev. 2008. Scientific paper summarization using citation summary networks. In *International Conference on Computational Linguistics*, COLING '08, pages 689–696, Stroudsburg, PA, USA. Association for Computational Linguistics.

Martin Rosvall and Carl T. Bergstrom. 2008. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123.

Charles R Shipan and Craig Volden. 2008. The mechanisms of policy diffusion. *American Journal of Political Science*, 52(4):840–857.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 299–306.

Lu Wang and Claire Cardie. 2013. Domain-independent abstract generation for focused meeting summarization. In *Annual Meeting of the Association for Computational Linguistics*, ACL'13, pages 1395–1405.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document summarization by maximizing informative content-words. In *International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1776–1782.

Which argument is more convincing? Analyzing and predicting convincingness of Web arguments using bidirectional LSTM

Ivan Habernal[†] and **Iryna Gurevych**[†‡]
[†]Ubiquitous Knowledge Processing Lab (UKP)
Department of Computer Science, Technische Universität Darmstadt
[‡]Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

Abstract

We propose a new task in the field of computational argumentation in which we investigate qualitative properties of Web arguments, namely their convincingness. We cast the problem as relation classification, where a pair of arguments having the same stance to the same prompt is judged. We annotate a large datasets of 16k pairs of arguments over 32 topics and investigate whether the relation "A is more convincing than B" exhibits properties of total ordering; these findings are used as global constraints for cleaning the crowdsourced data. We propose two tasks: (1) predicting which argument from an argument pair is more convincing and (2) ranking all arguments to the topic based on their convincingness. We experiment with feature-rich SVM and bidirectional LSTM and obtain 0.76-0.78 accuracy and 0.35-0.40 Spearman's correlation in a cross-topic evaluation. We release the newly created corpus *UKPConvArg1* and the experimental software under open licenses.

1 Introduction

What makes a good argument? Despite the recent achievements in computational argumentation, such as identifying argument components (Habernal and Gurevych, 2015; Habernal and Gurevych, 2016), finding evidence for claims (Rinott et al., 2015), or predicting argument structure (Peldszus and Stede, 2015; Stab and Gurevych, 2014), this question remains too hard to be answered.

Even Aristotle claimed that perceiving an argument as a "good" one depends on multiple factors (Aristotle and Kennedy (translator), 1991) — not only the logical structure of the argument (*logos*), but also on the speaker (*ethos*), emotions (*pathos*), or context (*cairos*) (Schiappa and Nordin, 2013). Experiments also show that different audiences perceive the very same arguments differently (Mercier and Sperber, 2011). A solid body of argumentation research has been devoted to the quality of arguments (Walton, 1989; Johnson and Blair, 2006), giving more profound criteria that "good" arguments should fulfill. However, the empirical evidence proving applicability of many theories falls short on everyday arguments (Boudry et al., 2015).

Since the main goal of argumentation is persuasion (Nettel and Roque, 2011; Mercier and Sperber, 2011; Blair, 2011; OKeefe, 2011) we take a pragmatic perspective on qualitative properties of argumentation and investigate a new high-level task. We asked whether we could quantify and predict how convincing an argument is.

Prompt: Should physical education be mandatory in schools? **Stance:** Yes!

Argument 1	Argument 2
physical education should be mandatory cuhz 112,000 people have died in the year 2011 so far and it's because of the lack of physical activity and people are becoming obese!!!!	YES, because some children don't understand anything excect physical education especially rich children of rich parents.

Figure 1: Example of an argument pair.

If we take Argument 1 from Figure 1, assigning a single "convincingness score" is highly subjective, given the lack of context, reader's prejudice, beliefs, etc. However, when comparing both arguments from the same example, one can decide that A1 is probably more convincing than A2, because it uses at least some statistics, addresses the health

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1589–1599,
Berlin, Germany, August 7-12, 2016.

factor, and A2 is just harsh and attacks.[1] We adapt pairwise comparison as our backbone approach.

We propose a novel task of predicting convincingness of arguments in an argument pair, as well as ranking arguments related to a certain topic. Since no data for such a task are available, we create a new annotated corpus. We employ SVM model with rich linguistic features as well as bidirectional Long Short-Term Memory (BLSTM) neural networks because of their excellent performance across various end-to-end NLP tasks (Goodfellow et al., 2016; Piech et al., 2015; Wen et al., 2016; Dyer et al., 2015; Rocktäschel et al., 2016).

Main contributions of this article are (1) large annotated dataset consisting of 16k argument pairs with 56k reasons in natural language (700k tokens), (2) thorough investigation of the annotated data with respect to properties of convincingness as a measure, (3) a SVM model and end-to-end BLSTM model. The annotated data, licensed under CC-BY-SA license, and the experimental code are publicly available at `https://github.com/UKPLab/acl2016-convincing-arguments`. We hope it will foster future research in computational argumentation and beyond.

2 Related Work

Recent years can be seen as a dawn of computational argumentation – an emerging sub-field of NLP in which natural language arguments and argumentation are modeled, searched, analyzed, generated, and evaluated. The main focus has been paid to analyzing argument structures, under the umbrella entitled argumentation mining.

Web discourse as a data source has been exploited in several tasks in argumentation mining, such as classifying propositions in user comments into three classes (verifiable experiential, verifiable non-experiential, and unverifiable) (Park and Cardie, 2014), or mapping argument components to Toulmin's model of argument in user-generated Web discourse (Habernal and Gurevych, 2015), to name a few. While these approaches are crucial for understanding the structure of an argument, they do not directly address any qualitative criteria of argumentation.

Argumentation quality has been an active topic among argumentation scholars. Walton (1989) discusses validity of arguments in informal logic, while Johnson and Blair (2006) elaborate on criteria for practical argument evaluation (namely Relevance, Acceptability, and Sufficiency). Yet, empirical research on argumentation quality does not seem to reflect these criteria and leans toward simplistic evaluation using argument structures, such as how many premises support a claim (Stegmann et al., 2011), or by the complexity of the analyzed argument scheme (Garcia-Mila et al., 2013). To the best of our knowledge, there have been only few attempts in computational argumentation that go deeper than analyzing argument structures (e.g., (Park and Cardie, 2014) mentioned above). Persing and Ng (2015) model argument strength in persuasive essays using a manually annotated corpus of 1,000 documents labeled with a 1–4 score value.

Our newly created corpus of annotated pairs of arguments might resemble recent large-scale corpora for textual inference. Bowman et al. (2015) introduced a 570k sentence pairs written by crowd-workers, the largest corpus to date. Whereas their task is to classify whether the sentence pair represents *entailment*, *contradiction*, or is *neutral* (thus heading towards a deep *semantic* understanding), our goal is to assess the *pragmatical* properties of the given multiple sentence-long arguments (to which extent they fulfill the goal of persuasion). Moreover, each of our annotated argument pairs is accompanied with five textual reasons that explain the rationale behind the labeler's decision. This is, to the best of our knowledge, a unique novel feature of our data.

Pairwise assessment for obtaining relative preference was examined by (Chen et al., 2013), among many others.[2] Their system was tested on ranking documents by their reading difficulty. Relative preference annotations have also been heavily employed in assessing machine translation (Aranberri et al., 2016). By contrast to our work, the underlying relations (reading difficulty 1-12 or better translation) have well known properties of total ordering, while convincingness of arguments is a yet unexplored task, thus no assumptions can be made apriori. There is also a substantial body of work on learning to rank, where also a *pairwise approach* is widely used (Cao et al., 2007). These methods have been traditionally used in IR,

[1] These are actual *reasons* provided by annotators, as will be explained later in Section 3.

[2] See (Shah et al., 2015) for a recent overview.

where the retrieved documents are ranked according to their relevance and pairs of documents are automatically sampled.

Employing LSTM for natural language inference tasks has recently gained popularity (Rocktäschel et al., 2016; Wang and Jiang, 2016; Cheng et al., 2016). These methods are usually tested on the SNLI data introduced above (Bowman et al., 2015).

3 Data annotation

Since assessing convincingness of a single argument directly is a very subjective task with high probability of introducing annotator's bias (because of personal preferences, beliefs, or background), we cast the problem as a relation annotation task. Given two arguments, one should be selected as more convincing, or they might be both equally convincing (see an example in Figure 1).

3.1 Sampling annotation candidates

Sampling large sets of arguments for annotation from the Web poses several challenges. First, we must be sure that the obtained texts are actual arguments. Second, the context of the argument should be known (the prompt and the stance). Finally, we need sources with permissive licenses, which allow us to release the resulting corpus further to the community. These criteria are met by arguments from two *debate portals*.[3]

We will use the following terminology. We use *topic* to refer to a subset of an on-line debate with a given prompt and a certain stance (for example, *"Should physical education be mandatory in schools? – yes"* is considered as a single topic). Each debate has two *topics*, one for each stance. *Argument* is a single comment directly addressing the debate prompt. *Argument pair* is an ordered set of two arguments (*A1* and *A2*) belonging to the same topic; see Figure 1.

We automatically selected debates that contained at least 25 top-level[4] arguments that were 10-110 words long (the mean for all top-level arguments was 66 ± 130 and the median 36, so we excluded the lengthy outliers in our sampling). We manually filtered out obvious silly debates (e.g., *'Superman vs. Batman'*) and ended up with 32 *topics* (the full topic list is presented together with experimental results later in Table 3). From each topic we automatically sampled 25-35 random arguments and created $(n * (n-1)/2)$ *argument pairs* by combining all selected arguments. Sampling argument pairs only from the same topics and not combining opposite stances was a design decision how to mitigate annotators' bias.[5] The order of arguments A1 and A2 in each argument pair was randomly shuffled. In total we sampled 16,927 argument pairs.

3.2 Crowdsourcing annotations

Let us extend our terminology. *Worker* is a single annotator in Amazon Mechanical Turk. *Reason* is an explanation why *A1* is more convincing than *A2* (or the other way round, or why they are equally convincing). *Gold reason* is a reason whose label matches the gold label in the argument pair (see Figure 2).

In the HIT, workers were presented with an argument pair, the prompt, and the stance as in Figure 1. They had to choose either *"A1 is more convincing than A2"* (A1>A2), *"A1 is less convincing than A2"* (A1<A2), or *"A1 and A2 are convincing equally"* (A1=A2). Moreover, they were obliged to write the *reason* 30-140 characters long. An example of fully annotated argument pair is shown in Figure 2. The workers were also provided with clear and crisp instructions (e.g., do not judge the truth of the proposition; be objective; do not express your position; etc.).

All 16,927 argument pairs were annotated by five workers each (85k assignments in total). We also allowed workers to express their own standpoint toward the topics. While 66% of workers had no standpoint, 14% had the opposite view and 20% the same view. This indicates that there should be no systematic bias in the data. Crowdsourcing took about six weeks in 2016 plus two weeks of pilot studies. In total, about 3,900 workers participated. Total costs including pilot studies and bonus payments were 5,520 USD.

3.3 Quality control and agreement

We performed several steps in controlling the quality of the crowdsourced data. First, we allowed only workers from the U.S. with $\geq 96\%$ acceptance rate to work on the task. Second, we employed MACE (Hovy et al., 2013) for estimating

[3] Namely, `createdebate.com` and `procon.org`.

[4] Such arguments directly address the topic and are not a part of a threaded discussion.

[5] As some topics touch the very fundamental human beliefs and values, such as faith, trust, or sexuality, it is hard to put them consciously aside when assessing convincingness.

Argument 1	Argument 2
physical education should be mandatory cuhz 112,000 people have [...]	YES, because some children don't understand anything expect [...]

- A1>A2, *because* A1 uses statistics, and doesn't make assumptions.
- A1>A2, *because* A1 talks about the importance of health.
- A1>A2, *because* A1 provides a health-related argument.
- A1>A2, *because* A2 is very harsh and attacks
- A1=A2, *because* Neither A1 or A2 cite evidence to support their claims.

Figure 2: Example of an *argument pair* annotated by five workers. The arguments are shortened versions of Figure 1. The explanations (called *reasons*) after 'because' are written by workers; the estimated gold label for this pair is probably *A1>A2*, thus there are four *gold reasons*.

the true labels and ranking the annotators. We set the MACE's parameter *threshold* to 0.95 to keep only instances whose entropy is among the 95% best estimates. Third, we manually checked all the *reasons* for each worker. With paying more attention to workers with low MACE scores, we rejected all assignments of workers if they (1) copied&pasted the same or very similar *reasons* across *argument pairs*, (2) were only copying or rephrasing the texts from the arguments, (3) provided their opinion or were arguing, (4) had many typos or provided obvious nonsense. In total, we rejected 1161 assignments.

We do not report any 'standard' inter-annotator agreement measures such as Fleiss' κ or Krippendorff's α, as their suitability for crowdsourcing has been recently disputed (Passonneau and Carpenter, 2014). However, in order to estimate the human performance, we analyzed the output of the pilot study. For each *argument pair*, we took the best-ranked worker for that particular pair (worker ranks are globally estimated by MACE) and computed her accuracy against the estimated gold labels.[6] The best-ranked worker for each argument pair is not necessarily the globally best-ranked worker; in the pilot study, the average global rank of this hypothetical worker was 11 ± 6.6. This rank can be interpreted as a decently performing worker; the obtained score reached 0.935 accuracy.

[6] A similar approach was recently reported by Nakov et al. (2016).

3.4 Examining properties of convincingness

3.4.1 What makes a convincing argument?

We manually examined a small sample of 200 *gold reasons* to find out what makes one argument more convincing than the other. A very common type of answer mentioned **giving examples or actual reasons** (*"A1 cited several reasons to back up their argument."*) and **facts** (*"A1 cites an outside source which can be more credible than opinion"*). This is not surprising, as argumentation is often perceived as reason giving (Freeley and Steinberg, 2008). Others point out strengths in **explaining the reasoning** or **logical coherence** (*"A1 gives a succinct and logical answer to the argument. A2 strays away from the argument in the response."*). The **confirmation bias** (Mercier and Sperber, 2011) also played a role (*"A1 argues both viewpoints, A2 chooses a side."*). Given the noisiness of Web data, some of the arguments might be **non-sense**, which was also pointed out as a reason (*"A1 attempts to argue that since porn exists, we should watch it. A2 doesn't make sense or answer the question."*). Apart from the logical structure of the argument, **emotional aspects** and **rhetorical moves** were also spotted (*"A1 contributes a viewpoint based on morality, which is a stronger argument than A2, which does not argue for anything at all."*, or *"A1 calls for the killing of all politicians, which is an immature knee-jerk reaction to a topic. A2's argument is more intellectually presented."*).

3.4.2 Transitivity evaluation using argument graphs

The previous section shows a variety of reasons that makes one argument more convincing than other arguments. Considering *A1 is more convincing than A2* as a binary relation R, we thus asked the following research question: Is convincingness a measure with total strict order or strict weak order? Namely, is relation R that compares convingcingness of two arguments transitive, antisymmetric, and total?

In particular, does is exhibit properties such that if A$\geq$B and B$\geq$C, then A$\geq$C (total ordering)? We can treat arguments as nodes in a graph and argument pairs as graph edges. We will denote such graph as *argument graph* (and use nodes/arguments and edges/pairs interchangeably in this section).[7] As the sampled argument pairs

[7] Argument pair A>B becomes a directed edge $A \rightarrow B$

contained all argument pair combinations for each topic, we ended up with an almost fully connected argument graph for each topic (remember that we discarded 5% of argument pair annotations with lowest reliability). We further investigate the properties of the argument graphs. Transitivity is only guaranteed, if the argument graph is a DAG (directed acyclic graph).

Building argument graph from crowdsourced argument pairs We build the argument graph iteratively by sampling annotated argument pairs and adding them as graph edges (see Algorithm 1). We consider two possible scenarios in the graph building algorithm. In the first scenario, we accept only argument pairs without equivalency (thus A>B is allowed but A=B is forbidden and discarded). The second scenario accepts all pairs, but since the resulting graph must be DAG, equivalent arguments are merged into one node. We use Johnson's algorithm for finding all elementary cycles in DAG (Johnson, 1975).

Argument pair weights By building argument graph from all pairs, introducing cycles into the graph seems to be inevitable, given a certain amount of noise in the annotations. We asked the following question: to which extent does occurrence of cycles in an argument graph depend on the quality of annotations?

We thus compute a weight for each argument pair. Let e_i be a particular annotation pair (edge). Let G_i be all labels in that pair that match the predicted gold label, and O_i opposite labels (different from the gold label). Let v be a single worker's vote and c_v a global worker's competence score. Then the weight w of edge e_i is computed as follows:

$$w_{e_i} = \sigma\left(\sum_{v \in G_i} c_v - \lambda \sum_{v \in O_i} c_v\right) \qquad (1)$$

where σ is a sigmoid function $\sigma = \frac{1}{1+e^{-x}}$ to squeeze the weight into the $(0, 1)$ interval and λ is a penalty for opposite labels (we set empirically λ to 10.0 to ensure strict penalization). For example, if the predicted gold label from Figure 2 were A1>A2, then G_i would contain four votes and O_i one vote (the last one).

This weight allows us to sort argument pairs before sampling them for building the argument graph. We test three following strategies. As a baseline, we use random shuffling (*Rand*), where no prior information about the weight of the pairs is given. The other two sorting algorithms use the argument pair weight computed by Equation 1. As the worst case scenario, we sort the pairs in ascending order (*Asc*), which means that the "worse" pairs come first to the graph building algorithm. We used this scenario to see how much the prior pair weight information actually matters, because building a graph preferably from bad pair label estimates should cause more harm. Finally, the *Desc* algorithm sorts the pairs given their weight in descending order (the "better" estimates come first).

Algorithm 1: Building DAG from sorted argument pairs.

```
input : argumentPairs; sortingAlg
output: DAG
SortPairs(argumentPairs, sortingAlg);
finalPairs ← [];
foreach pair in argumentPairs do
    currentPairs ← [finalPairs, pair ];
    /* cluster edges labeled as equal so they will be
    treated as a single node */
    clusters ← clusterEqNodes(currentPairs);
    /* wire the pairs into directed graph */
    g ← buildGraph(currentPairs, clusters);
    if hasCycles(g) then
        // report about breaking DAG
    else
        finalPairs += pair;
return buildGraph(finalPairs);
```

Measuring transitivity score We measure how "good" the graph is by a *transitivity score*. Here we assume that the graph is a DAG. Given two nodes A and Z, let P_L be the longest path between these nodes and P_S the shortest path, respectively. For example, let $P_L = A \rightarrow B \rightarrow C \rightarrow Z$ and $P_S = A \rightarrow D \rightarrow Z$. Then the *transitivity score* is the ratio of longest and shortest path $\frac{|P_L|}{|P_S|}$. (which is 1.5 is our example). The *average transitivity score* is then an average of transitivity scores for each pair of nodes from the graph that are connected by two or more paths. Analogically, the *maximum transitivity score* is the maximal value. We restrict the shortest path to be a direct edge only.

The motivation for the *transitivity score* is the following. If the longest path between A and Z ($A \rightarrow \ldots \rightarrow Z$) consists of 10 other nodes, than the total ordering property requires that there also exists a direct edge $A \rightarrow Z$. This is indeed em-

in the argument graph.

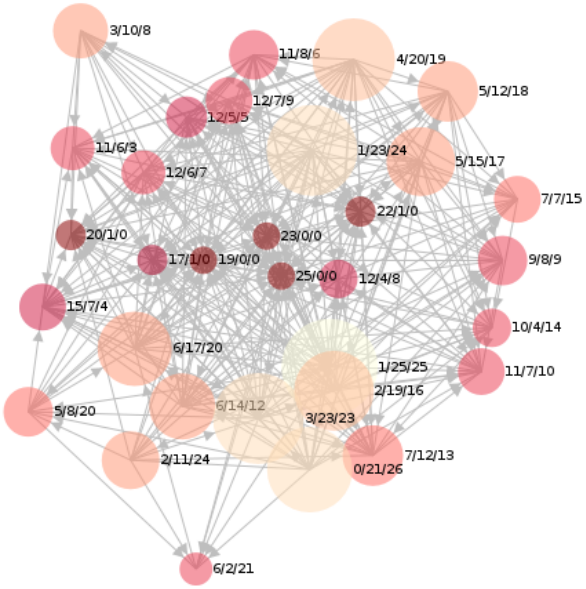


Figure 3: Final argument graph example (topic: *"Christianity or Atheism? Atheism"*). Node labels: *I/O/Tr* (*I*: incoming edges, *O*: outgoing edges, *Tr*: maximum transitivity score). Lighter nodes have prevailing *O*; larger nodes have higher absolute number of *O*.

pirically confirmed by the presence of the shortest path between A and Z. Thus the longer the longest path and the shorter the shortest path are on average, the bigger empirical evidence is given about the transitivity property.

Figure 3 shows an example of argument graph built using only non-equivalent pairs and *desc* prior sort or argument pairs. There are few "bad" arguments in the middle (many incoming edges, none outcoming) and few very convincing arguments (large circles). Notice the high maximum transitivity score even for medium-sized nodes.

Observations First, let us compare the different sorting algorithms for each sampling strategy. As Table 1 shows, on average, 158 pairs are ignored in total when all pairs are used for sampling (26 removed by MACE and 132 by the graph building algorithm), while 164 pairs are ignored when only non-equivalent pairs are sampled (129 had already been removed apriori—26 by MACE and 103 as equivalent pairs—and 35 by the graph algorithm).

The results show a tendency that, when sampling annotated argument pairs for building a DAG, sorting argument pairs by their weight based on workers' scores influences the number of pairs that break the DAG by introducing cycles. In particular, starting with more confident argumentation pairs, the graph grows bigger while keeping its DAG consistency. The presence of the *equal* relation causes cycles to break the DAG sooner as compared to argument pairs in which one argument is more convincing than the other. We interpret this finding as that it is easier for humans to judge A>B than A=B consistently across all possible pairs of arguments from a given topic.

3.4.3 Gold-standard corpora

Our experiments show that convincingness between a pair of arguments exhibits properties of strict total order when the possibility of two equally convincing arguments is prohibited. We thus used the above-mentioned method for graph building as a tool for posterior gold data filtering. We discard the equal argument pairs in advance and filter out argument pairs that break the DAG properties. As a result, a set of 11,650 argument pairs labeled as either A>B or A<B remains, which is summarized in Table 2. We call this corpus *UKPConvArgStrict*.

However, since the total strict ordering property of convincingness is only an empirically confirmed working hypothesis, we also propose another realistic application. We construct a mixed graph by treating equal argument pairs (A=B) as undirected edges. Using PageRank, we rank the arguments (nodes) globally. The higher the PageRank for a particular node is, the "less convincing" the argument is (has a global higher probability of incoming edges). This allows us to rank all arguments for a particular topic. We call this dataset *UKPConvArgRank* (see Table 2).

We also release the full dataset *UKPConvArgAll*. In this data, no global filtering using graph construction methods is applied, only the local pre-filtering using MACE. We believe this dataset can be used as a supporting training data for some tasks that do not rely on the property of total ordering. Along the actual argument texts, all the gold-standard corpora contain the *reasons* as well as full workers' information and debate meta-data.

4 Experiments

We experiment with two machine learning algorithms on two tasks using the two new benchmark corpora (*UKPConvArgStrict* and *UKPConvArgRank*). In both tasks, we perform 32-fold cross-topic cross-validation (one topic is test data, remaining 31 topics are training ones). This rather

		All pairs			No equivalency pairs		
		Rand.	Asc.	Desc.	Rand.	Asc.	Desc.
Fixed values	All annotated pairs	529	529	529	529	529	529
	Pairs removed apriori	26	26	26	129	129	129
Before first cycle detected	Edges in graph	32	16	**86**	84	37	**199**
	Nodes in graph	20	16	**25**	32	28	**33**
	Pairs sampled	44	26	**132**	84	37	**199**
	First cycle length	1,8	2,0	1,5	4,2	3,9	3,4
Final statistics after all pairs sampled	avg. Transitivity score	6,4	5,9	6,8	11,1	11,3	10,8
	max. Transitivity score	14,5	13,2	15,5	24,4	25,6	24,0
	Edges in graph	105	81	114	357	339	365
	Nodes in graph	16	14	16	33	33	33
	Pairs sampled	339	294	**369**	356	339	**364**
	Ignored pairs	162	208	**132**	42	60	**35**

Table 1: Values averaged over all 32 topics reported by different sampling strategies and scenarios for argument graph building.

Dataset	**Size**	**Instance type**	**Size per topic**	**Gold label distribution**			**Gold reasons**	
				a1	a2	eq	**size**	**tokens**
UKPConvArgAll	16,081	argument pair	502.5 ± 91.3	6,398	6,394	3,289	56,446	696,537
UKPConvArgStrict	11,650	argument pair	364.1 ± 71.1	5,872	5,778	0	44,121	547,057
UKPConvArgRank	1,052	argument	32.9 ± 3.2	—			—	

Table 2: Properties of resulting gold data.

challenging setting ensures that no arguments are seen in both training and test data.

4.1 Predicting convincingness of pairs

Since this task is a binary classification and the classes are equally distributed (see Table 2), we report accuracy and average the final score over folds (Forman and Scholz, 2010).

Methods As a "traditional" method, we employ SVM with RBF kernel[8] based on a large set of rich linguistic features. They include **uni- and bi-gram presence**, ratio of **adjective and adverb endings** that may signalize neuroticism (Corney et al., 2002), **contextuality measure** (Heylighen and Dewaele, 2002), **dependency tree depth**, ratio of **exclamation** or **quotation** marks, ratio of **modal verbs**, counts of several **named entity types**, ratio of **past** vs. **future** tense verbs, **POS** n-grams, presence of dependency tree **production rules**, seven different **readability measures** (e.g., *Ari* (Senter and Smith, 1967), *Coleman-Liau* (Coleman and Liau, 1975), *Flesch* (Flesch, 1948), and others), five **sentiment scores** (from very negative to very positive) (Socher et al., 2013), **spell-checking** using standard Unix `words`, ratio of **superlatives**, and some **surface** features such as sentence lengths, longer words count, etc. The resulting feature vector dimension is about 64k.

We also use bidirectional Long Short-Term Memory (BLSTM) neural network for end-to-end processing.[9] The input layer relies on pre-trained word embeddings, in particular *GloVe* (Pennington et al., 2014) trained on 840B tokens from Common Crawl;[10] the embedding weights are further updated during training. The core of the model consists of two bi-directional LSTM networks with 64 output neurons each. Their output is then concatenated into a single drop-out layer and passed to the final sigmoid layer for binary predictions. We train the network with ADAM optimizer (Kingma and Ba, 2015) using binary cross-entropy loss function and regularize by early stopping (5 training epochs) and high drop-out rate (0.5) in the dropout layer. For both models, each training/test instance simply concatenates A1 and A2 from the argument pair.

Results and error analysis As shown in Table 3, SVM (0.78) outperforms BSLTM (0.76) with a subtle but significant difference. It is also apparent that some topics are more challenging regardless of the system (e.g., *"Is it better to have a lousy father or to be fatherless? – Lousy father"*). Both systems outperform a simple baseline (lemma n-gram presence features with SVM, not reported in detail, achieved 0.65 accuracy) but still do not reach the human upper bounds (0.93 as reported in Section 3.3).

[8] Using LISBVM (Chang and Lin, 2011).

[9] Using `http://keras.io/`

[10] `http://nlp.stanford.edu/projects/glove/`

	Topic	SVM	BLSTM
Ban Plastic Water Bottles?	No	.85	.76
	Yes	.90	.83
Christianity or Atheism	Atheism	.81	.80
	Christianity	.68	.75
Evolution vs. Creation	Creation	.84	.88
	Evolution	.66	.77
Firefox vs. Internet Explorer	IE	.84	.81
	Firefox	.82	.78
Gay marriage - right or wrong?	Right	.76	.74
	Wrong	.82	.87
Should parents use spanking?	No	.84	.78
	Yes	.79	.68
If your spouse committed murder [...]	No	.71	.64
	Yes	.79	.72
India has the potential to lead the world	No	.82	.77
	Yes	.69	.79
Is it better to have a lousy father or to be fatherless?	Fatherless	.77	.69
	Lousy father	.67	.60
Is porn wrong?	No	.82	.79
	Yes	.85	.85
Is the school uniform a good or bad idea?	Bad	.75	.78
	Good	.83	.74
Pro choice vs. Pro life	Choice	.71	.68
	Life	.79	.80
Should physical edu. be mandatory?	No	.79	.80
	Yes	.79	.78
TV is better than books	No	.78	.73
	Yes	.78	.75
Personal pursuit or common good?	Common	.72	.78
	Personal	.67	.68
Farquhar as the founder of Singapore	No	.79	.63
	Yes	.85	.76
Average		**.78**	**.76**

Table 3: Accuracy results on *UKPConvArgStrict* data. The difference between **SVM** and bidirectional **LSTM** is significant, $p = 0.0414$ using two-tailed Wilcoxon signed-rank test.

We examined about fifty random false predictions to gain some insight into the limitations of both systems. We looked into argument pairs, in which both methods failed, as well as into instances where only one model was correct. BLSTM won in few cases by properly catching jokes or off-topic arguments; SVM was properly catching all-upper-case arguments (considered as less convincing). By examining failures common to both systems, we found several cases where the prediction was wrong due to very negative sentiment (which might be a sign of the less convincing argument), but in other cases an argument with strong negative sentiment was actually the more convincing one. In general, we did not find any tendency on failures; they were also independent of the worker assignments distribution, thus not caused by likely ambiguous (hard) instances.

	SVM	BLSTM	p-value
Pearson's r	.351	.270	$\ll 0.01$
Spearman's ρ	.402	.354	$\ll 0.01$

Table 4: Correlation results on *UKPConvArgRank*.

4.2 Ranking arguments

We address this problem as a regression task. We use the *UKPConvArgRank* data, in which a real-value score is assigned to each argument so the arguments can be ranked by their convincingness (for each topic independently). The task is thus to predict a real-value score for each argument from the test topic (remember that we use 32-fold cross validation). We measure Spearman's and Pearson's correlation coefficients on all results combined (not on each fold separately).

Without any modifications, we use the same SVM and features as described in Section 4.1. Regarding the BLSTM, we only replace the output layer with a linear activation function and optimize mean absolute error loss. Table 4 shows that SVM outperforms BLSTM. All correlations are highly statistically significant.

4.3 Results discussion

Although the "traditional" SVM with rich linguistic features outperforms BLSTM in both tasks, there are other aspects to be considered. First, the employed features require heavy language-specific preprocessing machinery (lemmatizer, POS tagger, parser, NER, sentiment analyzer). By contrast, BLSTM only requires pre-trained embedding vectors, while delivering comparable results. Second, we only experimented with vanilla LSTMs. Recent developments of deep neural networks (especially attention mechanisms or grid-LSTMs) open up many future possibilities to gain performance in this end-to-end task.

5 Conclusion and future work

We propose a novel task of predicting Web argument convincingness. We crowdsourced a large corpus of 16k argument pairs over 32 topics and used global constraints based on transitivity properties of convincingness relation for cleaning the data. We experimented with feature-rich SVM and bidirectional LSTM and obtain 0.76-0.78 accuracy and 0.35-0.40 Spearman's correlation in a cross-topic scenario. We release the newly created corpus *UKPConvArg1* and the experimental software

under free licenses.[11] To the best of our knowledge, we are the first who deal with argument convincingness in Web data on such a large scale.

In the current article, we have only slightly touched the annotated natural text *reasons*. We believe that the presence of 44k reasons (550k tokens) is another important asset of the newly created corpus, which deserves future investigation.

Acknowledgements

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant Nº I/82806, by the German Institute for Educational Research (DIPF), by the German Research Foundation (DFG) via the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1), by the GRK 1994 AIPHES (DFG), and by Amazon Web Services in Education Grant award. Lastly, we would like to thank the anonymous reviewers for their valuable feedback.

References

Nora Aranberri, Gorka Labaka, Arantza Díaz de Ilarraza, and Kepa Sarasola. 2016. Ebaluatoia: crowd evaluation for English-Basque machine translation. *Language Resources and Evaluation*. In press.

Aristotle and George Kennedy (translator). 1991. *On Rhetoric: A Theory of Civil Discourse*. Oxford University Press, USA.

J. Anthony Blair. 2011. Argumentation as rational persuasion. *Argumentation*, 26(1):71–81.

Maarten Boudry, Fabio Paglieri, and Massimo Pigliucci. 2015. The Fake, the Flimsy, and the Fallacious: Demarcating Arguments in Real Life. *Argumentation*, 29(4):431–456.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 129–136, New York, NY, USA. ACM.

[11] https://github.com/UKPLab/acl2016-convincing-arguments

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, page 193, New York, New York, USA. ACM Press.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. *arXiv*.

Meri Coleman and T. L. Liau. 1975. A computer readability formula designed for machine scoring. *Journal of Applied Psychology*, 60:283–284.

Malcolm Corney, Olivier de Vel, Alison Anderson, and George Mohay. 2002. Gender-preferential text mining of e-mail discourse. In *Proceedings of the 18th Annual Computer Security Applications Conference (ACSAC02)*, pages 282–289.

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

Rudolf Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32:221–233.

George Forman and Martin Scholz. 2010. Apples-to-Apples in Cross-Validation Studies: Pitfalls in Classifier Performance Measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57.

Austin J. Freeley and David L. Steinberg. 2008. *Argumentation and Debate*. Cengage Learning, Stamford, CT, USA, 12th edition.

Merce Garcia-Mila, Sandra Gilabert, Sibel Erduran, and Mark Felton. 2013. The effect of argumentative task goal on the quality of argumentative discourse. *Science Education*, 97(4):497–523.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep learning. Book in preparation for MIT Press.

Ivan Habernal and Iryna Gurevych. 2015. Exploiting debate portals for semi-supervised argumentation mining in user-generated web discourse. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2127–2137, Lisbon, Portugal, September. Association for Computational Linguistics.

Ivan Habernal and Iryna Gurevych. 2016. Argumentation Mining in User-Generated Web Discourse. *Computational Linguistics*. Under review. http://arxiv.org/abs/1601.02403.

Francis Heylighen and Jean-Marc Dewaele. 2002. Variation in the contextuality of language: An empirical measure. *Foundations of Science*, 7(3):293–340.

Dirk Hovy, Taylor Berg-Kirkpatrick, Ashish Vaswani, and Eduard Hovy. 2013. Learning Whom to Trust with MACE. In *Proceedings of NAACL-HLT 2013*, pages 1120–1130, Atlanta, Georgia. Association for Computational Linguistics.

Ralph H. Johnson and Anthony J. Blair. 2006. *Logical Self-Defense*. International Debate Education Association.

Donald B. Johnson. 1975. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference for Learning Representations*, San Diego, CA.

Hugo Mercier and Dan Sperber. 2011. Why do humans reason? Arguments for an argumentative theory. *The Behavioral and Brain Sciences*, 34(2):57–74; discussion 74–111.

Preslav Nakov, Sara Rosenthal, Svetlana Kiritchenko, Saif M. Mohammad, Zornitsa Kozareva, Alan Ritter, Veselin Stoyanov, and Xiaodan Zhu. 2016. Developing a successful SemEval task in sentiment analysis of Twitter and other social media texts. *Language Resources and Evaluation*, 50(1):35–65.

Ana Laura Nettel and Georges Roque. 2011. Persuasive argumentation versus manipulation. *Argumentation*, 26(1):55–69.

Daniel J. OKeefe. 2011. Conviction, persuasion, and argumentation: Untangling the ends and means of influence. *Argumentation*, 26(1):19–32.

Joonsuk Park and Claire Cardie. 2014. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, Baltimore, Maryland, June. Association for Computational Linguistics.

Rebecca J. Passonneau and Bob Carpenter. 2014. The Benefits of a Model of Annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326.

Andreas Peldszus and Manfred Stede. 2015. Joint prediction in MST-style discourse parsing for argumentation mining. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 938–948, Lisbon, Portugal, September. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552, Beijing, China, July. Association for Computational Linguistics.

Chris Piech, Jonathan Bassen, Jonathan Huang, Surya Ganguli, Mehran Sahami, Leonidas J. Guibas, and Jascha Sohl-Dickstein. 2015. Deep Knowledge Tracing. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 505—-513, Montreal, CA. Curran Associates, Inc.

Ruty Rinott, Lena Dankin, Carlos Alzate Perez, Mitesh M. Khapra, Ehud Aharoni, and Noam Slonim. 2015. Show me your evidence - an automatic method for context dependent evidence detection. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 440–450, Lisbon, Portugal, September. Association for Computational Linguistics.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the 2016 International Conference on Learning Representations (ICLR)*, pages 1–9.

Edward Schiappa and John P. Nordin. 2013. *Argumentation: Keeping Faith with Reason*. Pearson UK, 1st edition.

J. R. Senter and E. A. Smith. 1967. Automated readability index. Technical report AMRL-TR-66-220, Aerospace Medical Research Laboratories, Ohio.

Nihar B. Shah, Sivaraman Balakrishnan, Joseph K. Bradley, Abhay Parekh, Kannan Ramchandran, and Martin J. Wainwright. 2015. Estimation from Pairwise Comparisons: Sharp Minimax Bounds with Topology Dependence. *arXiv preprint*, abs/1505.01462.

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

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 46–56, Doha, Qatar, October. Association for Computational Linguistics.

Karsten Stegmann, Christof Wecker, Armin Weinberger, and Frank Fischer. 2011. Collaborative argumentation and cognitive elaboration in a computer-supported collaborative learning environment. *Instructional Science*, 40(2):297–323.

Douglas N. Walton. 1989. *Informal Logic: A Handbook for Critical Argument*. Cambridge University Press.

Shuohang Wang and Jing Jiang. 2016. Learning Natural Language Inference with LSTM. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page (to appear), San Diego, CA, June. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016. Multi-domain Neural Network Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page (to appear).

Discovery of Treatments from Text Corpora

Christian Fong
Stanford University
655 Serra Street
Stanford, CA 94305, USA
christianfong@stanford.edu

Justin Grimmer
Stanford University
616 Serra Street
Stanford, CA 94305, USA
jgrimmer@stanford.edu

Abstract

An extensive literature in computational social science examines how features of messages, advertisements, and other corpora affect individuals' decisions, but these analyses must specify the relevant features of the text before the experiment. Automated text analysis methods are able to discover features of text, but these methods cannot be used to obtain the estimates of causal effects—the quantity of interest for applied researchers. We introduce a new experimental design and statistical model to simultaneously discover treatments in a corpora and estimate causal effects for these discovered treatments. We prove the conditions to identify the treatment effects of texts and introduce the supervised Indian Buffet process to discover those treatments. Our method enables us to discover treatments in a training set using a collection of texts and individuals' responses to those texts, and then estimate the effects of these interventions in a test set of new texts and survey respondents. We apply the model to an experiment about candidate biographies, recovering intuitive features of voters' decisions and revealing a penalty for lawyers and a bonus for military service.

1 Introduction

Computational social scientists are often interested in inferring how blocks of text, such as messages from political candidates or advertising content, affect individuals' decisions (Ansolabehere and Iyengar, 1995; Mutz, 2011; Tomz and Weeks, 2013). To do so, they typically attempt to estimate the causal effect of the text: they model the outcome of interest, Y, as a function of the block of text presented to the respondent, t, and define the treatment effect of t relative to some other block of text t' as $Y(t) - Y(t')$ (Rubin, 1974; Holland, 1986). For example, in industrial contexts researchers design A/B tests to compare two potential texts for a use case. Academic researchers often design one text that has a feature of interest and another text that lacks that feature but is otherwise identical (for example, (Albertson and Gadarian, 2015)). Both kinds of experiments assume researchers already know the features of text to vary and offer little help to researchers who would like to discover the features to vary.

Topic models and related methods can discover important features in corpora of text data, but they are constructed in a way that makes it difficult to use the discovered features to estimate causal effects (Blei et al., 2003). Consider, for example, supervised latent Dirichlet allocation (sLDA) (Mcauliffe and Blei, 2007). It associates a topic-prevalence vector, $\boldsymbol{\theta}$, with each document where the estimated topics depend upon both the content of documents and a label associated with each document. If K topics are included in the model, then $\boldsymbol{\theta}$ is defined on the $K-1$-dimensional unit simplex. It is straightforward to define a treatment effect as the difference between two treatments $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ (or points on the simplex) $Y(\boldsymbol{\theta}) - Y(\boldsymbol{\theta}')$. It is less clear how to define the *marginal* effect of any one dimension. This is because bigger values on some dimensions implies smaller values on other dimensions, making the effect of any one topic necessarily a combination of the differences obtained when averaging across all the dimensions (Aitchison, 1986; Katz and King, 1999). This problem will befall all topic models because the zero-sum nature of the topic-prevalence vector implies that increasing the prevalence of any one topic necessarily decreases the prevalence of some

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1600–1609,
Berlin, Germany, August 7-12, 2016.

other topic. The result is that it is difficult (or impossible) to interpret the effect of any one topic marginalizing over the other topics. Other applications of topic models to estimate causal effects treat text as the response, rather than the treatment (Roberts et al., 2016). And still other methods require a difficult to interpret assumption of how text might affect individuals' responses (Beauchamp, 2011).

To facilitate the discovery of treatments and to address the limitation of existing unsupervised learning methods, we introduce a new experimental design, framework, and statistical model for discovering treatments within blocks of text and then reliably inferring the effects of those treatments. By doing so, we combine the utility of discovering important features in a topic model with the scientific value of causal treatment effects estimated in a potential outcomes framework. We present a new statistical model—the supervised Indian Buffet Process—to both discover treatments in a training set and infer the effects treatments in a test set (Ghahramani and Griffiths, 2005). We prove that randomly assigning blocks of text to respondents in an experiment is sufficient to identify the effects of latent treatments that comprise blocks of text.

Our framework provides the first of its kind approach to automatically discover treatment effects in text, building on literatures in both social science and machine learning (Blei et al., 2003; Beauchamp, 2011; Mcauliffe and Blei, 2007; Roberts et al., 2016). The use of the training and test set ensures that this discovery does not come at the expense of credibly inferring causal effects, insulating the research design from concerns about "p-hacking" and overfitting (Ioannidis, 2005; Humphreys et al., 2013; Franco et al., 2014). Critically, we use a theoretical justification for our methodology: we select our particular approach because it enables us to estimate causal effect of interest. Rather than demonstrating that our method performs better at some predictive task, we prove that our method is able to estimate useful causal effects from the data.

We apply our framework to study how features of a political candidate's background affect voters' decisions. We use a collection of candidate biographies collected from Wikipedia to automatically discover treatments in the biographies and then infer their effects. This reveals a penalty for lawyers and career politicians and a bonus for military service and advanced degrees. While we describe our procedure throughout the paper, we summarize our experimental protocol and strategy for discovering treatment effects in Table 1.

Table 1: Experimental Protocol for Discovering and Estimating Treatment Effects

1) Randomly assign texts, $\boldsymbol{X}_j$, to respondents
2) Obtain response Y_i for each respondent.
3) Divide texts and responses into training and test set
4) In training set:
    a) Use supervised Indian Buffet Process (sIBP) applied to documents and responses to infer latent treatments in texts
    b) Model selection via quantitative fit and qualitative assessment
5) In test set:
    a) Use sIBP trained on training set to infer latent treatments on test set documents
    b) Estimate effect of treatments with regression, with a bootstrap procedure to estimate uncertainty

2 A Framework for Discovering Treatments from Text

Our goal is to discover a set of features—treatments—underlying texts and then estimate the effect of those treatments on some response from an individual. We first show that randomly assigning texts to respondents is sufficient to identify treatment effects. We then provide a statistical model for using both the text and responses to discover latent features in the text that affect the response. Finally, we show that we can use the mapping from text to features discovered on a training set to estimate the presence of features in a test set, which allows us to estimate treatment effects in the test set.

2.1 Randomizing Texts Identifies Underlying Treatment Effects

When estimating treatment effects, researchers often worry that the respondents who received one treatment systematically differ from those who received some other treatment. In a study of advertising, if all of the people who saw one advertisement were men and all of the people who saw a different advertisement were women, it would be impossible to tell whether differences in their responses were driven by the fact that they saw different advertisements or by their pre-existing differences. Randomized experiments are the gold standard for overcoming this problem (Gerber and Green, 2012). However, in text experiments, individuals are randomly assigned to blocks of text rather than to the latent features of the text that we analyze as the treatments. In this section, we show that randomly assigning blocks of text is sufficient to identify treatment effects.

To establish our result, we suppose we have a corpora of J texts, $\mathcal{X}$. We represent a specific text with $\boldsymbol{X}_j \in \mathcal{X}$, with $\boldsymbol{X}_j \in \Re^D$. Throughout we will assume that we have standardized the variable $\boldsymbol{X}_j$ to be a per-document word usage rate with each column normalized to have mean zero and variance one. We have a sample of N respondents from a population, with the response of individual i to text $j[i]$ given by the potential outcome $Y_i(\boldsymbol{X}_{j[i]})$. We use the notation $j[i]$ because multiple individuals may be assigned to the same text; if i and i' are assigned to the same text, then $j[i] = j[i']$. We suppose that for each document j there is a corresponding vector of K binary treatments $\boldsymbol{Z}_j \in \mathcal{Z}$ where $\mathcal{Z}$ contains all 2^K possible combinations of treatments, $\{0,1\}^K$. The function $g : \mathcal{X} \rightarrow \mathcal{Z}$ maps from the texts to the set of binary treatments: we will learn this function using the supervised Indian Buffet process introduced in the next section. Note that distinct elements of $\mathcal{X}$ may map to the same element of $\boldsymbol{Z}$.

To establish our identification result, we assume (Assumption 1) $Y_i(\boldsymbol{X}) = Y_i(\boldsymbol{X}_{j[i]})$ for all i. This assumption ensures that each respondent's treatment assignment depends only on her assigned text, a version of the Stable Unit Treatment Value Assumption (SUTVA) for our application (Rubin, 1986). We also assume (Assumption 2) that $Y_i(\boldsymbol{X}_{j[i]}) = Y_i(g(\boldsymbol{X}_{j[i]}))$ for all $\boldsymbol{X}_{j[i]} \in \mathcal{X}$ and all i, or that $\boldsymbol{Z}_{j[i]}$ is sufficent to describe the effect of a document on individual i's response. Stated differently, we assume an individual would respond in the same way to two different texts if those texts have the same latent features. We further suppose (Assumption 3) that texts are randomly assigned to respondents according to probability measure h, ensuring that $Y_i(g(\boldsymbol{X}_{j[i]})) \perp\!\!\!\perp \boldsymbol{X}_{j[i]}$ for all $\boldsymbol{X}_{j[i]} \in \mathcal{X}$ and for all individuals i. This assumption ensures unobserved characteristics of individuals are not confounding inferences about the effects of texts. The random assignment of texts to individuals induces a distribution over a probability measure on treatment vectors $\boldsymbol{Z}$, $f(\boldsymbol{Z}) = \int_{\mathcal{X}} 1(\boldsymbol{Z} = g(\boldsymbol{X}))h(\boldsymbol{X})d\boldsymbol{X}$. Finally, we assume (Assumption 4) that $f(\boldsymbol{Z}) > 0$ for all $\boldsymbol{Z} \in \mathcal{Z}$.[1] This requires that every combination of treatment effects is possible from the documents in our corpus. In practice, when designing our study we want to ensure that the treatments are not *aliased* or perfectly correlated. If perfect correlation exists between factors, we are unable to disentangle the effect of individual factors.

In this paper we focus on estimating the Average Marginal Component Specific Effect for factor k (AMCE$_k$) (Hainmueller et al., 2014).[2] The AMCE$_k$ is useful for finding the effect of one feature, k, when k interacts with the other features in some potentially complicated way. It is defined as the difference in outcomes when the feature is present and when it is not present, averaged over the values of all of the other features. Formally, $\text{AMCE}_k = \int_{\boldsymbol{Z}_{-k}} \mathbb{E}\left[Y(Z_k = 1, \boldsymbol{Z}_{-k}) - Y(Z_k = 0, \boldsymbol{Z}_{-k})\right] m(\boldsymbol{Z}_{-k}) d\boldsymbol{Z}_{-k}$ where $m(\boldsymbol{Z}_{-k})$ is some analyst-defined density on all elements but k of the treatment vector. For example, $m(\cdot)$ can be chosen as the density of Z_{-k} in the population to obtain the marginal component effect of k in the empirical population. The most commonly used $m(\cdot)$ in applied work is uniform across all Z_{-k}'s, and we follow this convention here.

We now prove that assumptions 1, 2, 3, and 4 are sufficient to identify the AMCE$_k$ for all k.

Proposition 1. *Assumptions 1, 2, 3, and 4 are suf-*

[1] Note for this assumption to hold it is necessary, but not sufficient that g is a surjection from $\mathcal{X}$ onto $\mathcal{Z}$.

[2] The procedure here can be understood as a method for discovering the treatments that are imposed by assumption in conjoint analysis, as presented by (Hainmueller et al., 2014). We deploy the regression estimator used in conjoint analysis as a subroutine of our procedure (see Step 5b in Table 1), but otherwise our experimental design, statistical method, and proof is distinct.

ficient to identify the AMCE$_k$ for arbitrary k.

Proof. To obtain a useful form, we first marginalize over the documents to obtain, $\int_{\boldsymbol{Z}_{-k}} \int_{\mathcal{X}} \mathbb{E}\left[Y(Z_k = 1, \boldsymbol{Z}_{-k})\right] f(\boldsymbol{Z}_{-k}|Z_k = 1, \boldsymbol{X}) - \mathbb{E}\left[Y(Z_k = 0, \boldsymbol{Z}_{-k})\right] f(\boldsymbol{Z}_{-k}|Z_k = 0, \boldsymbol{X}) h(\boldsymbol{X}) d\boldsymbol{X} d\boldsymbol{Z}_{-k} =$

$$\int_{\boldsymbol{Z}_{-k}} \mathbb{E}\left[Y(Z_k = 1, \boldsymbol{Z}_{-k})\right] f(\boldsymbol{Z}_{-k}|Z_k = 1) - \mathbb{E}\left[Y(Z_k = 0, \boldsymbol{Z}_{-k})\right] f(\boldsymbol{Z}_{-k}|Z_k = 0) d\boldsymbol{Z}_{-k}$$

.

Where $f(\boldsymbol{Z}_{-k}|Z_k = 1)$ and $f(\boldsymbol{Z}_{-k}|Z_k = 0)$ are the induced distributions over latent features from averaging over documents. If $f(\boldsymbol{Z}_{-k}|Z_k = 0) = f(\boldsymbol{Z}_{-k}|Z_k = 1) = m(\boldsymbol{Z}_{-k})$ then this is the AMCE$_k$. Otherwise consider $m(\boldsymbol{Z}) > 0$ for all $\boldsymbol{Z} \in \mathcal{Z}$. Because $f(\boldsymbol{Z}) > 0$, $f(\boldsymbol{Z}_{-k}|Z_k = 0) > 0$ and $f(\boldsymbol{Z}_{-k}|Z_k = 1) > 0$. Thus, there exists conditional densities $h(\boldsymbol{Z}|Z_k = 1)$ and $h(\boldsymbol{Z}|Z_k = 0)$ such that $\frac{f(\boldsymbol{Z}_{-k}|Z_k=1)}{h(\boldsymbol{Z}_{-k}|Z_k=1)} = \frac{f(\boldsymbol{Z}_{-k}|Z_k=0)}{h(Z_{-k}|Z_k=0)} = m(\boldsymbol{Z}_{-k})$ □

2.2 A Statistical Model for Identifying Features

The preceding section shows that if we are able to discover features in the data, we can estimate their AMCEs by randomly assigning texts to respondents. We now present a statistical model for discovering those features. As we argued in the introduction, it is difficult to use the topics obtained from topic models like sLDA because the topic vector exists on the simplex. When we compare the outcomes associated with two different topic vectors, we do not know whether the change in the response is caused by increasing the degree to which the document about one topic or decreasing the degree to which it is about another, because the former mathematically entails the latter. Other models, such as LASSO regression, would necessarily suppose that the presence and absence of words are the treatments (Hastie et al., 2001; Beauchamp, 2011). This is problematic substantively, because it is hard to know exactly what the presence or absence of a single word implies as a treatment in text.

We therefore develop the supervised Indian Buffet Process (sIBP) to discover features in the document. For our purposes, the sIBP has two essential properties. First, it produces a binary topic vector, avoiding the complications of treatments assigned on the simplex. Second, unlike the Indian Buffet Process upon which it builds (Ghahramani and Griffiths, 2005), it incorporates information about the outcome associated with various texts, and therefore discovers features that explain both the text and the response.[3]

Figure 1 describes the posterior distribution for the sIBP and a summary of the posterior is given in Equation 1. We describe the model in three steps: the treatment assignment process, document creation, and response. The result is a model that creates a link between document content and response through a vector of treatment assignments.

Treatment Assignment We assume that $\boldsymbol{\pi}$ is a K-vector (where we take the limit as $K \to \infty$) where π_k describes the population proportion of documents that contain latent feature k. We suppose that $\boldsymbol{\pi}$ is generated by the stick-breaking construction (Doshi-Velez et al., 2009). Specifically, we suppose that $\eta_k \sim \text{Beta}(\alpha, 1)$ for all K. We label $\pi_1 = \eta_1$ and for each remaining topic, we assume that $\pi_k = \prod_{z=1}^{k} \eta_z$. For document j and topic k, we suppose that $z_{j,k} \sim \text{Bernoulli}(\pi_k)$, which importantly implies that the occurrence of treatments are not zero sum. We collect the treatment vector for document j into $\boldsymbol{Z}_j$ and collect all the treatment vectors into $\mathbf{Z}$ an $N_{\text{texts}} \times K$ binary matrix, where N_{texts} refers to number of unique documents. Throughout we will assume that $N_{\text{texts}} = N$ or that the number of documents and responses are equal and index the documents with i.

Document Creation We suppose that the documents are created as a combination of latent factors. For topic k we suppose that $\boldsymbol{A}_k$ is a $D-$dimensional vector that maps latent features onto observed text. We collect the vectors into $\boldsymbol{A}$, a $K \times D$ matrix, and suppose that $\boldsymbol{X}_i \sim \text{MVN}(\boldsymbol{Z}_i \boldsymbol{A}, \sigma_n^2 I_D)$, where $X_{i,d}$ is the standardized number of times word d appears in document i. While it is common to model texts as draws from multinomial distributions, the multi-

[3] We note that there is a different model also called the supervised Indian Buffet Process (Quadrianto et al., 2013). There are fundamental differences between the model presented here and the sIBP in (Quadrianto et al., 2013). Their outcome is a preference relation tuple, while ours is a real-valued scalar. Because of this difference, the two models are fundamentally different. This leads to a distinct data generating process, model inference procedures, and inferences of features on the test set. To leverage the analogy between LDA and sLDA vis a vis IBP and sIBP, we overload the term sIBP in our paper. We expect that in future applications of sIBP, it will be clear from the context which sIBP is being employed.

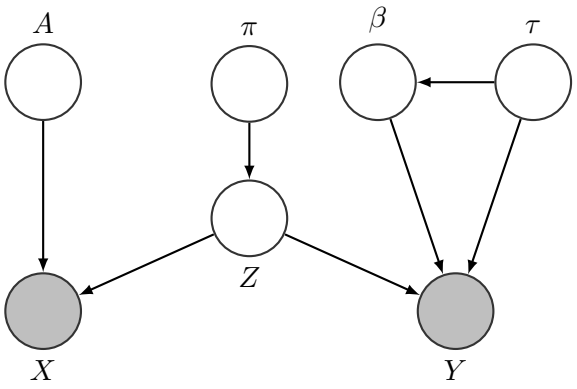


Figure 1: Graphical Model for the Supervised Indian Buffet Process

variate normal distribution is useful for our purposes for two reasons. First, we normalize our data by transforming each column $\boldsymbol{X}_{\cdot,d}$ to be mean 0 and variance 1, ensuring that the multivariate normal distribution captures the overall contours of the data. Note that this implies that $X_{i,d}$ can be negative. Second, we show that assuming a multivariate normal for document generation results in parameters that capture the distinctive rate words are used for each latent feature (Doshi-Velez et al., 2009).

Response to Treatment Vector We assume that a $K-$vector of parameters $\boldsymbol{\beta}$ describes the relationship between the treatment vector and response. Specifically, we use a standard parameterization and suppose that $\tau \sim \text{Gamma}(a, b)$, $\boldsymbol{\beta} \sim \text{MVN}(0, \tau^{-1})$ and that $Y_i \sim \text{Normal}(\boldsymbol{Z}_i\boldsymbol{\beta}, \tau^{-1})$.

$$\begin{aligned}
\pi_k &\sim \text{Stick-Breaking}\,(\alpha) \\
z_{i,k} &\sim \text{Bernoulli}(\pi_k) \\
\mathbf{X}_i|\mathbf{Z}_i, \mathbf{A} &\sim \text{MVN}(\mathbf{Z}_i\mathbf{A}, \sigma_X^2 I_D) \\
\mathbf{A}_k &\sim \text{MVN}(\mathbf{0}, \sigma_A^2 I_D) \\
\mathbf{Y}_i|\mathbf{Z}_i, \boldsymbol{\beta} &\sim \text{Normal}(\mathbf{Z}_i\boldsymbol{\beta}, \tau^{-1}) \\
\tau &\sim \text{Gamma}(a, b) \\
\boldsymbol{\beta}|\tau &\sim \text{MVN}(\mathbf{0}, \tau^{-1} I_K) \qquad (1)
\end{aligned}$$

2.2.1 Inference for the Supervised Indian Buffet Process

We approximate the posterior distribution with a variational approximation, building on the algorithm introduced in (Doshi-Velez et al., 2009). We approximate the non-parametric posterior setting K to be large and use a factorized approximation, assuming that $p(\boldsymbol{\pi}, \boldsymbol{Z}, \boldsymbol{A}, \boldsymbol{\beta}, \tau|\boldsymbol{X}, \boldsymbol{Y}, \alpha, \sigma_A^2, \sigma_X^2, a, b) = q(\boldsymbol{\pi})q(\boldsymbol{A})q(\boldsymbol{Z})q(\boldsymbol{\beta}, \tau)$

A standard derivation that builds on (Doshi-Velez et al., 2009) leads to the following distributional forms and update steps:

- $q(\pi_K) = \text{Beta}(\pi_k|\boldsymbol{\lambda}_k)$. The update values are $\lambda_{k,1} = \frac{\alpha}{K} + \sum_{i=1}^N \nu_{i,k}$ and $\lambda_{k,2} = 1 + \sum_{i=1}^N (1 - \nu_{i,k})$.
- $q(\boldsymbol{A}_k) = \text{Multivariate Normal}(\boldsymbol{A}_k|\bar{\boldsymbol{\phi}}_k, \boldsymbol{\Phi}_k)$. The updated parameter values are,

  $$\bar{\phi}_k = \left[\frac{1}{\sigma_X^2}\sum_{i=1}^N \nu_{i,k}\left(\mathbf{X}_i - \left(\sum_{l:l\neq k}\nu_{i,l}\bar{\phi}_l\right)\right)\right]\boldsymbol{\Phi}_k$$

  $$\boldsymbol{\Phi}_k = \left(\frac{1}{\sigma_A^2} + \frac{\sum_{i=1}^N \nu_{i,k}}{\sigma_X^2}\right)^{-1} I$$

- $q(\boldsymbol{\beta}, \boldsymbol{\tau}) = \text{Multivariate Normal}(\boldsymbol{\beta}|\boldsymbol{m}, \boldsymbol{S}) \times \text{Gamma}(\tau|c, d)$. The updated parameter values are,

  $$\begin{aligned}
  \boldsymbol{m} &= \mathbf{S}\mathbb{E}[\mathbf{Z}^T]\mathbf{Y} \\
  \boldsymbol{S} &= (\mathbb{E}[\mathbf{Z}^T\mathbf{Z}] + I_K)^{-1} \\
  c &= a + \frac{N}{2} \\
  d &= b + \frac{\mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbb{E}[\mathbf{Z}]\mathbf{S}\mathbb{E}[\mathbf{Z}^T]\mathbf{Y}}{2}
  \end{aligned}$$

  Where typical element of $\mathbb{E}[\boldsymbol{Z}^T]_{j,k} = \nu_{j,k}$ and typical on-diagonal element of $\mathbb{E}[\boldsymbol{Z}^T\boldsymbol{Z}]_{k,k} = \sum_{i=1}^N \nu_{i,k}$ and off-diagonal element is $\mathbb{E}[\boldsymbol{Z}^T\boldsymbol{Z}]_{j,k} = \sum_{i=1}^N \nu_{i,j}\nu_{i,k}$.
- $q(z_{i,k}) = \text{Bernoulli}(z_{i,k}|\nu_{i,k})$. The updated parameter values are
  $v_{i,k} = \psi(\lambda_{k,1}) - \psi(\lambda_{k,2}) - \frac{1}{2\sigma_X^2}[-2\bar{\phi}_k\mathbf{X}_i^T$
  $+(\text{tr}(\boldsymbol{\Phi}_k) + \bar{\phi}_k\bar{\phi}_k^T) + 2\bar{\phi}_k\left(\sum_{l:l\neq k}\nu_{i,l}\bar{\phi}_l^T\right)]$
  $-\frac{c}{2d}(-2m_kY_i + \left(\frac{dS_{k,k}}{c-1} + m_k^Tm_k\right) +$
  $2m_k\left(\sum_{l:l\neq k}\nu_{i,l}m_l\right))$
  $\nu_{i,k} = \frac{1}{1+\exp(-v_{i,k})}$
  where $\psi(\cdot)$ is the digamma function. We repeat the algorithm until the change in the parameter vector drops below a threshold.

To select the final model using the training set data, we perform a two-dimensional line search over values of α and σ_X.[4] We then run the model

[4]We assign σ_A, a, and b values which lead to diffuse priors.

several times for each combination of values for α and σ_X to evaluate the output at several different local modes. To create a candidate set of models, we use a quantitative measure that balances coherence and exclusivity (Mimno et al., 2011; Roberts et al., 2014). Let $\mathcal{I}_k$ be the set of documents for which $\nu_{i,k} \geq 0.5$, and let $\mathcal{I}_k^C$ be the complement of this set. We identify the top ten words for intervention k as the ten words with the largest value in $\boldsymbol{A}_k$, $\boldsymbol{t}_k$ and define $N_k = \sum_{i=1}^{N} I\{\nu_{i,k} \geq 0.5\}$. We then obtain measure CE for a particular model $\text{CE} = \sum_{k=1}^{K} N_k \sum_{l,c \in t_k} \text{cov}(\boldsymbol{X}_{\mathcal{I}_k,l}, \boldsymbol{X}_{\mathcal{I}_k,c}) - \sum_{k=1}^{K} (N - N_k) \sum_{l,c \in t_k} \text{cov}(\boldsymbol{X}_{\mathcal{I}_k^C,l}, \boldsymbol{X}_{\mathcal{I}_k^C,c})$ where here $X_{\mathcal{I}_k,l}$ refers to the l^{th} column and $\mathcal{I}_k$th rows of $\boldsymbol{X}$. We make a final model selection based on the model that provides the most substantively clear treatments.

2.3 Inferring Treatments and Estimating Effects in Test Set

To discover the treatment effects, we first suppose that we have randomly assigned a set of respondents a text based treatment $\boldsymbol{X}_i$ according to some probability measure $h(\cdot)$ and that we have observed their response Y_i. We collect the assigned texts into $\boldsymbol{X}$ and the responses into $\boldsymbol{Y}$. As we describe below, we will often assign each respondent their own distinctive message, with the probability of receiving any one message at $\frac{1}{N}$ for all respondents and messages. We use the sIBP model trained our training set documents and responses to infer the effect of those treatments among the test set documents. Separating the documents and responses into training and test sets ensures that Assumption 1, SUTVA, holds. We learn the mapping from texts to binary vectors in the training set, $\hat{g}(\cdot)$ and then apply this mapping to the test set to infer the latent treatments present in the test set documents, without considering the test set responses. Dividing texts and responses into training and test sets provides a solution to SUTVA violations present in other attempts at causal inference in text analysis (Roberts et al., 2014).

We approximate the posterior distribution for the treatment vectors using the variational approximation from the training set parameters $(\widehat{\boldsymbol{\lambda}}, \widehat{\bar{\boldsymbol{\phi}}}, \widehat{\boldsymbol{\Phi}}, \widehat{\boldsymbol{m}}, \widehat{\boldsymbol{S}}, \widehat{c}, \widehat{d}, \widehat{\sigma_X^2}, \widehat{\sigma_A^2})$ and a modified update step on $q(z_{i,k}^{\text{test}})$. In this modified update step, we remove the component of the update that incorporates information about the outcome. Specifically for individual i in the test set for category k we have the following update step $v_{i,k}^{\text{test}} = \psi(\widehat{\lambda_{k,1}}) - \psi(\widehat{\lambda_{k,2}}) - \frac{1}{2(\widehat{\sigma_X^2})} \times [-2\widehat{\bar{\phi}_k}(\mathbf{X}_i^T) + (tr(\widehat{\boldsymbol{\Phi}_k}) + \widehat{\bar{\phi}_k}(\widehat{\bar{\phi}_k})^T) + 2\widehat{\bar{\phi}_k}\left(\sum_{l:l \neq k} \nu_{i,l} \left(\widehat{\bar{\phi}_l}^T\right)\right)]$
$\nu_{i,k}^{\text{test}} = \frac{1}{1+\exp(-v_{i,k}^{\text{test}})}$.
For each text in the test set we repeat this update several times until $\boldsymbol{\nu}^{\text{test}}$ has converged. Note that for the test set we have excluded the component of the model that links the latent features to the response, ensuring that SUTVA holds.

With the approximating distribution $q(\boldsymbol{Z}^{\text{test}})$ we then measure the effect of the treatments in the test set. Using the treatments, the most straightforward model to estimate assumes that there are no interactions between each of the components. Under the no interactions assumption, we estimate the effects of the treatments and infer confidence intervals using the following bootstrapping procedure that incorporates uncertainty both from estimation of treatments and uncertainty about the effects of those treatments:

1) For each respondent i and component k we draw $\tilde{z}_{i,k} \sim \text{Bernoulli}(\nu_{i,k}^{\text{test}})$, resulting in matrix $\tilde{\boldsymbol{Z}}$.

2) Given the matrix $\tilde{\boldsymbol{Z}}$, we sample $(\boldsymbol{Y}^{\text{test}}, \tilde{\boldsymbol{Z}})$ with replacement and for each sample estimate the regression $\boldsymbol{Y}^{\text{test}} = \boldsymbol{\beta}^{\text{test}} \tilde{\boldsymbol{Z}} + \boldsymbol{\epsilon}$.

We repeat the bootstrap steps 1000 times, keeping $\boldsymbol{\beta}^{\text{test}}$ for each iteration. The result of the procedure is a point estimate of the effects and confidence interval of the treatments under no interactions. Technically, it is possible to estimate the treatment effects in our variational approximation. But we estimate the effects in a second-stage regression because variational approximations tend to understate uncertainty, the bootstrap provides a straightforward method for including uncertainty from estimation of the latent features and the effect estimates, and it ensures that SUTVA is not violated.

3 Application: Voter Evaluations of an Ideal Candidate

We demonstrate our method in an experiment to assess how features of a candidate's background affect respondents evaluations of the candidates. There is a rich literature in political science about

the ideal attributes of political candidates (Canon, 1990; Popkin, 1994; Carnes, 2012; Campbell and Cowley, 2014). We build on this literature and use a collection of candidate biographies to discover features of candidates' backgrounds that voters find appealing. To uncover the features of candidate biographies that voters are responsive to we acquired a collection of 1,246 Congressional candidate biographies from Wikipedia. We then anonymize the biographies—replacing names and removing other identifiable information—to ensure that the only information available to the respondent was explicitly present in the text.

In Section 2.1 we show that a necessary condition for this experiment to uncover latent treatments is that each vector of treatments has nonzero probability of occuring. This is equivalent to assuming that none of the treatments are *aliased*, or perfectly correlated (Hainmueller et al., 2014). Aliasing would be more likely if there are only a few distinct texts that are provided to participants in our experiment. Therefore, we assign each respondent in each evaluation round a distinct candidate biography. To bolster our statistical power, we ask our respondents to evaluate up to four distinct candidate biographies, resulting in each respondent evaluating 2.8 biographies on average.[5] After presenting the respondents with a candidate's biography, we ask each respondent to rate the candidate using a *feeling thermometer*: a well-established social science scale that goes from 0 when a respodent is "cold" to a candidate to 100 when a respondent is "warm" to the candidate.

We recruited a sample of 1,886 participants using Survey Sampling International (SSI), an online survey platform. Our sample is census matched to reflect US demographics on sex, age, race, and education. Using the sample we obtain 5,303 total observations. We assign 2,651 responses to the training set and 2,652 to the test set. We then apply the sIBP process to the training data. To apply the model, we standardize the feeling thermometer to have mean zero and standard deviation 1. We set K to a relatively low value ($K = 10$) reflecting a quantitative and qualitative search over K. We then select the final model varying the parameters and evaluating the CE score.

Table 2 provides the top words for each of the ten treatments the sIBP discovered in the training set. We selected ten treatments using a combination of guidance from the sIBP, assessment using CE scores, and our own qualitative assessment of the models (Grimmer and Stewart, 2013). While it is true that our final selection depends on human input, some reliance on human judgment at this stage is appropriate. If one set includes a treatment about military service but not legal training and another set includes a treatment about legal training but not military service, then model selection is tantamount to deciding which hypotheses are most worthy of investigation. Our CE scores identify sets of treatments that are most likely to be interesting, but the human analyst should make the final decision about which hypotheses he would like to test. However, it is extremely important for the analyst to select a set of treatments first and only afterwards estimate the effects of those treatments. If the analyst observes the effects of some treatments and then decides he would like to test other sets, then the integrity of any p-values he might calculate are undermined by the multiple testing problem. A key feature of our procedure is that it draws a clear line between the selection of hypotheses to test (which leverages human judgment) and the estimation of effects (which is purely mechanical).

The estimated treatments cover salient features of Congressional biographies from the time period that we analyze. For example, treatments 6 and 10 capture a candidate's military experience. Treatment 5 and 7 are about previous political experience and Treatment 3 and 9 refer to a candidate's education experience. Clearly, there are many features of a candidate's background missing, but the treatments discovered provide a useful set of dimensions to assess how voters respond to a candidate's background. Further, the discovered treatments are a combination of those that are both prevalent in the biographies and have an effect on the thermometer rating. The absence of biographical features that we might think matters for candidate evaluation could be because there are few of those biographies in our data set, or because the respondents were unresponive to those features.

After training the model on the training set, we apply it to the test set to infer the treatments in the biographies. We assume there are no interactions

[5]The multiple evaluations of candidate biographies is problematic if there is spillover across rounds of our experiment. We have little reason to believe observing one candidate biography would systematically affect the response in subsequent rounds.

Treatment 1	Treatment 2	Treatment 3	Treatment 4	Treatment 5
appointed	fraternity	director	received	elected
school_graduated	distinguished	university	washington_university	house
governor	war_ii	received	years	democratic
worked	chapter	president	death	seat
older	air_force	master_arts	company	republican
law_firm	phi	phd	training	served
elected	reserve	policy	military	committee
grandfather	delta	public	including	appointed
office	air	master	george_washington	defeated
legal	states_air	affairs	earned_bachelors	office
Treatment 6	**Treatment 7**	**Treatment 8**	**Treatment 9**	**Treatment 10**
united_states	republican	star	law	war
military	democratic	bronze	school_law	enlisted
combat	elected	germany	law_school	united_states
rank	appointed	master_arts	juris_doctor	assigned
marine_corps	member	awarded	student	army
medal	incumbent	played	earned_juris	air
distinguished	political	yale	earned_law	states_army
air_force	father	football	law_firm	year
states_air	served	maternal	university_school	service
air	state	division	body_president	officer

Table 2: Top Words for 10 Treatments sIBP Discovered

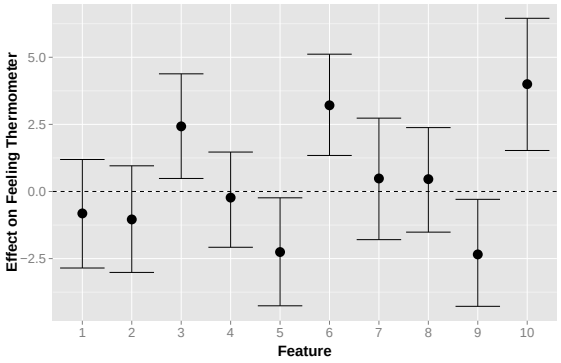


Figure 2: 95% Confidence Intervals for Effects of Discovered Treatments: The mean value of the feeling thermometer is 62.3

between the discovered treatments in order to estimate their effects.[6] Figure 2 shows the point estimate and 95-percent confidence intervals, which take into account uncertainty in inferring the treatments from the texts and the relationship between those treatments and the response.

The treatment effects reveal intuitive, though interesting, features of candidate biographies that affect respondent's evaluations. For example, Figure 2 reveals a distaste for political and legal experience—even though a large share of Congressional candidates have previous political experience and a law degree. Treatment 5, which describes a candidate's previous political experience, causes an 2.26 point reduction in feeling thermometer evaluation (95 percent confidence interval, [-4.26,-0.24]). Likewise, Treatment 9 shows that respondents dislike lawyers, with the presence of legal experience causing a 2.34 point reduction in feeling thermometer (95-percent confidence interval, [-4.28,-0.29]). The aversion to lawyers is not, however, an aversion to education. Treatment 3, a treatment that describes advanced degrees, causes a 2.43 point increase in feeling thermometer evaluations (95-percent confidence interval, [0.49,4.38]).

In contrast, Figure 2 shows that there is a consistent bonus for military experience. This is consistent with intuition from political observers that the public supports veterans. For example, treatment 6, which describes a candidate's military record, causes a 3.21 point increase in feeling thermometer rating (95-percent confidence interval, [1.34,5.12]) and treatment 10 causes a 4.00 point increase (95-percent confidence interval, [1.53,6.45]).

Because simultaneously discovering treatments from labeled data and estimating their average marginal component effects is a novel task, we cannot compare the performance of our framework against any benchmark. Even so, one natural question is whether the user could obtain much more coherent topics by foresaking the estimation of causal effects and using a more traditional topic

[6]This assumption is not necessary for the framework we propose here. Interaction effects could be modeled, but it would require us to make much stronger parametric assumptions using a method for heterogeneous treatments such as (Imai and Ratkovic, 2013).

modeling method. We provide the topics discovered by sLDA in Table 3. sIBP discovered most of the same features sLDA did. Both find military service, legal training, political background, and higher education. The Greek life feature is less coherent in sIBP than it is in sLDA, and sLDA finds business and ancestry features that sIBP does not. Both have a few incoherent treatments. This comparison suggests that sIBP does almost as well as sLDA at identifying coherent latent features, while also facilitating the estimation of marginal treatment effects.

4 Conclusion

We have presented a methodology for discovering treatments in text and then inferring the effect of those treatments on respondents' decisions. We prove that randomizing texts is sufficient to identify the underlying treatments and introduce the supervised Indian Buffet process for discovering the effects. The use of a training and test set ensures that our method provides accurate confidence intervals and avoids the problems of overfitting or "p-hacking" in experiments. In an application to candidate biographies, we discover a penalty for political and legal experience and a bonus for military service and non-legal advanced degrees.

Our methodology has a wide variety of applications. This includes numerous alternative experimental designs, providing a methodology that computational social scientists could use widely to discover and then confirm the effects of messages in numerous domains—including images and other high dimensional data. The methodology is also useful for observational data—for studying the effects of complicated treatments, such as how a legislator's roll call voting record affects their electoral support.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-114747. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation

References

John Aitchison. 1986. *The Statistical Analysis of Compositional Data*. Chapman and Hall.

Bethany Albertson and Shana Kushner Gadarian. 2015. *Anxious Politics: Democratic Citizenship in a Threatening World*. Cambridge University Press.

Stephen Ansolabehere and Shanto Iyengar. 1995. *Going Negative: How Political Advertisements Shrink and Polarize The Electorate*. Simon & Schuster, Inc.

Nick Beauchamp. 2011. A bottom-up approach to linguistic persuasion in advertising. *The Political Methodologist*.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning and Research*, 3:993–1022.

Rosie Campbell and Philip Cowley. 2014. What voters want: Reactions to candidate characteristics in a survey experiment. *Political Studies*, 62(4):745–765.

David T. Canon. 1990. *Actors, Athletes, and Astronauts: Political Amateurs in the United States Congress*. University of Chicago Press.

Nicholas Carnes. 2012. Does the numerical underrepresentation of the working class in congress matter? *Legislative Studies Quarterly*, 37(1):5–34.

Finale Doshi-Velez, Kurt T. Miller, Jurgen Van Gael, and Yee Whye Teh. 2009. Variational inference for the indian buffet process. Technical Report, University of Cambridge.

Annie Franco, Neil Malhotra, and Gabor Simonovits. 2014. Publication bias in the social sciences: Unlocking the file drawer. *Science*, 345(6203):1502–1505.

Alan S. Gerber and Donald P. Green. 2012. *Field Experiment: Design, Analysis, and Interpretation*. W.W. Norton & Company.

Zoubin Ghahramani and Thomas L Griffiths. 2005. Infinite latent feature models and the indian buffet process. In *Advances in neural information processing systems*, pages 475–482.

Justin Grimmer and Brandon M. Stewart. 2013. Text as data: The promise and pitfalls of automatic content analysis methods for political texts. *Political Analysis*, 21(3):267–297.

Jens Hainmueller, Daniel Hopkins, and Teppei Yamamoto. 2014. Causal inference in conjoint analysis: Understanding multi-dimensional choices via stated preference experiments. *Political Analysis*, 22(1):1–30.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2001. *The Elements of Statistical Learning*. Springer.

Treatment 1	Treatment 2	Treatment 3	Treatment 4	Treatment 5
years	father	school	family	united_states
work	née	medical	father	states_army
national	mother	college	white	served_united
worked	business	public_schools	parents	war
board	irish	attended	year	war_ii
young	son	county	mother	army
local	long	city	brother	served
director	family	born	married	service
social	ancestry	schools	years	lieutenant
community	descent	studied	played	military
Treatment 6	**Treatment 7**	**Treatment 8**	**Treatment 9**	**Treatment 10**
fraternity	elected	company	board	law_school
member	republican	officer	graduated	law
student	served	united_staets	college	school_law
phi	army	mexico	bachelor_arts	attorney
delta	democratic	air_force	harvard	juris_doctor
kappa	member	years	state_university	bar
hall	house_representatives	military	professor	law_firm
chi	state	national_guard	masters_degree	court
graduated	senate	insurance	high_school	law_degree
son	election	business	economics	judge

Table 3: Top Words for 10 Treatments sLDA Discovered

Paul Holland. 1986. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396):945–960.

Macartan Humphreys, Raul Sanchez de la Sierra, and Peter van der Windt. 2013. Fishing, commitment, and communication: A proposal for comprehensive nonbinding research registration. *Political Analysis*, 21(1):1–20.

Kosuke Imai and Marc Ratkovic. 2013. Estimating treatment effect heterogeneity in randomized program evaluation. *The Annals of Applied Statistics*, 7(1):443–470.

John P. A. Ioannidis. 2005. Why most published research findings are false. *PLoS Medicine*, 2(8):696–701.

Jonathan Katz and Gary King. 1999. A statistical model for multiparty electoral data. *The American Political Science Review*, 93(1):15–32.

Jon D. Mcauliffe and David M. Blei. 2007. Supervised topic models. In *Advances in Neural Information Processing Systems 20 (NIPS 2007)*.

David Mimno, Hanna M Wallach, Edmund Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 262–272.

Diana C Mutz. 2011. *Population-Based Survey Experiments*. Princeton University Press.

Samuel L Popkin. 1994. *The Reasoning Voter: Communication and Persuasion in Presidential Campaigns*. University of Chicago Press.

Novi Quadrianto, Viktoriia Sharmanska, David A. Knowles, and Zoubin Ghahramani. 2013. The supervised ibp: Neighbourhood preserving infinite latent feature models. page 101.

Margaret E Roberts, Brandon M Stewart, Dustin Tingley, Chris Lucas, Jetson Leder-Luis, Bethany Albertson, Shana Gadarian, and David Rand. 2014. Topic models for open ended survey responses with applications to experiments. *American Journal of Political Science*.

Margaret E. Roberts, Brandon M. Stewart, and Edo M. Airoldi. 2016. A model of text for experimentation in the social sciences. *Journal of the American Statistical Association*. Forthcoming.

Don Rubin. 1974. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66:688–701.

Donald B. Rubin. 1986. Statistics and causal inference: Comment: Which ifs have causal answers. *Journal of the American Statistical Association*, 81(396):961–962.

Michael Tomz and Jessica Weeks. 2013. Public opinion and the democratic peace. *American Political Science Review*, 107(4):849–865.

Learning Structured Predictors from Bandit Feedback for Interactive NLP

Artem Sokolov[◇,*] and **Julia Kreutzer**[*] and **Christopher Lo**[†,*] and **Stefan Riezler**[‡,*]
[*]Computational Linguistics & [‡]IWR, Heidelberg University, Germany
{sokolov,kreutzer,riezler}@cl.uni-heidelberg.de
[†]Department of Mathematics, Tufts University, Boston, MA, USA
chris.aa.lo@gmail.com
[◇]Amazon Development Center, Berlin, Germany

Abstract

Structured prediction from bandit feedback describes a learning scenario where instead of having access to a gold standard structure, a learner only receives partial feedback in form of the loss value of a predicted structure. We present new learning objectives and algorithms for this interactive scenario, focusing on convergence speed and ease of elicitability of feedback. We present supervised-to-bandit simulation experiments for several NLP tasks (machine translation, sequence labeling, text classification), showing that bandit learning from relative preferences eases feedback strength and yields improved empirical convergence.

1 Introduction

Structured prediction from partial information can be described by the following learning protocol: On each of a sequence of rounds, the learning algorithm makes a prediction, and receives partial information in terms of feedback on the predicted point. This single-point feedback is used to construct a parameter update that is an unbiased estimate of the respective update rule for the full information objective. In difference to the full information scenario, the learner does not know what the correct prediction looks like, nor what would have happened if it had predicted differently. This learning scenario has been investigated under the names of *learning from bandit feedback*[1] or *reinforcement learning*[2], and has (financially) important real world applications such as online advertising (Chapelle et al., 2014). In this application, the probability that an ad will be clicked (and the advertiser has to pay) is estimated by trading off exploration (a new ad needs to be displayed in order to learn its click-through rate) and exploitation (displaying the ad with the current best estimate is better in the short term) in displaying ads to users. Similar to the online advertising scenario, there are many potential applications to interactive learning in NLP. For example, in interactive statistical machine translation (SMT), user feedback in form of post-edits of predicted translations is used for model adaptation (Bertoldi et al., 2014; Denkowski et al., 2014; Green et al., 2014). Since post-editing feedback has a high cost and requires professional expertise of users, weaker forms of feedback are desirable. Sokolov et al. (2015) showed in a simulation experiment that partial information in form of translation quality judgements on predicted translations is sufficient for model adaptation in SMT. However, one drawback of their *bandit expected loss minimization* algorithm is the slow convergence speed, meaning that impractically many rounds of user feedback would be necessary for learning in real-world interactive SMT. Furthermore, their algorithms requires feedback in form of numerical assessments of translation quality. Such absolute feedback is arguably harder to elicit from human users than relative judgements.

The goal of this work is a preparatory study of different objectives and algorithms for structured prediction from partial information with real-world interactive scenarios in mind. Since the algorithm of Sokolov et al. (2015) can be characterized as stochastic optimization of a non-convex

[*]The work for this paper was done while the authors were at Heidelberg University.

[1]The name is inherited from a model where in each round a gambler pulls an arm of a different slot machine ("one-armed bandit"), with the goal of maximizing his reward relative to the maximal possible reward, without apriori knowledge of the optimal slot machine. See Bubeck and Cesa-Bianchi (2012) for an overview.

[2]See Szepesvári (2009) for an overview of algorithms for reinforcement learning and their relation to bandit learning.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1610–1620,
Berlin, Germany, August 7-12, 2016.

objective, a possible avenue to address the problem of convergence speed is a (strong) convexification of the learning objective, which we formalize as *bandit cross-entropy minimization*. To the aim of easing elicitability of feedback, we present a *bandit pairwise preference learning* algorithm that requires only relative feedback in the form of pairwise preference rankings.

The focus of this paper is on an experimental evaluation of the empirical performance and convergence speed of the different algorithms. We follow the standard practice of early stopping by measuring performance on a development set, and present results of an extensive evaluation on several tasks with different loss functions, including BLEU for SMT, Hamming loss for optical character recognition, and F1 score for chunking. In our experiments, we use a standard supervised-to-bandit transformation where a reward signal is simulated by evaluating a task loss against gold standard structures without revealing them to the learning algorithm (Agarwal et al., 2014). From the perspective of real-world interactive applications, bandit pairwise preference learning is the preferred algorithm since it only requires comparative judgements for learning. This type of relative feedback been shown to be advantageous for human decision making (Thurstone, 1927). However, in our simulation experiments we found that relative feedback also results in improved empirical convergence speed for bandit pairwise preference learning. The picture of fastest empirical convergence of bandit pairwise preference learning is consistent across different tasks, both compared to bandit expected loss minimization and bandit cross-entropy minimization. Given the improved convergence and the ease of elicitability of relative feedback, the presented bandit pairwise preference learner is an attractive choice for interactive NLP tasks.

2 Related Work

Reinforcement learning (RL) has the goal of maximizing the expected reward for choosing an action at a given state in a Markov Decision Process (MDP) model, where rewards are received at each state or once at the final state. The algorithms in this paper can be seen as one-state MDPs where choosing an action corresponds to predicting a structured output. Most closely related are RL approaches that use gradient-based optimization of a parametric policy for action selection (Bertsekas and Tsitsiklis, 1996; Sutton et al., 2000). Policy gradient approaches have been applied to NLP tasks by Branavan et al. (2009), Chang et al. (2015) or Ranzato et al. (2016).

Bandit learning operates in a similar scenario of maximizing the expected reward for selecting an arm of a multi-armed slot machine. Similar to our case, the models consist of a single state, however, arms are usually selected from a small set of options while structures are predicted over exponential output spaces. While bandit learning is mostly formalized as online regret minimization with respect to the best fixed arm in hindsight, we investigate asymptotic convergence of our algorithms. In the spectrum of stochastic (Auer et al., 2002a) versus adversarial bandits (Auer et al., 2002b), our approach takes a middle path by making stochastic assumptions on inputs, but not on rewards. Most closely related are algorithms that optimize parametric models, e.g., contextual bandits (Langford and Zhang, 2007; Li et al., 2010) or combinatorial bandits (Dani et al., 2007; Cesa-Bianchi and Lugosi, 2012). To the best of our knowledge, these types of algorithms have not yet been applied in the area of NLP.

Pairwise preference learning has been studied in the full information supervised setting (see Herbrich et al. (2000), Joachims (2002), Freund et al. (2003), Cortes et al. (2007), Fürnkranz and Hüllermeier (2010), *inter alia*) where given preference pairs are assumed. Stochastic optimization from two-point (or multi-point) feedback has been investigated in the framework of gradient-free optimization (see Yue and Joachims (2009), Agarwal et al. (2010), Ghadimi and Lan (2012), Jamieson et al. (2012), Duchi et al. (2015), *inter alia*), while our algorithms can be characterized as stochastic gradient descent algorithms.

3 Probabilistic Structured Prediction

3.1 Full Information vs. Bandit Feedback

The objectives and algorithms presented in this paper are based on the well-known expected loss criterion for probabilistic structured prediction (see Och (2003), Smith and Eisner (2006), Gimpel and Smith (2010), Yuille and He (2012), He and Deng (2012), *inter alia*). The objective is defined as a minimization of the expectation of a given task loss function with respect to the conditional distribution over structured outputs. This criterion

has the form of a continuous, differentiable, and in general, non-convex objective function. More formally, let $\mathcal{X}$ be a structured input space, let $\mathcal{Y}(x)$ be the set of possible output structures for input x, and let $\Delta_y : \mathcal{Y} \rightarrow [0, 1]$ quantify the loss $\Delta_y(y')$ suffered for predicting y' instead of the gold standard structure y; as a rule, $\Delta_y(y') = 0$ iff $y = y'$. In the full information setting, for a data distribution $p(x, y)$, the learning criterion is defined as minimization of the expected loss with respect to $w \in \mathbb{R}^d$ where

$$\mathbb{E}_{p(x,y)p_w(y'|x)}\left[\Delta_y(y')\right] = \sum_{x,y} p(x,y) \sum_{y' \in \mathcal{Y}(x)} \Delta_y(y') p_w(y'|x). \quad (1)$$

Assume further that output structures given inputs are distributed according to an underlying Gibbs distribution (a.k.a. conditional exponential or log-linear model)

$$p_w(y|x) = \exp(w^\top \phi(x, y))/Z_w(x),$$

where $\phi : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$ is a joint feature representation of inputs and outputs, $w \in \mathbb{R}^d$ is an associated weight vector, and $Z_w(x)$ is a normalization constant. For this model, the gradient of objective (1) is as follows:

$$\nabla \mathbb{E}_{p(x,y)p_w(y'|x)}\left[\Delta_y(y')\right] = \mathbb{E}_{p(x,y)p_w(y'|x)}\Big[\Delta_y(y')\left(\phi(x, y') - \mathbb{E}_{p_w(y'|x)}[\phi(x, y')]\right)\Big]. \quad (2)$$

Unlike in the full information scenario, bandit feedback in structured prediction means that the gold standard output structure y, with respect to which the objective function is evaluated, is not revealed to the learner. Thus we can neither evaluate the task loss Δ nor calculate the gradient (2) of the objective function (1). A solution to this problem is to pass the evaluation of the loss function to the user, i.e, we access the loss directly through user feedback without assuming existence of a fixed reference y. We indicate this by dropping the subscript referring to the gold standard structure in the definition of Δ. In all algorithms presented below we need to make the following assumptions:

1. We assume a sequence of input structures $x_t, t = 1, \ldots, T$ that are generated by a fixed, unknown distribution $p(x)$.

Algorithm 1 Bandit Expected Loss Minimization

Input: sequence of learning rates γ_t
Initialize w_0
for $t = 0, \ldots, T$ **do**
  Observe x_t
  Calculate $\mathbb{E}_{p_{w_t}(y|x_t)}[\phi(x_t, y)]$
  Sample $\tilde{y}_t \sim p_{w_t}(y|x_t)$
  Obtain feedback $\Delta(\tilde{y}_t)$
  $w_{t+1} = w_t - \gamma_t \, \Delta(\tilde{y}_t)$
    $\times\big(\phi(x_t, \tilde{y}_t) - \mathbb{E}_{p_{w_t}}[\phi(x_t, y)]\big)$

Algorithm 2 Bandit Pairwise Preference Learning

Input: sequence of learning rates γ_t
Initialize w_0
for $t = 0, \ldots, T$ **do**
  Observe x_t
  Calculate $\mathbb{E}_{p_{w_t}(\langle y_i, y_j\rangle|x_t)}[\phi(x_t, \langle y_i, y_j\rangle)]$
  Sample $\langle \tilde{y}_i, \tilde{y}_j \rangle_t \sim p_{w_t}(\langle y_i, y_j\rangle \,|x_t)$
  Obtain feedback $\Delta(\langle \tilde{y}_i, \tilde{y}_j \rangle_t)$
  $w_{t+1} = w_t - \gamma_t \, \Delta(\langle \tilde{y}_i, \tilde{y}_j \rangle_t)$
    $\times\big(\phi(x_t, \langle \tilde{y}_i, \tilde{y}_j \rangle_t) - \mathbb{E}_{p_{w_t}}[\phi(x_t, \langle y_i, y_j\rangle)]\big)$

2. We use a Gibbs model as sampling distribution to perform simultaneous exploitation (use the current best estimate) / exploration (get new information) on output structures.

3. We use feedback to the sampled output structures to construct a parameter update rule that is an unbiased estimate of the true gradient of the respective objective.

3.2 Learning Objectives and Algorithms

Bandit Expected Loss Minimization. Algorithm 1 has been presented in Sokolov et al. (2015) and minimizes the objective below by stochastic gradient descent optimization. It is non-convex for the specific instantiations in this paper:

$$\mathbb{E}_{p(x)p_w(y|x)}\left[\Delta(y)\right] = \sum_{x} p(x) \sum_{y \in \mathcal{Y}(x)} \Delta(y) p_w(y|x). \quad (3)$$

Intuitively, the algorithm compares the sampled feature vector to the average feature vector, and performs a step into the opposite direction of this difference, the more so the higher the loss of the sampled structure is. In the extreme case, if the sampled structure is correct ($\Delta(\tilde{y}_t) = 0$), no update is performed.

Algorithm 3 Bandit Cross-Entropy Minimization

Input: sequence of learning rates γ_t
Initialize w_0
for $t = 0, \ldots, T$ **do**
  Observe x_t
  Sample $\tilde{y}_t \sim p_{w_t}(y|x_t)$
  Obtain feedback $g(\tilde{y}_t)$
  $w_{t+1} = w_t - \gamma_t \frac{g(\tilde{y}_t)}{p_{w_t}(\tilde{y}_t|x_t)}$
    $\times\big(- \phi(x_t, \tilde{y}_t) + \mathbb{E}_{p_{w_t}}[\phi(x_t, \tilde{y}_t)]\big)$

Bandit Pairwise Preference Learning. Decomposing complex problems into a series of pairwise comparisons has been shown to be advantageous for human decision making (Thurstone, 1927) and for machine learning (Fürnkranz and Hüllermeier, 2010). For our case, this idea can be formalized as an expected loss objective with respect to a conditional distribution of pairs of structured outputs. Let $\mathcal{P}(x) = \{\langle y_i, y_j\rangle \,|y_i, y_j \in \mathcal{Y}(x)\}$ denote the set of output pairs for an input x, and let $\Delta(\langle y_i, y_j\rangle) : \mathcal{P}(x) \to [0, 1]$ denote a task loss function that specifies a dispreference of y_i compared to y_j. Instantiating objective (3) to the case of pairs of output structures defines the following objective:

$$\mathbb{E}_{p(x)p_w(\langle y_i,y_j\rangle|x)}\left[\Delta(\langle y_i, y_j\rangle)\right]. \tag{4}$$

Stochastic gradient descent optimization of this objective leads to Algorithm 2. The objective is again non-convex in the use cases in this paper. Minimization of this objective will assure that high probabilities are assigned to pairs with low loss due to misranking y_j over y_i. Stronger assumptions on the learned probability ranking can be made if assumptions of transitivity and asymmetry of the ordering of feedback structures are made. For efficient sampling and calculation of expectations, we assume a Gibbs model that factorizes as follows:

$$\begin{aligned} p_w(\langle y_i, y_j\rangle\,|x) &= \frac{e^{w^\top(\phi(x,y_i)-\phi(x,y_j))}}{\sum\limits_{\langle y_i,y_j\rangle\in\mathcal{P}(x)} e^{w^\top(\phi(x,y_i)-\phi(x,y_j))}} \\ &= p_w(y_i|x)p_{-w}(y_j|x). \end{aligned}$$

If a sample from the p_{-w} distribution is preferred over a sample from the p_w distribution, this is a strong signal for model correction.

Bandit Cross-Entropy Minimization. The standard theory of stochastic optimization predicts considerable improvements in convergence speed depending on the functional form of the objective. This motivates the formalization of convex upper bounds on expected normalized loss as presented in Green et al. (2014). Their objective is based on a gain function $g : \mathcal{Y} \to [0, 1]$ (in this work, $g(y) = 1 - \Delta(y)$) that is normalized over n-best lists where $\bar{g}(y) = \frac{g(y)}{Z_g(x)}$ and $Z_g(x) = \sum_{y\in n\text{-best}(x)} g(y)$. It can be seen as the cross-entropy of model $p_w(y|x)$ with respect the "true" distribution $\bar{g}(y)$:

$$\begin{aligned} &\mathbb{E}_{p(x)\bar{g}(y)}\left[-\log p_w(y|x)\right] \\ &\quad = -\sum_x p(x) \sum_{y\in\mathcal{Y}(x)} \bar{g}(y) \log p_w(y|x). \end{aligned} \tag{5}$$

For a proper probability distribution $\bar{g}(y)$, an application of Jensen's inequality to the convex negative logarithm function shows that objective (5) is a convex upper bound on objective (3). However, normalizing the gain function is prohibitive in a bandit setting since it would require to elicit user feedback for each structure in the output space or n-best list. We thus work with an unnormalized gain function which sacrifices the upper bound but preserves convexity. This can be seen by rewriting the objective as the sum of a linear and a convex function in w:

$$\begin{aligned} &\mathbb{E}_{p(x)g(y)}\left[-\log p_w(y|x)\right] \\ &\quad = -\sum_x p(x) \sum_{y\in\mathcal{Y}(x)} g(y) w^\top \phi(x, y) \\ &\qquad + \sum_x p(x)(\log \sum_{y\in\mathcal{Y}(x)} \exp(w^\top\phi(x, y)))\alpha(x), \end{aligned} \tag{6}$$

where $\alpha(x) = \sum_{y\in\mathcal{Y}(x)} g(y)$ is a constant factor not depending on w. The gradient of objective (6) is as follows:

$$\begin{aligned} &\nabla(-\sum_x p(x) \sum_{y\in\mathcal{Y}(x)} g(y) \log p_w(y|x)) \\ &\quad = \mathbb{E}_{p(x)p_s(y|x)}\big[\frac{g(y)}{p_s(y|x)}\big(- \phi(x, y) \\ &\qquad\qquad + \mathbb{E}_{p_w(y|x)}[\phi(x, y)]\big)\big]. \end{aligned}$$

Minimization of this objective will assign high probabilities to structures with high gain, as desired. Algorithm 3 minimizes this objective by sampling from a distribution $p_s(y|x)$, receiving feedback, and updating according to the ratio of gain versus current probability of the sampled structure. A positive ratio expresses a preference

of the sampled structure under the gain function compared to the current probability estimate. We compare the sampled feature vector to the average feature vector, and we update towards the sampled feature vector relative to this ratio. We instantiate $p_s(y|x)$ to the current update of $p_{w_t}(y|x)$ in order to present progressively more useful structures to the user. In contrast to Algorithms 1 and 2, each update is thus affected by a probability that changes over time and is unreliable when training is started. This further increases the variance already present in stochastic optimization. We deal with this problem by clipping too small sampling probabilities (Ionides, 2008) or by reducing variance using momentum techniques (Polyak, 1964).

3.3 Remarks on Theoretical Analysis

Convergence of our algorithms can be analyzed using results of standard stochastic approximation theory. For example, Sokolov et al. (2015) analyze the convergence of Algorithm 1 in the pseudogradient framework of Polyak and Tsypkin (1973), relying on the fact that a positive inner product of the update vector with the gradient in expectation suffices for convergence. Sokolov et al. (2016) analyze convergence in the framework of stochastic first-order optimization of Ghadimi and Lan (2012), relying on the fact that the update vectors of the algorithms are stochastic gradients of the respective objectives, that is, the update vectors are unbiased gradient measurements that equal the gradient of the full information objective in expectation. Note that the latter analysis covers the use of constant learning rates.

Convergence speed is analyzed in standard stochastic approximation theory in terms of the number of iterations needed to reach an accuracy of ϵ for a gradient-based criterion

$$\mathbb{E}[\|\nabla J(w_t)\|^2] \leq \epsilon, \tag{7}$$

where $J(w_t)$ denotes the objective to be minimized. Following Ghadimi and Lan (2012), the iteration complexity of the non-convex objectives underlying our Algorithms 1 and 2 can be given as $\mathcal{O}(1/\epsilon^2)$ (see Sokolov et al. (2016)). Algorithm 3 can be seen as stochastic optimization of a strongly convex objective that is attained by adding an ℓ_2 regularizer $\frac{\lambda}{2}\|w\|^2$ with constant $\lambda > 0$ to objective (6). In the standard stochastic approximation theory, the iteration complexity of stochastic gradient algorithms using decreasing learning rates can be given as $\mathcal{O}(1/\epsilon)$ for an objective value-based criterion

$$\mathbb{E}[J(w_t)] - J(w^*) \leq \epsilon,$$

where $w^* = \arg\min_w J(w)$ (Polyak, 1987). For constant learning rates, even faster convergence can be shown provided certain additional conditions are met (Solodov, 1998).

While the asymptotic iteration complexity bounds predict faster convergence for Algorithm 3 compared to Algorithms 1 and 2, and equal convergence speed for the latter two, Sokolov et al. (2016) show that the hidden constant of variance of the stochastic gradient can offset this advantage empirically. They find smallest variance of stochastic updates and fastest empirical convergence under the gradient-based criterion (7) for Algorithm 2. In the next section we will present experimental results that show similar relations of fastest convergence of Algorithm 2 under a convergence criterion based on task loss evaluation on heldout data.

4 Experiments

Experimental design. Our experiments follow an online learning protocol where on each of a sequence of rounds, an output structure is randomly sampled, and feedback to it is used to update the model (Shalev-Shwartz, 2012). We simulate bandit feedback by evaluating Δ against gold standard structures which are never revealed to the learner (Agarwal et al., 2014). Training is started from $w_0 = \mathbf{0}$ or from an out-of-domain model (for SMT).

Following the standard practice of early stopping by performance evaluation on a development set, we compute convergence speed as the number of iterations needed to find the point of optimal performance before overfitting on the development set occurs. The convergence criterion is thus based on the respective task loss function $\Delta(\hat{y}_{w_t}(x))$ under MAP prediction $\hat{y}_w(x) = \arg\max_{y \in \mathcal{Y}(x)} p_w(y|x)$, microaveraged on the development data. This lets us compare convergence across different objectives, and is justified by the standard practice of performing online-to-batch conversion by early stopping on a development set (Littlestone, 1989), or by tolerant training to avoid overfitting (Solodov, 1998). As a further measure for comparability of convergence

	task	Algorithm 1	Algorithm 2	Algorithm 3
	Text classification	$\gamma_t = 1.0$	$\gamma_t = 10^{-0.75}$	$\gamma_t = 10^{-1}$
CRF	OCR	$T_0 = 0.4, \gamma_t = 10^{-3.5}$	$T_0 = 0.1, \gamma_t = 10^{-4}$	$\lambda = 10^{-5}, k = 10^{-2}, \gamma_t = 10^{-6}$
CRF	Chunking	$\gamma_t = 10^{-4}$	$\gamma_t = 10^{-4}$	$\lambda = 10^{-6}, k = 10^{-2}, \gamma_t = 10^{-6}$
SMT	News (n-best, dense)	$\gamma_t = 10^{-5}$	$\gamma_t = 10^{-4.75}$	$\lambda = 10^{-4}, \mu = 0.99, \gamma_t = 10^{-6}/\sqrt{t}$
SMT	News (h-graph, sparse)	$\gamma_t = 10^{-5}$	$\gamma_t = 10^{-4}$	$\lambda = 10^{-6}, k = 5 \cdot 10^{-3}, \gamma_t = 10^{-6}$

Table 1: Metaparameter settings determined on *dev* sets for constant learning rate γ_t, temperature coefficient T_0 for annealing under the schedule $T = T_0/\sqrt[3]{\text{epoch} + 1}$ (Rose, 1998; Arun et al., 2010), momentum coefficient $\min\{1 - 1/(t/2 + 2), \mu\}$ (Polyak, 1964; Sutskever et al., 2013), clipping constant k used to replace $p_{w_t}(\tilde{y}_t|x_t)$ with $\max\{p_{w_t}(\tilde{y}_t|x_t), k\}$ in line 7 of Algorithm 3 (Ionides, 2008), ℓ_2 regularization constant λ. Unspecified parameters are set to zero.

speeds across algorithms, we employ small constant learning rates in all experiments. The use of constant learning rates for Algorithms 1 and 2 is justified by the analysis of Ghadimi and Lan (2012). For Algorithm 3, the use of constant learning rates effectively compares convergence speed towards an area in close vicinity of a local minimum in the search phase of the algorithm (Bottou, 2004).

The development data are also used for metaparameter search. Optimal configurations are listed in Table 1. Final testing was done by computing Δ on a further unseen test set using the model found by online-to-batch conversion. For bandit-type algorithms, final results are averaged over 3 runs with different random seeds. For statistical significance testing of results against baselines we use Approximate Randomization testing (Noreen, 1989).

Multiclass classification. Multiclass text classification on the Reuters RCV1 dataset (Lewis et al., 2004) is a standard benchmark for (simplified) structured prediction that has been used in a bandit setup by Kakade et al. (2008). The simplified problem uses a binary Δ function indicating incorrect assignment of one out of 4 classes. Following Kakade et al. (2008), we used documents with exactly one label from the set of labels {CCAT, ECAT, GCAT, MCAT} and converted them to *tfidf* word vectors of dimension 244,805 in training. The data were split into the sets *train* (509,381 documents from original `test_pt[0-2].dat` files), *dev* (19,486 docs: every 8th entry from `test_pt3.dat` and *test* (19,806 docs from `train.dat`).

As shown in Table 2 (row 1), all loss results are small and comparable since the task is relatively easy. For comparison, the partial information classification algorithm Banditron (Kakade et al., 2008) (after adjusting the exploration/exploitation constant on the dev set) scored 0.047 on the test set. However, our main interest is in convergence speed. Table 3 (row 1) shows that pairwise ranking (Algorithm 2) yields fastest convergence by a factor of 2-4 compared to the other bandit algorithms. Table 1 confirms that this improvement is not attributable to larger learning rates (Algorithm 2 employs a similar or smaller learning rate than Algorithms 1 and 3, respectively.)

Sequence labeling for OCR and chunking. Handwritten optical character recognition (OCR) is a standard benchmark task for structured prediction (Taskar et al., 2003), where the Hamming distance between the predicted word and the gold standard labeling (normalized by word length) is assumed as the Δ function. We used their dataset of 6,876 handwritten words, from 150 human subjects, under a split where 5,546 examples (folds 2-9) were used as *train* set, 704 examples (fold 1) as *dev*, and 626 (fold 0) as *test* set. We assumed the classical linear-chain Conditional Random Field (CRF) (Lafferty et al., 2001) model with input images x^i at every ith node, tabular state-transition probabilities between 28 possible labels of the $(i-1)$th and ith node (Latin letters plus two auxiliary `start` and `stop` states).[3]

To test the CRF-based model also with sparse features, we followed Sha and Pereira (2003) in applying CRFs to the noun phrase chunking task

[3]The feature set is composed of a 16×8 binary pixel representation for each character, yielding $28 \times 16 \times 8 + 28^2 = 4,368$ features for the training set. We based our code on the `pystruct` kit (Müller and Behnke, 2014).

	task	gain/loss	full information		partial information		
					Alg. 1	Alg. 2	Alg. 3
	Text classification	0/1 ↓	percep., $\lambda = 10^{-6}$	0.040	$0.0306_{\pm 0.0004}$	$0.083_{\pm 0.002}$	$0.035_{\pm 0.001}$
CRF	OCR (dense)	Hamming ↓	likelihood	0.099	$0.261_{\pm 0.003}$	$0.332_{\pm 0.011}$	$0.257_{\pm 0.004}$
CRF	Chunking (sparse)	F1-score ↑	likelihood	0.935	$0.923_{\pm 0.002}$	$0.914_{\pm 0.002}$	$0.891_{\pm 0.005}$
			out-of-domain	**in-domain**	**Alg. 1**	**Alg. 2**	**Alg. 3**
SMT	News (n-best list, dense)	BLEU ↑	0.2588	0.2841	$0.2689_{\pm 0.0003}$	$0.2745_{\pm 0.0004}$	$0.2763_{\pm 0.0005}$
SMT	News (hypergraph, sparse)		0.2651	0.2831	$0.2667_{\pm 0.00008}$	$0.2733_{\pm 0.0005}$	$0.2713_{\pm 0.001}$

Table 2: Test set evaluation for full information lower and upper bounds and partial information bandit learners (expected loss, pairwise loss, cross-entropy). ↑ and ↓ indicate the direction of improvement for the respective evaluation metric.

on the CoNLL-2000 dataset[4]. We split the original training set into a *dev* set (top 1,000 sent.) and used the rest as *train* set (7,936 sent.); the *test* set was kept intact (2,012 sent.). For an input sentence x, each CRF node x^i carries an observable word and its part-of-speech tag, and has to be assigned a chunk tag c^i out of 3 labels: **B**eginning, **I**nside, or **O**utside (of a noun phrase). Chunk labels are not nested. As in Sha and Pereira (2003), we use second order Markov dependencies (bigram chunk tags), such that for sentence position i, the state is $y^i = c^{i-1}c^i$, increasing the label set size from 3 to 9. Out of the full list of Sha and Pereira (2003)'s features we implemented all except two feature templates, $y^i = y$ and $c(y^i) = c$, to simplify implementation. Impossible bigrams (OI) and label transitions of the pattern $\star O \rightarrow I\star$ were prohibited by setting the respective potentials to $-\infty$. As the active feature count in the train set was just under 2M, we hashed all features and weights into a sparse array of 2M entries. Despite the reduced train size and feature set, and hashing, our full information baseline trained with log-likelihood attained the test F1-score of 0.935, which is comparable to the original result of 0.9438.

Table 2 (rows 2-3) and Table 3 (rows 2-3) show evaluation and convergence results for the OCR and chunking tasks. For the chunking task, the F1-score results obtained for bandit learning are close to the full-information baseline. For the OCR task, bandit learning does decrease Hamming loss, but it does not quite achieve full-information performance. However, pairwise ranking (Algorithm 2) again converges faster than the alternative bandit algorithms by a factor of 2-4, despite similar learning rates for Algorithms 1 and 2 and a compensation of smaller learning rates in Algorithm 3 by variance reduction and regularization.

[4] `http://www.cnts.ua.ac.be/conll2000/chunking/`

	task	Alg. 1	Alg. 2	Alg. 3
	Text classification	2.0M	0.5M	1.1M
CRF	OCR	14.4M	9.3M	37.9M
CRF	Chunking	7.5M	4.7M	5.9M
SMT	News (n-best, dense)	3.8M	1.2M	1.2M
SMT	News (h-graph, sparse)	370k	115k	281k

Table 3: Number of iterations required to meet stopping criterion on development data.

Discriminative ranking for SMT. Following Sokolov et al. (2015), we apply bandit learning to simulate personalized MT where a given SMT system is adapted to user style and domain based on feedback to predicted translations. We perform French-to-English domain adaptation from Europarl to NewsCommentary domains using the data of Koehn and Schroeder (2007). One difference of our experiment compared to Sokolov et al. (2015) is our use of the SCFG decoder `cdec` (Dyer et al., 2010) (instead of the phrase-based `Moses` decoder). Furthermore, in addition to bandit learning for re-ranking on unique 5,000-best lists, we perform ranking on hypergraphs with re-decoding after each update. Sampling and computation of expectations on the hypergraph uses the Inside-Outside algorithm over the expectation semiring (Li and Eisner, 2009). The re-ranking model used 15 dense features (6 lexicalized reordering features, two (out-of- and in-domain) language models, 5 translation model features, distortion and word penalty). The hypergraph experiments used additionally lexicalized sparse features: rule-id features, rule source and target bigram features, and rule shape features.

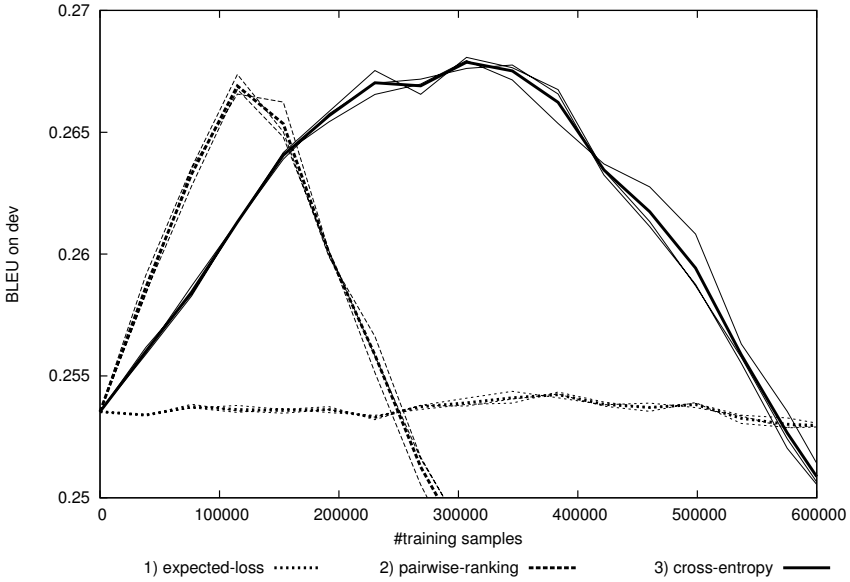


Figure 1: Learning curves for task loss BLEU on development data for SMT hypergraph re-decoding models, together with averages over three runs of the respective algorithms.

For all SMT experiments we tokenized, lowercased and aligned words using `cdec` tools, trained 4-gram in-domain and out-of-domain language models (on the English sides of Europarl and in-domain NewsCommentary) For dense feature models, the *out-of-domain* baseline SMT model was trained on 1.6M parallel Europarl data and tuned with `cdec`'s lattice MERT (Och, 2003) on out-of-domain Europarl `dev2006` dev set (2,000 sent.). The full-information *in-domain* SMT model tuned by MERT on news in-domain sets (`nc-dev2007`, 1,057 sent.) gives the range of possible improvements by the difference of its BLEU score to the one of the out-of-domain model (2.5 BLEU points). For sparse feature models, in-domain and out-of-domain baselines were trained on the same data using MIRA (Chiang, 2012). The in-domain MIRA model contains 133,531 active features, the out-of-domain MIRA model 214,642. MERT and MIRA runs for both settings were repeated 7 times and median results are reported.

Learning under *bandit feedback* starts at the learned weights of the out-of-domain median models. It uses the parallel in-domain data (`news-commentary`, 40,444 sent.) to simulate bandit feedback, by evaluating the sampled translation against the reference using as loss function Δ a smoothed per-sentence $1 - \text{BLEU}$ (zero n-gram counts being replaced with 0.01). For pairwise preference learning we use binary feedback resulting from the comparison of the BLEU scores of the sampled translations. To speed up training for hypergraph re-decoding, the training instances were reduced to those with at most 60 words (38,350 sent.). Training is distributed across 38 shards using multitask-based feature selection for sparse models (Simianer et al., 2012), where after each epoch of distributed training, the top 10k features across all shards are selected, all other features are set to zero. The meta-parameters were adjusted on the in-domain dev sets (`nc-devtest2007`, 1,064 parallel sentences). The final results are obtained on separate in-domain test sets (`nc-test2007`, 2,007 sentences) by averaging three independent runs for the optimal dev set meta-parameters.

The results for n-best re-ranking in Table 2 (4th row) show statistically significant improvements of 1-2 BLEU points over the out-of-domain SMT model (that includes an in-domain language model) for all bandit learning methods, confirming the results of Sokolov et al. (2015) for a different decoder. Similarly, the results for hypergraph re-coding with sparse feature models (row 5 in Table 2) show significant improvements over the out-of-domain baseline for all bandit learners. Table 3 (row 4) shows the convergence speed for n-best re-ranking, which is similar for Algorithms 2 and 3, and improved over Algorithm 1 by a factor of 3. For hypergraph re-decoding, Table 3 (row 5) shows fastest convergence for Algorithm 2 com-

pared to Algorithms 1 and 3 by a factor of 2-4.[5] Again, we note that for both n-best re-ranking and hypergraph re-decoding, learning rates are similar for Algorithms 1 and 2, and smaller learning rates in Algorithm 3 are compensated by variance reduction or regularization.

Figure 1 shows the learning curves of BLEU for SMT hypergraph re-decoding on the development set that were used to find the stopping points. For each algorithm, we show learning curves for three runs with different random seeds, together with an average learning curve. We see that Algorithm 2, optimizing the pairwise preference ranking objective, reaches the stopping point of peak performance on development data fastest, followed by Algorithms 1 and 3. Furthermore, the larger variance of the runs of Algorithm 3 is visible, despite the smallest learning rate used.

5 Conclusion

We presented objectives and algorithms for structured prediction from bandit feedback, with a focus on improving convergence speed and ease of elicitability of feedback. We investigated the performance of all algorithms by test set performance on different tasks, however, the main interest of this paper was a comparison of convergence speed across different objectives by early stopping on a convergence criterion based on heldout data performance. Our experimental results on different NLP tasks showed a consistent advantage of convergence speed under this criterion for bandit pairwise preference learning. In light of the standard stochastic approximation analysis, which predicts a convergence advantage for strongly convex objectives over convex or non-convex objectives, this result is surprising. However, the result can be explained by considering important empirical factors such as the variance of stochastic updates. Our experimental results support the numerical results of smallest stochastic variance and fastest convergence in gradient norm (Sokolov et al., 2016) by consistent fastest empirical convergence for bandit pairwise preference learning under the criterion of early stopping on heldout data performance. Given the advantages of faster convergence and the fact that only relative feedback in terms of comparative evaluations is required, bandit pairwise preference learning is a promising framework for future real-world interactive learning.

[5]The faster convergence speed hypergraph re-decoding compared to n-best re-ranking is due to the distributed feature selection and thus orthogonal to the comparison of objective functions that is of interest here.

Acknowledgments

This research was supported in part by the German research foundation (DFG), and in part by a research cooperation grant with the Amazon Development Center Germany.

References

Alekh Agarwal, Ofer Dekel, and Liu Xiao. 2010. Optimal algorithms for online convex optimization with multi-point bandit feedback. In *COLT*, Haifa, Israel.

Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert E. Schapire. 2014. Taming the monster: A fast and simple algorithm for contextual bandits. In *ICML*, Beijing, China.

Abhishek Arun, Barry Haddow, and Philipp Koehn. 2010. A unified approach to minimum risk training and decoding. In *Workshop on SMT and Metrics (MATR)*, Uppsala, Sweden.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002a. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256.

Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. 2002b. The nonstochastic multiarmed bandit problem. *SIAM J. on Computing*, 32(1):48–77.

Nicola Bertoldi, Patrick Simianer, Mauro Cettolo, Katharina Wäschle, Marcello Federico, and Stefan Riezler. 2014. Online adaptation to post-edits for phrase-based statistical machine translation. *Machine Translation*, 29:309–339.

Dimitri P. Bertsekas and John N. Tsitsiklis. 1996. *Neuro-Dynamic Programming*. Athena Scientific.

Léon Bottou. 2004. Stochastic learning. In Olivier Bousquet, Ulrike von Luxburg, and Gunnar Rätsch, editors, *Advanced Lectures on Machine Learning*, pages 146–168. Springer, Berlin.

S.R.K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *ACL*, Suntec, Singapore.

Sébastian Bubeck and Nicolò Cesa-Bianchi. 2012. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

Nicolò Cesa-Bianchi and Gábor Lugosi. 2012. Combinatorial bandits. *J. of Computer and System Sciences*, 78:1401–1422.

Kai-Wei Chang, Akshay Krishnamurthy, Alekh Agarwal, Hal Daume, and John Langford. 2015. Learning to search better than your teacher. In *ICML*, Lille, France.

Olivier Chapelle, Eren Masnavoglu, and Romer Rosales. 2014. Simple and scalable response prediction for display advertising. *ACM Trans. on Intelligent Systems and Technology*, 5(4).

David Chiang. 2012. Hope and fear for discriminative training of statistical translation models. *JMLR*, 12:1159–1187.

Corinna Cortes, Mehryar Mohri, and Asish Rastogi. 2007. Magnitude-preserving ranking algorithms. In *ICML*, Corvallis, OR.

Varsha Dani, Thomas P. Hayes, and Sham M. Kakade. 2007. The price of bandit information for online optimization. In *NIPS*, Vancouver, Canada.

Michael Denkowski, Chris Dyer, and Alon Lavie. 2014. Learning from post-editing: Online model adaptation for statistical machine translation. In *EACL*, Gothenburg, Sweden.

John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. 2015. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Translactions on Information Theory*, 61(5):2788–2806.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Jonathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *ACL Demo*, Uppsala, Sweden.

Yoav Freund, Ray Iyer, Robert E. Schapire, and Yoram Singer. 2003. An efficient boosting algorithm for combining preferences. *JMLR*, 4:933–969.

Johannes Fürnkranz and Eyke Hüllermeier. 2010. Preference learning and ranking by pairwise comparison. In Johannes Fürnkranz and Eyke Hüllermeier, editors, *Preference Learning*. Springer.

Saeed Ghadimi and Guanghui Lan. 2012. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM J. on Optimization*, 4(23):2342–2368.

Kevin Gimpel and Noah A. Smith. 2010. Softmax-margin training for structured log-linear models. Technical Report CMU-LTI-10-008, Carnegie Mellon University, Pittsburgh, PA.

Spence Green, Sida I. Wang, Jason Chuang, Jeffrey Heer, Sebastian Schuster, and Christopher D. Manning. 2014. Human effort and machine learnability in computer aided translation. In *EMNLP*, Doha, Qatar.

Xiaodong He and Li Deng. 2012. Maximum expected BLEU training of phrase and lexicon translation models. In *ACL*, Jeju Island, Korea.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 2000. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. Cambridge, MA.

Edward L. Ionides. 2008. Truncated importance sampling. *J. of Comp. and Graph. Stat.*, 17(2):295–311.

Kevin G. Jamieson, Robert D. Nowak, and Benjamin Recht. 2012. Query complexity of derivative-free optimization. In *NIPS*, Lake Tahoe, CA.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *KDD*, New York, NY.

Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. 2008. Efficient bandit algorithms for online multiclass prediction. In *ICML*, Helsinki, Finland.

Philipp Koehn and Josh Schroeder. 2007. Experiments in domain adaptation for statistical machine translation. In *WMT*, Prague, Czech Republic.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, San Francisco, CA.

John Langford and Tong Zhang. 2007. The epoch-greedy algorithm for contextual multi-armed bandits. In *NIPS*, Vancouver, Canada.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A new benchmark collection for text categorization research. *JMLR*, 5:361–397.

Zhifei Li and Jason Eisner. 2009. First-and second-order expectation semirings with applications to minimum-risk training on translation forests. In *EMNLP*, Edinburgh, UK.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *WWW*, Raleigh, NC.

Nick Littlestone. 1989. From on-line to batch learning. In *COLT*, Santa Cruz, CA.

Andreas C. Müller and Sven Behnke. 2014. pystruct - learning structured prediction in python. *JMLR*, 15:2055–2060.

Eric W. Noreen. 1989. *Computer Intensive Methods for Testing Hypotheses. An Introduction*. Wiley, New York.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *HLT-NAACL*, Edmonton, Canada.

Boris T. Polyak and Yakov Z. Tsypkin. 1973. Pseudogradient adaptation and training algorithms. *Automation and remote control*, 34(3):377–397.

Boris T. Polyak. 1964. Some methods of speeding up the convergence of iteration methods. *USSR Comp. Math. and Math. Phys.*, 4(5):1–17.

Boris T. Polyak. 1987. *Introduction to Optimization*. Optimization Software, Inc., New York.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*, San Juan, Puerto Rico.

Kenneth Rose. 1998. Deterministic annealing for clustering, compression, classification, regression and related optimization problems. *IEEE*, 86(11).

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *NAACL*, Edmonton, Canada.

Shai Shalev-Shwartz. 2012. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194.

Patrick Simianer, Stefan Riezler, and Chris Dyer. 2012. Joint feature selection in distributed stochastic learning for large-scale discriminative training in SMT. In *ACL*, Jeju Island, Korea.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *COLING-ACL*, Sydney, Australia.

Artem Sokolov, Stefan Riezler, and Tanguy Urvoy. 2015. Bandit structured prediction for learning from user feedback in statistical machine translation. In *MT Summit XV*, Miami, FL.

Artem Sokolov, Julia Kreutzer, and Stefan Riezler. 2016. Stochastic structured prediction under bandit feedback. *CoRR*, abs/1606.00739.

Mikhail V. Solodov. 1998. Incremental gradient algorithms with stepsizes bounded away from zero. *Computational Optimization and Applications*, 11:23–35.

Ilya Sutskever, James Martens, George E. Dahl, and Geoffrey E. Hinton. 2013. On the importance of initialization and momentum in deep learning. In *ICML*, Atlanta, GA.

Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, Vancouver, Canada.

Csaba Szepesvári. 2009. *Algorithms for Reinforcement Learning*. Morgan & Claypool.

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

Louis Leon Thurstone. 1927. A law of comparative judgement. *Psychological Review*, 34:278–286.

Yisong Yue and Thorsten Joachims. 2009. Interactively optimizing information retrieval systems as a dueling bandits problem. In *ICML*, Montreal, Canada.

Alan Yuille and Xuming He. 2012. Probabilistic models of vision and max-margin methods. *Frontiers of Electrical and Electronic Engineering*, 7(1):94–106.

Deep Reinforcement Learning with a Natural Language Action Space

Ji He*, **Jianshu Chen**†, **Xiaodong He**†, **Jianfeng Gao**†, **Lihong Li**†
Li Deng† **and Mari Ostendorf***
*Department of Electrical Engineering, University of Washington, Seattle, WA 98195, USA
{jvking, ostendor}@uw.edu
†Microsoft Research, Redmond, WA 98052, USA
{jianshuc, xiaohe, jfgao, lihongli, deng}@microsoft.com

Abstract

This paper introduces a novel architecture for reinforcement learning with deep neural networks designed to handle state and action spaces characterized by natural language, as found in text-based games. Termed a deep reinforcement relevance network (DRRN), the architecture represents action and state spaces with separate embedding vectors, which are combined with an interaction function to approximate the Q-function in reinforcement learning. We evaluate the DRRN on two popular text games, showing superior performance over other deep Q-learning architectures. Experiments with paraphrased action descriptions show that the model is extracting meaning rather than simply memorizing strings of text.

1 Introduction

This work is concerned with learning strategies for sequential decision-making tasks, where a system takes actions at a particular state with the goal of maximizing a long-term reward. More specifically, we consider tasks where both the states and the actions are characterized by natural language, such as in human-computer dialog systems, tutoring systems, or text-based games. In a text-based game, for example, the player (or system, in this case) is given a text string that describes the current state of the game and several text strings that describe possible actions one could take. After selecting one of the actions, the environment state is updated and revealed in a new textual description. A reward is given either at each transition or in the end. The objective is to understand, at each step, the state text and all the action texts to pick the most relevant action, navigating through the sequence of texts so as to obtain the highest long-term reward. Here the notion of relevance is based on the joint state/action impact on the reward: an action text string is said to be "more relevant" (to a state text string) than the other action texts if taking that action would lead to a higher long-term reward. Because a player's action changes the environment, reinforcement learning (Sutton and Barto, 1998) is appropriate for modeling long-term dependency in text games.

There is a large body of work on reinforcement learning. Of most interest here are approaches leveraging neural networks because of their success in handling a large state space. Early work — TD-gammon — used a neural network to approximate the state value function (Tesauro, 1995). Recently, inspired by advances in deep learning (LeCun et al., 2015; Hinton et al., 2012; Krizhevsky et al., 2012; Dahl et al., 2012), significant progress has been made by combining deep learning with reinforcement learning. Building on the approach of Q-learning (Watkins and Dayan, 1992), the "Deep Q-Network" (DQN) was developed and applied to Atari games (Mnih et al., 2013; Mnih et al., 2015) and shown to achieve human level performance by applying convolutional neural networks to the raw image pixels. Narasimhan et al. (2015) applied a Long Short-Term Memory network to characterize the state space in a DQN framework for learning control policies for parser-based text games. More recently, Nogueira and Cho (2016) have also proposed a goal-driven web navigation task for language based sequential decision making study. Another stream of work focuses on continuous control with deep reinforcement learning (Lillicrap et al., 2016), where an actor-critic algorithm operates over a known continuous action space.

Inspired by these successes and recent work using neural networks to learn phrase- or sentence-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1621–1630,
Berlin, Germany, August 7-12, 2016.

level embeddings (Collobert and Weston, 2008; Huang et al., 2013; Le and Mikolov, 2014; Sutskever et al., 2014; Kiros et al., 2015), we propose a novel deep architecture for text understanding, which we call a deep reinforcement relevance network (DRRN). The DRRN uses separate deep neural networks to map state and action text strings into embedding vectors, from which "relevance" is measured numerically by a general interaction function, such as their inner product. The output of this interaction function defines the value of the Q-function for the current state-action pair, which characterizes the optimal long-term reward for pairing these two text strings. The Q-function approximation is learned in an end-to-end manner by Q-learning.

The DRRN differs from prior work in that earlier studies mostly considered action spaces that are bounded and known. For actions described by natural language text strings, the action space is inherently discrete and potentially unbounded due to the exponential complexity of language with respect to sentence length. A distinguishing aspect of the DRRN architecture — compared to simple DQN extensions — is that two different types of meaning representations are learned, reflecting the tendency for state texts to describe scenes and action texts to describe potential actions from the user. We show that the DRRN learns a continuous space representation of actions that successfully generalize to paraphrased descriptions of actions unseen in training.

2 Deep Reinforcement Relevance Network

2.1 Text Games and Q-learning

We consider the sequential decision making problem for text understanding. At each time step t, the agent will receive a string of text that describes the state s_t (i.e., "state-text") and several strings of text that describe all the potential actions a_t (i.e., "action-text"). The agent attempts to understand the texts from both the state side and the action side, measuring their relevance to the current context s_t for the purpose of maximizing the long-term reward, and then picking the best action. Then, the environment state is updated $s_{t+1} = s'$ according to the probability $p(s'|s, a)$, and the agent receives a reward r_t for that particular transition. The *policy* of the agent is defined to be the probability $\pi(a_t|s_t)$ of taking action a_t at state s_t. Define the Q-function $Q^\pi(s, a)$ as the expected return starting from s, taking the action a, and thereafter following policy $\pi(a|s)$ to be:

$$Q^\pi(s,a) = \mathbb{E}\left\{\sum_{k=0}^{+\infty} \gamma^k r_{t+k} \middle| s_t = s, a_t = a\right\}$$

where γ denotes a discount factor. The optimal policy and Q-function can be found by using the Q-learning algorithm (Watkins and Dayan, 1992):

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \eta_t \cdot \left(r_t + \gamma \cdot \max_a Q(s_{t+1},a) - Q(s_t,a_t)\right) \quad (1)$$

where η_t is the learning rate of the algorithm. In this paper, we use a softmax selection strategy as the exploration policy during the learning stage, which chooses the action a_t at state s_t according to the following probability:

$$\pi(a_t = a_t^i|s_t) = \frac{\exp(\alpha \cdot Q(s_t, a_t^i))}{\sum_{j=1}^{|\mathcal{A}_t|} \exp(\alpha \cdot Q(s_t, a_t^j))}, \quad (2)$$

where $\mathcal{A}_t$ is the set of feasible actions at state s_t, a_t^i is the i-th feasible action in $\mathcal{A}_t$, $|\cdot|$ denotes the cardinality of the set, and α is the scaling factor in the softmax operation. α is kept constant throughout the learning period. All methods are initialized with small random weights, so initial Q-value differences will be small, thus making the Q-learning algorithm more explorative initially. As Q-values better approximate the true values, a reasonable α will make action selection put high probability on the optimal action (exploitation), but still maintain a small exploration probability.

2.2 Natural language action space

Let $\mathcal{S}$ denote the state space, and let $\mathcal{A}$ denote the entire action space that includes all the unique actions over time. A vanilla Q-learning recursion (1) needs to maintain a table of size $|\mathcal{S}| \times |\mathcal{A}|$, which is problematic for a large state/action space. Prior work using a DNN in Q-function approximation has shown high capacity and scalability for handling a large state space, but most studies have used a network that generates $|\mathcal{A}|$ outputs, each of which represents the value of $Q(s, a)$ for a particular action a. It is not practical to have a DQN architecture of a size that is explicitly dependence on the large number of natural language actions. Further, in many text games, the feasible action set $\mathcal{A}_t$ at each time t is an unknown subset of the unbounded action space $\mathcal{A}$ that varies over time.

For the case where the maximum number of possible actions at any point in time ($\max_t |\mathcal{A}_t|$) is known, the DQN can be modified to simply use that number of outputs ("Max-action DQN"), as illustrated in Figure 1(a), where the state and action vectors are concatenated (i.e., as an extended state vector) as its input. The network computes the Q-function values for the actions in the current feasible set as its outputs. For a complex game, $\max_t |\mathcal{A}_t|$ may be difficult to obtain, because $\mathcal{A}_t$ is usually unknown beforehand. Nevertheless, we will use this modified DQN as a baseline.

An alternative approach is to use a function approximation using a neural network that takes a state-action pair as input, and outputs a single Q-value for each possible action ("Per-action DQN" in Figure 1(b)). This architecture easily handles a varying number of actions and represents a second baseline.

We propose an alternative architecture for handling a natural language action space in sequential text understanding: the deep reinforcement relevance network (DRRN). As shown in Figure 1(c), the DRRN consists of a pair of DNNs, one for the state text embedding and the other for action text embeddings, which are combined using a pairwise interaction function. The texts used to describe states and actions could be very different in nature, e.g., a state text could be long, containing sentences with complex linguistic structure, whereas an action text could be very concise or just a verb phrase. Therefore, it is desirable to use two networks with different structures to handle state/action texts, respectively. As we will see in the experimental sections, by using two separate deep neural networks for state and action sides, we obtain much better results.

2.3 DRRN architecture: Forward activation

Given any state/action text pair (s_t, a_t^i), the DRRN estimates the Q-function $Q(s_t, a_t^i)$ in two steps. First, map both s_t and a_t^i to their embedding vectors using the corresponding DNNs, respectively. Second, approximate $Q(s_t, a_t^i)$ using an interaction function such as the inner product of the embedding vectors. Then, given a particular state s_t, we can select the optimal action a_t among the set of actions via $a_t = \arg\max_{a_t^i} Q(s_t, a_t^i)$.

More formally, let $h_{l,s}$ and $h_{l,a}$ denote the l-th hidden layer for state and action side neural networks, respectively. For the state side, $W_{l,s}$ and $b_{l,s}$ denote the linear transformation weight matrix and bias vector between the $(l-1)$-th and l-th hidden layers. $W_{l,a}$ and $b_{l,a}$ denote the equivalent parameters for the action side. In this study, the DRRN has L hidden layers on each side.

$$h_{1,s} = f(W_{1,s}s_t + b_{1,s}) \quad (3)$$

$$h_{1,a}^i = f(W_{1,a}a_t^i + b_{1,a}) \quad (4)$$

$$h_{l,s} = f(W_{l-1,s}h_{l-1,s} + b_{l-1,s}) \quad (5)$$

$$h_{l,a}^i = f(W_{l-1,a}h_{l-1,a}^i + b_{l-1,a}) \quad (6)$$

where $f(\cdot)$ is the nonlinear activation function at the hidden layers, which, for example, could be chosen as $\tanh(x)$, and $i = 1, 2, 3, ..., |\mathcal{A}_t|$ is the action index. A general interaction function $g(\cdot)$ is used to approximate the Q-function values, $Q(s, a)$, in the following parametric form:

$$Q(s, a^i; \Theta) = g\left(h_{L,s},\ h_{L,a}^i\right) \quad (7)$$

where Θ denotes all the model parameters. The interaction function could be an inner product, a bilinear operation, or a nonlinear function such as a deep neural network. In our experiments, the inner product and bilinear operation gave similar results. For simplicity, we present our experiments mostly using the inner product interaction function.

The success of the DRRN in handling a natural language action space $\mathcal{A}$ lies in the fact that the state-text and the action-texts are mapped into separate finite-dimensional embedding spaces. The end-to-end learning process (discussed next) makes the embedding vectors in the two spaces more aligned for "good" (or relevant) action texts compared to "bad" (or irrelevant) choices, resulting in a higher interaction function output (Q-function value).

2.4 Learning the DRRN: Back propagation

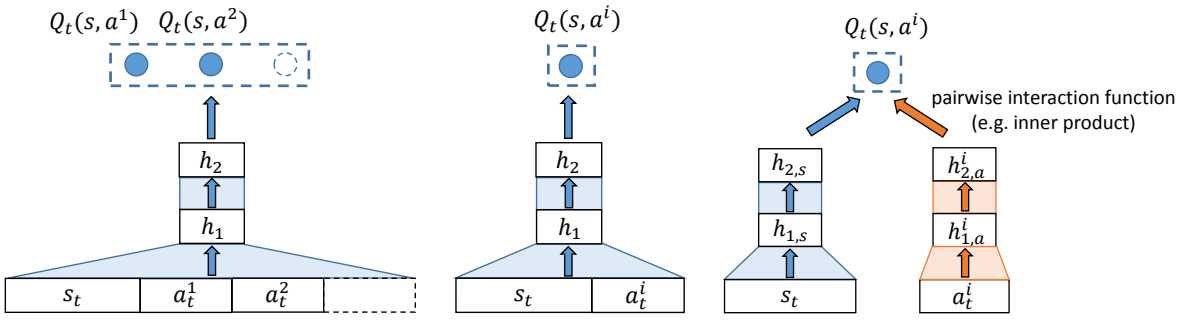
To learn the DRRN, we use the "experience-replay" strategy (Lin, 1993), which uses a fixed exploration policy to interact with the environment to obtain a sample trajectory. Then, we randomly sample a transition tuple (s_k, a_k, r_k, s_{k+1}), compute the temporal difference error for sample k:

$$d_k = r_k + \gamma \max_a Q(s_{k+1}, a; \Theta_{k-1}) - Q(s_k, a_k; \Theta_{k-1}),$$

and update the model according to the recursions:

$$W_{v,k} = W_{v,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial W_v} \quad (8)$$

$$b_{v,k} = b_{v,k-1} + \eta_k d_k \cdot \frac{\partial Q(s_k, a_k; \Theta_{k-1})}{\partial b_v} \quad (9)$$



(a) Max-action DQN (b) Per-action DQN (c) DRRN

Figure 1: Different deep Q-learning architectures: Max-action DQN and Per-action DQN both treat input text as concantenated vectors and compute output Q-values with a single NN. DRRN models text embeddings from state/action sides separately, and use an interaction function to compute Q-values.

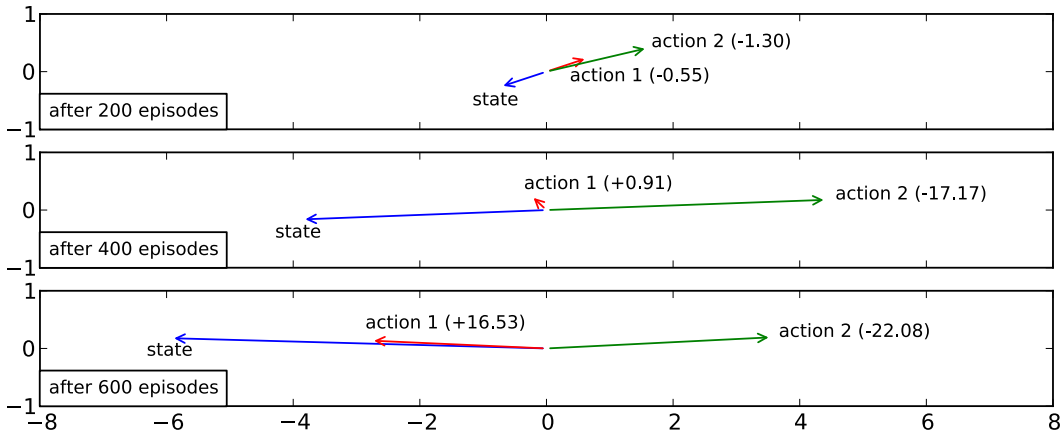


Figure 2: PCA projections of text embedding vectors for state and associated action vectors after 200, 400 and 600 training episodes. The state is "As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street." Action 1 (good choice) is "Look up", and action 2 (poor choice) is "Ignore the alarm of others and continue moving forward."

for $v \in \{s, a\}$. Expressions for $\frac{\partial Q}{\partial W_v}$, $\frac{\partial Q}{\partial b_v}$ and other algorithm details are given in supplementary materials. Random sampling essentially scrambles the trajectory from experience-replay into a "bag-of-transitions", which has been shown to avoid oscillations or divergence and achieve faster convergence in Q-learning (Mnih et al., 2015). Since the models on the action side share the same parameters, models associated with all actions are effectively updated even though the back propagation is only over one action. We apply back propagation to learn how to pair the text strings from the reward signals in an end-to-end manner. The representation vectors for the state-text and the action-text are *automatically learned* to be aligned with each other in the text embedding space from the reward signals. A summary of the full learning algorithm is given in Algorithm 1.

Figure 2 illustrates learning with an inner product interaction function. We used Principal Component Analysis (PCA) to project the 100-dimension last hidden layer representation (before the inner product) to a 2-D plane. The vector embeddings start with small values, and after 600 episodes of experience-replay training, the embeddings are very close to the converged embedding (4000 episodes). The embedding vector of the optimal action (Action 1) converges to a positive inner product with the state embedding vector, while Action 2 converges to a negative inner product.

3 Experimental Results

3.1 Text games

Text games, although simple compared to video games, still enjoy high popularity in online communities, with annual competitions held online

Algorithm 1 Learning algorithm for DRRN

Initialize replay memory $\mathcal{D}$ to capacity N.
Initialize DRRN with small random weights.
Initialize game simulator and load dictionary.
for $episode = 1, \ldots, M$ **do**
  Restart game simulator.
  Read raw state text and a list of action text from the simulator, and convert them to representation s_1 and $a_1^1, a_1^2, \ldots, a_1^{|\mathcal{A}_1|}$.
  for $t = 1, \ldots, T$ **do**
    Compute $Q(s_t, a_t^i; \Theta)$ for the list of actions using DRRN forward activation (Section 2.3).
    Select an action a_t based on probability distribution $\pi(a_t = a_t^i | s_t)$ (Equation 2)
    Execute action a_t in simulator
    Observe reward r_t. Read the next state text and the next list of action texts, and convert them to representation s_{t+1} and $a_{t+1}^1, a_{t+1}^2, \ldots, a_{t+1}^{|\mathcal{A}_{t+1}|}$.
    Store transition $(s_t, a_t, r_t, s_{t+1}, A_{t+1})$ in $\mathcal{D}$.
    Sample random mini batch of transitions $(s_k, a_k, r_k, s_{k+1}, A_{k+1})$ from $\mathcal{D}$.
    Set $y_k = \begin{cases} r_k & \text{if } s_{k+1} \text{ is terminal} \\ r_k + \gamma \max_{a' \in A_{k+1}} Q(s_{k+1}, a'; \Theta)) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_k - Q(s_k, a_k; \Theta))^2$ with respect to the network parameters Θ (Section 2.4). Back-propagation is performed only for a_k even though there are $|\mathcal{A}_k|$ actions at time k.
  end for
end for

since 1995. Text games communicate to players in the form of a text display, which players have to understand and respond to by typing or clicking text (Adams, 2014). There are three types of text games: parser-based (Figure 3(a)), choice-based (Figure 3(b)), and hypertext-based (Figure 3(c)). Parser-based games accept typed-in commands from the player, usually in the form of verb phrases, such as "eat apple", "get key", or "go east". They involve the least complex action language. Choice-based and hypertext-based games present actions after or embedded within the state text. The player chooses an action, and the story continues based on the action taken at this particular state. With the development of web browsing and richer HTML display, choice-based and hypertext-based text games have become more popular, increasing in percentage from 8% in 2010 to 62% in 2014.[1]

For parser-based text games, Narasimhan et al. (2015) have defined a fixed set of 222 actions, which is the total number of possible phrases the parser accepts. Thus the parser-based text game is reduced to a problem that is well suited to a fixed-action-set DQN. However, for choice-based and hypertext-based text games, the size of the action space could be exponential with the length of the action sentences, which is handled here by using a continuous representation of the action space.

Game	Saving John	Machine of Death
Text game type	Choice	Choice & Hypertext
Vocab size	1762	2258
Action vocab size	171	419
Avg. words/description	76.67	67.80
State transitions	Deterministic	Stochastic
# of states (underlying)	≥ 70	≥ 200

Table 1: Statistics for the games "Saving John" and and "Machine of Death".

In this study, we evaluate the DRRN with two games: a deterministic text game task called "Saving John" and a larger-scale stochastic text game called "Machine of Death" from a public archive.[2] The basic text statistics of these tasks are shown in Table 1. The maximum value of feasible actions (i.e., $\max_t |\mathcal{A}_t|$) is four in "Saving John", and nine in "Machine of Death". We manually annotate fi-

[1] Statistics obtained from `http://www.ifarchive.org`

[2] Simulators are available at `https://github.com/jvking/text-games`

(a) Parser-based (b) Choiced-based (c) Hypertext-based

Figure 3: Different types of text games

nal rewards for all distinct endings in both games (as shown in supplementary materials). The magnitude of reward scores are given to describe sentiment polarity of good/bad endings. On the other hand, each non-terminating step we assign with a small negative reward, to encourage the learner to finish the game as soon as possible. For the text game "Machine of Death", we restrict an episode to be no longer than 500 steps.

In "Saving John" all actions are choice-based, for which the mapping from text strings to a_t are clear. In "Machine of Death", when actions are hypertext, the actions are substrings of the state. In this case s_t is associated with the full state description, and a_t are given by the substrings without any surrounding context. For text input, we use raw bag-of-words as features, with different vocabularies for the state side and action side.

3.2 Experiment setup

We apply DRRNs with both 1 and 2 hidden layer structures. In most experiments, we use dot-product as the interaction function and set the hidden dimension to be the same for each hidden layer. We use DRRNs with 20, 50 and 100-dimension hidden layer(s) and build learning curves during experience-replay training. The learning rate is constant: $\eta_t = 0.001$. In testing, as in training, we apply softmax selection. We record average final rewards as performance of the model.

The DRRN is compared to multiple baselines: a linear model, two max-action DQNs (MA DQN) ($L = 1$ or 2 hidden layers), and two per-action DQNs (PA DQN) (again, $L = 1, 2$). All baselines use the same Q-learning framework with different function approximators to predict $Q(s_t, a_t)$ given the current state and actions. For the linear and MA DQN baselines, the input is the text-based state and action descriptions, each as a bag of words, with the number of outputs equal to the maximum number of actions. When there are fewer actions than the maximum, the highest scoring available action is used. The PA DQN baseline takes each pair of state-action texts as input, and generates a corresponding Q-value.

Eval metric	Average reward		
hidden dimension	20	50	100
Linear	4.4 (0.4)		
PA DQN ($L = 1$)	2.0 (1.5)	4.0 (1.4)	4.4 (2.0)
PA DQN ($L = 2$)	1.5 (3.0)	4.5 (2.5)	7.9 (3.0)
MA DQN ($L = 1$)	2.9 (3.1)	4.0 (4.2)	5.9 (2.5)
MA DQN ($L = 2$)	4.9 (3.2)	9.0 (3.2)	7.1 (3.1)
DRRN ($L = 1$)	17.1 (0.6)	18.3 (0.2)	18.2 (0.2)
DRRN ($L = 2$)	18.4 (0.1)	18.5 (0.3)	**18.7** (0.4)

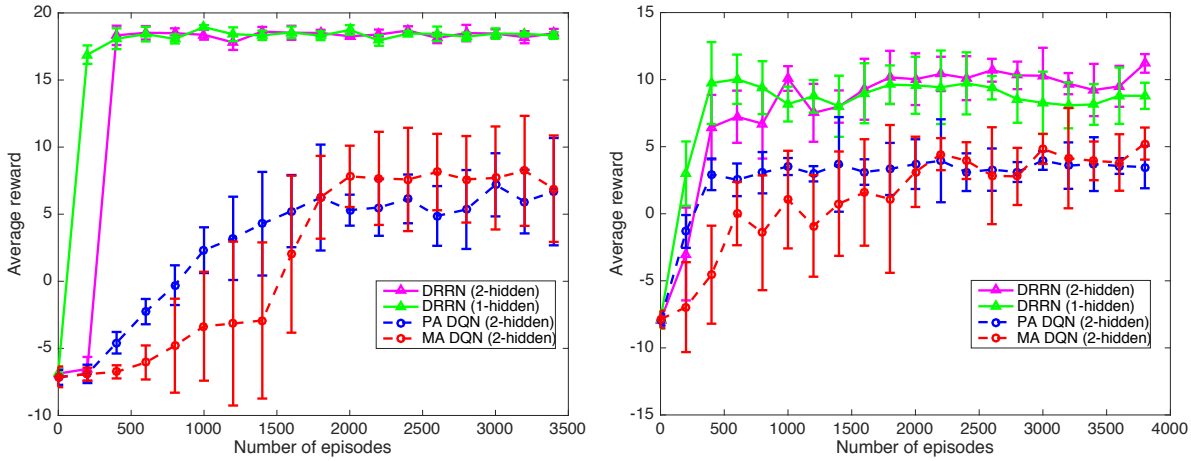
Table 2: The final average rewards and standard deviations on "Saving John".

We use softmax selection, which is widely applied in practice, to trade-off exploration vs. exploitation. Specifically, for each experience-replay, we first generate 200 episodes of data (about 3K tuples in "Saving John" and 16K tuples in "Machine of Death") using the softmax selection rule in (2), where we set $\alpha = 0.2$ for the first game and $\alpha = 1.0$ for the second game. The α is picked according to an estimation of range of the optimal Q-values. We then shuffle the generated data tuples (s_t, a_t, r_t, s_{t+1}) update the model as described in Section 2.4. The model is trained with multiple epochs for all configurations, and is evaluated after each experience-replay. The discount factor γ is set to 0.9. For DRRN and all baselines, network weights are initialized with small random values. To prevent algorithms from "remembering" state-action ordering and make choices based on action wording, each time the algorithm/player reads text from the simulator, we randomly shuffle the list of actions.[3] This will encourage the algorithms to make decisions based on the understanding of the texts that describe the states and actions.

3.3 Performance

In Figure 4, we show the learning curves of different models, where the dimension of the hid-

[3]When in a specific state, the simulator presents the possible set of actions in random order, i.e. they may appear in a different order the next time a player is in this same state.



(a) Game 1: "Saving John" (b) Game 2: "Machine of Death"

Figure 4: Learning curves of the two text games.

Eval metric	Average reward		
hidden dimension	20	50	100
Linear	3.3 (1.0)		
PA DQN ($L = 1$)	0.9 (2.4)	2.3 (0.9)	3.1 (1.3)
PA DQN ($L = 2$)	1.3 (1.2)	2.3 (1.6)	3.4 (1.7)
MA DQN ($L = 1$)	2.0 (1.2)	3.7 (1.6)	4.8 (2.9)
MA DQN ($L = 2$)	2.8 (0.9)	4.3 (0.9)	5.2 (1.2)
DRRN ($L = 1$)	7.2 (1.5)	8.4 (1.3)	8.7 (0.9)
DRRN ($L = 2$)	9.2 (2.1)	10.7 (2.7)	**11.2** (0.6)

Table 3: The final average rewards and standard deviations on "Machine of Death".

den layers in the DQNs and DRRN are all set to 100. The error bars are obtained by running 5 independent experiments. The proposed methods and baselines all start at about the same performance (roughly -7 average rewards for Game 1, and roughly -8 average rewards for Game 2), which is the random guess policy. After around 4000 episodes of experience-replay training, all methods converge. The DRRN converges much faster than the other three baselines and achieves a higher average reward. We hypothesize this is because the DRRN architecture is better at capturing relevance between state text and action text. The faster convergence for "Saving John" may be due to the smaller observation space and/or the deterministic nature of its state transitions (in contrast to the stochastic transitions in the other game).

The final performance (at convergence) for both baselines and proposed methods are shown in Tables 2 and 3. We test for different model sizes with 20, 50, and 100 dimensions in the hidden layers. The DRRN performs consistently better than all baselines, and often with a lower variance. For Game 2, due to the complexity of the underlying state transition function, we cannot compute the exact optimal policy score. To provide more insight into the performance, we averaged scores of 8 human players for initial trials (novice) and after gaining experience, yielding scores of -5.5 and 16.0, respectively. The experienced players do outperform our algorithm. The converged performance is higher with two hidden layers for all models. However, deep models also converge more slowly than their 1 hidden layer versions, as shown for the DRRN in Figure 4.

Besides an inner-product, we also experimented with more complex interaction functions: a) a bilinear operation with different action side dimensions; and b) a non-linear deep neural network using the concatenated state and action space embeddings as input and trained in an end-to-end fashion to predict Q values. For different configurations, we fix the state side embedding to be 100 dimensions and vary the action side embedding dimensions. The bilinear operation gave similar results, but the concatenation input to a DNN degraded performance. Similar behaviors have been observed on a different task (Luong et al., 2015).

3.4 Actions with paraphrased descriptions

To investigate how our models handle actions with "unseen" natural language descriptions, we had two people paraphrase all actions in the game "Machine of Death" (used in testing phase), except a few single-word actions whose synonyms are out-of-vocabulary (OOV). The word-level OOV rate of paraphrased actions is 18.6%,

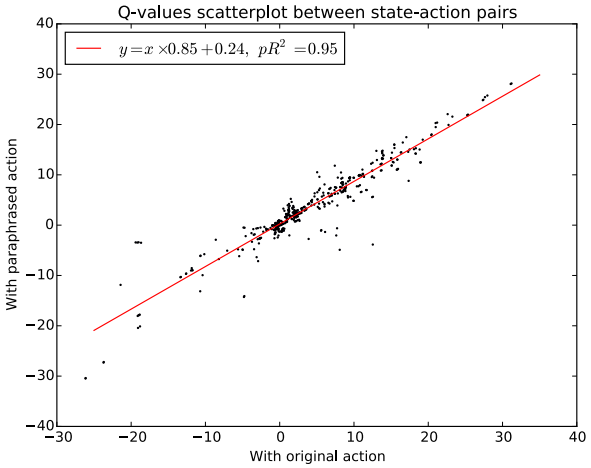


Figure 5: Scatterplot and strong correlation between Q-values of paraphrased actions versus original actions

and standard 4-gram BLEU score between the paraphrased and original actions is 0.325. The resulting 153 paraphrased action descriptions are associated with 532 unique state-action pairs.

We apply a well-trained 2-layer DRRN model (with hidden dimension 100), and predict Q-values for each state-action pair with fixed model parameters. Figure 5 shows the correlation between Q-values associated with paraphrased actions versus original actions. The predictive R-squared is 0.95, showing a strong positive correlation. We also run Q-value correlation for the NN interaction and $pR^2 = 0.90$. For baseline MA-DQN and PA-DQN, their corresponding pR^2 is 0.84 and 0.97, indicating they also have some generalization ability. This is confirmed in the paraphrasing-based experiments too, where the test reward on the paraphrased setup is close to the original setup. This supports the claim that deep learning is useful in general for this language understanding task, and our findings show that a decoupled architecture most effectively leverages that approach.

In Table 4 we provide examples with predicted Q-values of original descriptions and paraphrased descriptions. We also include alternative action descriptions with in-vocabulary words that will lead to positive / negative / irrelevant game development at that particular state. Table 4 shows actions that are more likely to result in good endings are predicted with high Q-values. This indicates that the DRRN has some generalization ability and gains a useful level of language understanding in the game scenario.

We use the baseline models and proposed DRRN model trained with the original action descriptions for "Machine of Death", and test on paraphrased action descriptions. For this game, the underlying state transition mechanism has not changed. The only change to the game interface is that during testing, every time the player reads the actions from the game simulator, it reads the paraphrased descriptions and performs selection based on these paraphrases. Since the texts in test time are "unseen" to the player, a good model needs to have some level of language understanding, while a naive model that memorizes all unique action texts in the original game will do poorly. The results for these models are shown in Table 5. All methods have a slightly lower average reward in this setting (10.5 vs. 11.2 for the original actions), but the DRRN still gives a high reward and significantly outperforms other methods. This shows that the DRRN can generalize well to "unseen" natural language descriptions of actions.

4 Related Work

There has been increasing interest in applying deep reinforcement learning to a variety problems, but only a few studies address problems with natural language state or action spaces. In language processing, reinforcement learning has been applied to a dialogue management system that converses with a human user by taking actions that generate natural language (Scheffler and Young, 2002; Young et al., 2013). There has also been interest in extracting textual knowledge to improve game control performance (Branavan et al., 2011), and mapping text instructions to sequences of executable actions (Branavan et al., 2009). In some applications, it is possible to manually design features for state-action pairs, which are then used in reinforcement learning to learn a near-optimal policy (Li et al., 2009). Designing such features, however, require substantial domain knowledge.

The work most closely related to our study involves application of deep reinforcement to learning decision policies for parser-based text games. Narasimhan et al. (2015) applied a Long Short-Term Memory DQN framework, which achieves higher average reward than the random and Bag-of-Words DQN baselines. In this work, actions are constrained to a set of known fixed command structures (one action and one argument object),

	Text (with predicted Q-values)
State	As you move forward, the people surrounding you suddenly look up with terror in their faces, and flee the street.
Actions in the original game	Ignore the alarm of others and continue moving forward. (-21.5) Look up. (16.6)
Paraphrased actions (not original)	Disregard the caution of others and keep pushing ahead. (-11.9) Turn up and look. (17.5)
Positive actions (not original)	Stay there. (2.8) Stay calmly. (2.0)
Negative actions (not original)	Screw it. I'm going carefully. (-17.4) Yell at everyone. (-13.5)
Irrelevant actions (not original)	Insert a coin. (-1.4) Throw a coin to the ground. (-3.6)

Table 4: Predicted Q-value examples

Eval metric	Average reward		
hidden dimension	20	50	100
PA DQN ($L = 2$)	0.2 (1.2)	2.6 (1.0)	3.6 (0.3)
MA DQN ($L = 2$)	2.5 (1.3)	4.0 (0.9)	5.1 (1.1)
DRRN ($L = 2$)	7.3 (0.7)	8.3 (0.7)	**10.5 (0.9)**

Table 5: The final average rewards and standard deviations on paraphrased game "Machine of Death".

based on a limited action-side vocabulary size. The overall action space is defined by the action-argument product space. This pre-specified product space is not feasible for the more complex text strings in other forms of text-based games. Our proposed DRRN, on the other hand, can handle the more complex text strings, as well as parser-based games. In preliminary experiments with the parser-based game from (Narasimhan et al., 2015), we find that the DRRN using a bag-of-words (BOW) input achieves results on par with their BOW DQN. The main advantage of the DRRN is that it can also handle actions described with more complex language.

The DRRN experiments described here leverage only a simple bag-of-words representation of phrases and sentences. As observed in (Narasimhan et al., 2015), more complex sentence-based models can give further improvements. In preliminary experiments with "Machine of Death", we did not find LSTMs to give improved performance, but we conjecture that they would be useful in larger-scale tasks, or when the word embeddings are initialized by training on large data sets.

As mentioned earlier, other work has applied deep reinforcement learning to a problem with a continuous action space (Lillicrap et al., 2016). In the DRRN, the action space is inherently discrete, but we learn a continuous representation of it. As indicated by the paraphrasing experiment, the continuous space representation seems to generalize reasonably well.

5 Conclusion

In this paper we develop a deep reinforcement relevance network, a novel DNN architecture for handling actions described by natural language in decision-making tasks such as text games. We show that the DRRN converges faster and to a better solution for Q-learning than alternative architectures that do not use separate embeddings for the state and action spaces. Future work includes: (i) adding an attention model to robustly analyze which part of state/actions text correspond to strategic planning, and (ii) applying the proposed methods to more complex text games or other tasks with actions defined through natural language.

Acknowledgments

We thank Karthik Narasimhan and Tejas Kulkarni for providing instructions on setting up their parser-based games.

References

E. Adams. 2014. *Fundamentals of game design*. Pearson Education.

S.R.K. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for mapping instructions to actions. In *Proc. of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th IJCNLP*, pages 82–90, August.

S.R.K. Branavan, D. Silver, and R. Barzilay. 2011. Learning to win by reading manuals in a monte-carlo framework. In *Proc. of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 268–277. Association for Computational Linguistics.

R. Collobert and J. Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of the 25th International Conference on Machine learning*, pages 160–167. ACM.

G. E Dahl, D. Yu, L. Deng, and A. Acero. 2012. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, 29(6):82–97.

P-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proc. of the ACM International Conference on Information & Knowledge Management*, pages 2333–2338. ACM.

R. Kiros, Y. Zhu, R. R Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3276–3284.

A. Krizhevsky, I. Sutskever, and G. E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Q. V Le and T. Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*.

Y. LeCun, Y. Bengio, and G. Hinton. 2015. Deep learning. *Nature*, 521(7553):436–444.

L. Li, J. D. Williams, and S. Balakrishnan. 2009. Reinforcement learning for spoken dialog management using least-squares policy iteration and fast feature selection. In *Proceedings of the Tenth Annual Conference of the International Speech Communication Association (INTERSPEECH-09)*, page 24752478.

T. P Lillicrap, J. J Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.

L-J. Lin. 1993. Reinforcement learning for robots using neural networks. Technical report, DTIC Document.

M-T. Luong, H. Pham, and C. D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, September.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. *NIPS Deep Learning Workshop*, December.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A Rusu, J. Veness, M. G Bellemare, A. Graves, M. Riedmiller, A. K Fidjeland, G. Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

K. Narasimhan, T. Kulkarni, and R. Barzilay. 2015. Language understanding for text-based games using deep reinforcement learning. In *Proc. of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, September.

R. Nogueira and K. Cho. 2016. Webnav: A new large-scale task for natural language based sequential decision making. *arXiv preprint arXiv:1602.02261*.

K. Scheffler and S. Young. 2002. Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proc. of the second International Conference on Human Language Technology Research*, pages 12–19.

I. Sutskever, O. Vinyals, and Q. V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112.

R. S Sutton and A. G Barto. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

G. Tesauro. 1995. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

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

S. Young, M. Gasic, B. Thomson, and J. D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Incorporating Copying Mechanism in Sequence-to-Sequence Learning

Jiatao Gu[†] **Zhengdong Lu**[‡] **Hang Li**[‡] **Victor O.K. Li**[†]
[†]Department of Electrical and Electronic Engineering, The University of Hong Kong
{jiataogu, vli}@eee.hku.hk
[‡]Huawei Noah's Ark Lab, Hong Kong
{lu.zhengdong, hangli.hl}@huawei.com

Abstract

We address an important problem in sequence-to-sequence (Seq2Seq) learning referred to as copying, in which certain segments in the input sequence are selectively replicated in the output sequence. A similar phenomenon is observable in human language communication. For example, humans tend to repeat entity names or even long phrases in conversation. The challenge with regard to copying in Seq2Seq is that new machinery is needed to decide when to perform the operation. In this paper, we incorporate copying into neural network-based Seq2Seq learning and propose a new model called COPYNET with encoder-decoder structure. COPYNET can nicely integrate the regular way of word generation in the decoder with the new copying mechanism which can choose subsequences in the input sequence and put them at proper places in the output sequence. Our empirical study on both synthetic data sets and real world data sets demonstrates the efficacy of COPYNET. For example, COPYNET can outperform regular RNN-based model with remarkable margins on text summarization tasks.

1 Introduction

Recently, neural network-based sequence-to-sequence learning (Seq2Seq) has achieved remarkable success in various natural language processing (NLP) tasks, including but not limited to Machine Translation (Cho et al., 2014; Bahdanau et al., 2014), Syntactic Parsing (Vinyals et al., 2015b), Text Summarization (Rush et al., 2015) and Dialogue Systems (Vinyals and Le, 2015). Seq2Seq is essentially an encoder-decoder model, in which the encoder first transform the input sequence to a certain representation which can then transform the representation into the output sequence. Adding the attention mechanism (Bahdanau et al., 2014) to Seq2Seq, first proposed for automatic alignment in machine translation, has led to significant improvement on the performance of various tasks (Shang et al., 2015; Rush et al., 2015). Different from the canonical encoder-decoder architecture, the attention-based Seq2Seq model revisits the input sequence in its raw form (array of word representations) and dynamically fetches the relevant piece of information based mostly on the feedback from the generation of the output sequence.

In this paper, we explore another mechanism important to the human language communication, called the "copying mechanism". Basically, it refers to the mechanism that locates a certain segment of the input sentence and puts the segment into the output sequence. For example, in the following two dialogue turns we observe different patterns in which some subsequences (colored blue) in the response (R) are copied from the input utterance (I):

I: Hello Jack, my name is Chandralekha.
R: Nice to meet you, Chandralekha.

I: This new guy doesn't perform exactly as we expected.
R: What do you mean by "doesn't perform exactly as we expected"?

Both the canonical encoder-decoder and its variants with attention mechanism rely heavily on the representation of "meaning", which might not be sufficiently inaccurate in cases in which the system needs to refer to sub-sequences of input like entity names or dates. In contrast, the

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1631–1640,
Berlin, Germany, August 7-12, 2016.

copying mechanism is closer to the rote memorization in language processing of human being, deserving a different modeling strategy in neural network-based models. We argue that it will benefit many Seq2Seq tasks to have an elegant unified model that can accommodate both understanding and rote memorization. Towards this goal, we propose COPYNET, which is not only capable of the regular generation of words but also the operation of copying appropriate segments of the input sequence. Despite the seemingly "hard" operation of copying, COPYNET can be trained in an end-to-end fashion. Our empirical study on both synthetic datasets and real world datasets demonstrates the efficacy of COPYNET.

2 Background: Neural Models for Sequence-to-sequence Learning

Seq2Seq Learning can be expressed in a probabilistic view as maximizing the likelihood (or some other evaluation metrics (Shen et al., 2015)) of observing the output (target) sequence given an input (source) sequence.

2.1 RNN Encoder-Decoder

RNN-based Encoder-Decoder is successfully applied to real world Seq2Seq tasks, first by Cho et al. (2014) and Sutskever et al. (2014), and then by (Vinyals and Le, 2015; Vinyals et al., 2015a). In the Encoder-Decoder framework, the source sequence $X = [x_1, ..., x_{T_S}]$ is converted into a fixed length vector $\mathbf{c}$ by the encoder RNN, i.e.

$$\mathbf{h}_t = f(x_t, \mathbf{h}_{t-1}); \quad \mathbf{c} = \phi(\{\mathbf{h}_1, ..., \mathbf{h}_{T_S}\}) \quad (1)$$

where $\{\mathbf{h}_t\}$ are the RNN states, $\mathbf{c}$ is the so-called context vector, f is the dynamics function, and ϕ summarizes the hidden states, e.g. choosing the last state $\mathbf{h}_{T_S}$. In practice it is found that gated RNN alternatives such as LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014) often perform much better than vanilla ones.

The decoder RNN is to unfold the context vector $\mathbf{c}$ into the target sequence, through the following dynamics and prediction model:

$$\begin{aligned} \mathbf{s}_t &= f(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{c}) \\ p(y_t|y_{<t}, X) &= g(y_{t-1}, \mathbf{s}_t, \mathbf{c}) \end{aligned} \quad (2)$$

where $\mathbf{s}_t$ is the RNN state at time t, y_t is the predicted target symbol at t (through function $g(\cdot)$) with $y_{<t}$ denoting the history $\{y_1, ..., y_{t-1}\}$. The prediction model is typically a classifier over the vocabulary with, say, 30,000 words.

2.2 The Attention Mechanism

The attention mechanism was first introduced to Seq2Seq (Bahdanau et al., 2014) to release the burden of summarizing the entire source into a fixed-length vector as context. Instead, the attention uses a dynamically changing context $\mathbf{c}_t$ in the decoding process. A natural option (or rather "soft attention") is to represent $\mathbf{c}_t$ as the weighted sum of the source hidden states, i.e.

$$\mathbf{c}_t = \sum_{\tau=1}^{T_S} \alpha_{t\tau}\mathbf{h}_\tau; \quad \alpha_{t\tau} = \frac{e^{\eta(\mathbf{s}_{t-1}, \mathbf{h}_\tau)}}{\sum_{\tau'} e^{\eta(s_{t-1}, \mathbf{h}_{\tau'})}} \quad (3)$$

where η is the function that shows the correspondence strength for attention, approximated usually with a multi-layer neural network (DNN). Note that in (Bahdanau et al., 2014) the source sentence is encoded with a Bi-directional RNN, making each hidden state $\mathbf{h}_\tau$ aware of the contextual information from both ends.

3 COPYNET

From a cognitive perspective, the copying mechanism is related to rote memorization, requiring less understanding but ensuring high literal fidelity. From a modeling perspective, the copying operations are more rigid and symbolic, making it more difficult than soft attention mechanism to integrate into a fully differentiable neural model. In this section, we present COPYNET, a differentiable Seq2Seq model with "copying mechanism", which can be trained in an end-to-end fashion with just gradient descent.

3.1 Model Overview

As illustrated in Figure 1, COPYNET is still an encoder-decoder (in a slightly generalized sense). The source sequence is transformed by **Encoder** into representation, which is then read by **Decoder** to generate the target sequence.

Encoder: Same as in (Bahdanau et al., 2014), a bi-directional RNN is used to transform the source sequence into a series of hidden states with equal length, with each hidden state $\mathbf{h}_t$ corresponding to word x_t. This new representation of the source, $\{\mathbf{h}_1, ..., \mathbf{h}_{T_S}\}$, is considered to be a short-term memory (referred to as $\mathbf{M}$ in the remainder of the paper), which will later be accessed in multiple ways in generating the target sequence (decoding).

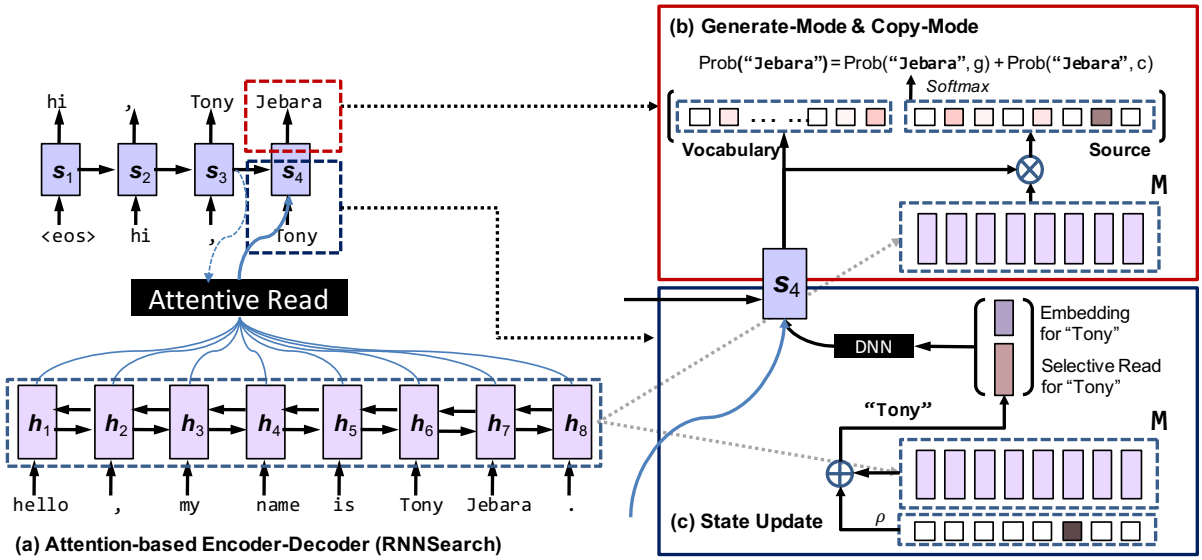


Figure 1: The overall diagram of COPYNET. For simplicity, we omit some links for prediction (see Sections 3.2 for more details).

Decoder: An RNN that reads $\mathbf{M}$ and predicts the target sequence. It is similar with the canonical RNN-decoder in (Bahdanau et al., 2014), with however the following important differences

- **Prediction:** COPYNET predicts words based on a mixed probabilistic model of two modes, namely the **generate-mode** and the **copy-mode**, where the latter picks words from the source sequence (see Section 3.2);
- **State Update:** the predicted word at time t–1 is used in updating the state at t, but COPYNET uses not only its word-embedding but also its corresponding location-specific hidden state in $\mathbf{M}$ (if any) (see Section 3.3 for more details);
- **Reading $\mathbf{M}$:** in addition to the attentive read to $\mathbf{M}$, COPYNET also has"selective read" to $\mathbf{M}$, which leads to a powerful hybrid of content-based addressing and location-based addressing (see both Sections 3.3 and 3.4 for more discussion).

3.2 Prediction with Copying and Generation

We assume a vocabulary $\mathcal{V} = \{v_1, ..., v_N\}$, and use UNK for any out-of-vocabulary (OOV) word. In addition, we have another set of words $\mathcal{X}$, for all the *unique* words in source sequence $X = \{x_1, ..., x_{T_S}\}$. Since $\mathcal{X}$ may contain words not in $\mathcal{V}$, copying sub-sequence in X enables COPYNET to output some OOV words. In a nutshell, the instance-specific vocabulary for source X is $\mathcal{V} \cup \text{UNK} \cup \mathcal{X}$.

Given the decoder RNN state $\mathbf{s}_t$ at time t together with $\mathbf{M}$, the probability of generating any target word y_t, is given by the "mixture" of probabilities as follows

$$p(y_t|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) = p(y_t, \mathsf{g}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) + p(y_t, \mathsf{c}|\mathbf{s}_t, y_{t-1}, \mathbf{c}_t, \mathbf{M}) \quad (4)$$

where g stands for the generate-mode, and c the copy mode. The probability of the two modes are given respectively by

$$p(y_t, \mathsf{g}|\cdot) = \begin{cases} \frac{1}{Z} e^{\psi_g(y_t)}, & y_t \in \mathcal{V} \\ 0, & y_t \in \mathcal{X} \cap \bar{V} \\ \frac{1}{Z} e^{\psi_g(\text{UNK})} & y_t \notin \mathcal{V} \cup \mathcal{X} \end{cases} \quad (5)$$

$$p(y_t, \mathsf{c}|\cdot) = \begin{cases} \frac{1}{Z} \sum_{j:x_j=y_t} e^{\psi_c(x_j)}, & y_t \in \mathcal{X} \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where $\psi_g(\cdot)$ and $\psi_c(\cdot)$ are score functions for generate-mode and copy-mode, respectively, and Z is the normalization term shared by the two modes, $Z = \sum_{v \in \mathcal{V} \cup \{\text{UNK}\}} e^{\psi_g(v)} + \sum_{x \in X} e^{\psi_c(x)}$. Due to the shared normalization term, the two modes are basically competing through a softmax function (see Figure 1 for an illustration with example), rendering Eq.(4) different from the canonical definition of the mixture model (McLachlan and Basford, 1988). This is also pictorially illustrated in Figure 2. The score of each mode is calculated:

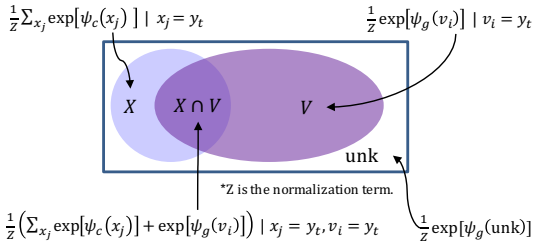


Figure 2: The illustration of the decoding probability $p(y_t|\cdot)$ as a 4-class classifier.

Generate-Mode: The same scoring function as in the generic RNN encoder-decoder (Bahdanau et al., 2014) is used, i.e.

$$\psi_g(y_t = v_i) = \mathbf{v}_i^\top \mathbf{W}_o \mathbf{s}_t, \quad v_i \in \mathcal{V} \cup \text{UNK} \quad (7)$$

where $\mathbf{W}_o \in \mathbb{R}^{(N+1)\times d_s}$ and $\mathbf{v}_i$ is the one-hot indicator vector for v_i.

Copy-Mode: The score for "copying" the word x_j is calculated as

$$\psi_c(y_t = x_j) = \sigma\left(\mathbf{h}_j^\top \mathbf{W}_c\right)\mathbf{s}_t, \quad x_j \in \mathcal{X} \quad (8)$$

where $\mathbf{W}_c \in \mathbb{R}^{d_h \times d_s}$, and σ is a non-linear activation function, considering that the non-linear transformation in Eq.(8) can help project s_t and h_j in the same semantic space. Empirically, we also found that using the $\tanh$ non-linearity worked better than linear transformation, and we used that for the following experiments. When calculating the copy-mode score, we use the hidden states $\{\mathbf{h}_1, ..., \mathbf{h}_{T_S}\}$ to "represent" each of the word in the source sequence $\{x_1, ..., x_{T_S}\}$ since the bi-directional RNN encodes not only the content, but also the location information into the hidden states in $\mathbf{M}$. The location informaton is important for copying (see Section 3.4 for related discussion). Note that we sum the probabilities of all x_j equal to y_t in Eq. (6) considering that there may be multiple source symbols for decoding y_t. Naturally we let $p(y_t, \mathsf{c}|\cdot) = 0$ if y_t does not appear in the source sequence, and set $p(y_t, \mathsf{g}|\cdot) = 0$ when y_t only appears in the source.

3.3 State Update

CopyNet updates each decoding state $\mathbf{s}_t$ with the previous state $\mathbf{s}_{t-1}$, the previous symbol y_{t-1} and the context vector $\mathbf{c}_t$ following Eq. (2) for the generic attention-based Seq2Seq model. However, there is some minor changes in the $y_{t-1} \longrightarrow \mathbf{s}_t$ path for the copying mechanism. More specifically, y_{t-1} will be represented as $[\mathbf{e}(y_{t-1}); \zeta(y_{t-1})]^\top$, where $\mathbf{e}(y_{t-1})$ is the word embedding associated with y_{t-1}, while $\zeta(y_{t-1})$ is the weighted sum of hidden states in $\mathbf{M}$ corresponding to y_t

$$\zeta(y_{t-1}) = \sum_{\tau=1}^{T_S} \rho_{t\tau}\mathbf{h}_\tau$$
$$\rho_{t\tau} = \begin{cases} \frac{1}{K}p(x_\tau, \mathsf{c}|\mathbf{s}_{t-1}, \mathbf{M}), & x_\tau = y_{t-1} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where K is the normalization term which equals $\sum_{\tau':x_{\tau'}=y_{t-1}} p(x_{\tau'}, c|\mathbf{s}_{t-1}, \mathbf{M})$, considering there may exist multiple positions with y_{t-1} in the source sequence. In practice, $\rho_{t\tau}$ is often concentrated on one location among multiple appearances, indicating the prediction is closely bounded to the location of words.

In a sense $\zeta(y_{t-1})$ performs a type of read to $\mathbf{M}$ similar to the attentive read (resulting $\mathbf{c}_t$) with however higher precision. In the remainder of this paper, $\zeta(y_{t-1})$ will be referred to as *selective read*. $\zeta(y_{t-1})$ is specifically designed for the copy mode: with its pinpointing precision to the corresponding y_{t-1}, it naturally bears the location of y_{t-1} in the source sequence encoded in the hidden state. As will be discussed more in Section 3.4, this particular design potentially helps copy-mode in covering a consecutive sub-sequence of words. If y_{t-1} is not in the source, we let $\zeta(y_{t-1}) = \mathbf{0}$.

3.4 Hybrid Addressing of M

We hypothesize that CopyNet uses a hybrid strategy for fetching the content in $\mathbf{M}$, which combines both content-based and location-based addressing. Both addressing strategies are coordinated by the decoder RNN in managing the attentive read and selective read, as well as determining when to enter/quit the copy-mode.

Both the semantics of a word and its location in X will be encoded into the hidden states in $\mathbf{M}$ by a properly trained encoder RNN. Judging from our experiments, the attentive read of CopyNet is driven more by the semantics and language model, therefore capable of traveling more freely on $\mathbf{M}$, even across a long distance. On the other hand, once CopyNet enters the copy-mode, the selective read of $\mathbf{M}$ is often guided by the location information. As the result, the selective read often takes rigid move and tends to cover consecutive words, including UNKs. Unlike the explicit design for hybrid addressing in Neural Turing Machine (Graves et al., 2014; Kurach et al., 2015), CopyNet is more subtle: it provides the archi-

tecture that can facilitate some particular location-based addressing and lets the model figure out the details from the training data for specific tasks.

Location-based Addressing: With the location information in $\{\mathbf{h}_i\}$, the information flow

$$\zeta(y_{t-1}) \xrightarrow{\text{update}} \mathbf{s}_t \xrightarrow{\text{predict}} y_t \xrightarrow{\text{sel. read}} \zeta(y_t)$$

provides a simple way of "moving one step to the right" on X. More specifically, assuming the selective read $\zeta(y_{t-1})$ concentrates on the ℓ^{th} word in X, the state-update operation $\zeta(y_{t-1}) \xrightarrow{\text{update}} \mathbf{s}_t$ acts as "`location` $\leftarrow$ `location+1`", making $\mathbf{s}_t$ favor the $(\ell+1)^{th}$ word in X in the prediction $\mathbf{s}_t \xrightarrow{\text{predict}} y_t$ in copy-mode. This again leads to the selective read $\hat{h}_t \xrightarrow{\text{sel. read}} \zeta(y_t)$ for the state update of the next round.

Handling Out-of-Vocabulary Words Although it is hard to verify the exact addressing strategy as above directly, there is strong evidence from our empirical study. Most saliently, a properly trained COPYNET can copy a fairly long segment full of OOV words, despite the lack of semantic information in its $\mathbf{M}$ representation. This provides a natural way to extend the effective vocabulary to include all the words in the source. Although this change is small, it seems quite significant empirically in alleviating the OOV problem. Indeed, for many NLP applications (e.g., text summarization or spoken dialogue system), much of the OOV words on the target side, for example the proper nouns, are essentially the replicates of those on the source side.

4 Learning

Although the copying mechanism uses the "hard" operation to copy from the source and choose to paste them or generate symbols from the vocabulary, COPYNET is fully differentiable and can be optimized in an end-to-end fashion using back-propagation. Given the batches of the source and target sequence $\{X\}_N$ and $\{Y\}_N$, the objectives are to minimize the negative log-likelihood:

$$\mathcal{L} = -\frac{1}{N}\sum_{k=1}^{N}\sum_{t=1}^{T}\log\left[p(y_t^{(k)}|y_{<t}^{(k)}, X^{(k)})\right], \quad (10)$$

where we use superscripts to index the instances. Since the probabilistic model for observing any target word is a mixture of generate-mode and copy-mode, there is no need for any additional labels for modes. The network can learn to coordinate the two modes from data. More specifically, if one particular word $y_t^{(k)}$ can be found in the source sequence, the copy-mode will contribute to the mixture model, and the gradient will more or less encourage the copy-mode; otherwise, the copy-mode is discouraged due to the competition from the shared normalization term Z. In practice, in most cases one mode dominates.

5 Experiments

We report our empirical study of COPYNET on the following three tasks with different characteristics

1. A synthetic dataset on with simple patterns;
2. A real-world task on text summarization;
3. A dataset for simple single-turn dialogues.

5.1 Synthetic Dataset

Dataset: We first randomly generate transformation rules with 5$\sim$20 symbols and variables $\mathbf{x}$ & $\mathbf{y}$, e.g.

a b **x** c d **y** e f $\longrightarrow$ g h **x** m,

with {a b c d e f g h m} being regular symbols from a vocabulary of size 1,000. As shown in the table below, each rule can further produce a number of instances by replacing the variables with randomly generated subsequences (1$\sim$15 symbols) from the same vocabulary. We create five types of rules, including "$\mathbf{x} \rightarrow \emptyset$". The task is to learn to do the Seq2Seq transformation from the training instances. This dataset is designed to study the behavior of COPYNET on handling simple and rigid patterns. Since the strings to repeat are random, they can also be viewed as some extreme cases of rote memorization.

Rule-type	Examples (e.g. **x** = i h k, **y** = j c)
$\mathbf{x} \rightarrow \emptyset$	a b c d **x** e f $\rightarrow$ c d g
$\mathbf{x} \rightarrow \mathbf{x}$	a b c d **x** e f $\rightarrow$ c d **x** g
$\mathbf{x} \rightarrow \mathbf{x}\,\mathbf{x}$	a b c d **x** e f $\rightarrow$ **x** d **x** g
$\mathbf{x}\,\mathbf{y} \rightarrow \mathbf{x}$	a b **y** d **x** e f $\rightarrow$ **x** d i g
$\mathbf{x}\,\mathbf{y} \rightarrow \mathbf{x}\,\mathbf{y}$	a b **y** d **x** e f $\rightarrow$ **x** d **y** g

Experimental Setting: We select 200 artificial rules from the dataset, and for each rule 200 instances are generated, which will be split into training (50%) and testing (50%). We compare the accuracy of COPYNET and the RNN Encoder-Decoder with (i.e. RNNsearch) or without attention (denoted as Enc-Dec). For a fair comparison, we use bi-directional GRU for encoder and another GRU for decoder for all Seq2Seq models, with hidden layer size = 300 and word embedding dimension = 150. We use bin size = 10 in beam search for testing. The prediction is considered

Rule-type	x $\rightarrow \emptyset$	x $\rightarrow$ x	x $\rightarrow$ xx	xy $\rightarrow$ x	xy $\rightarrow$ xy
Enc-Dec	**100**	3.3	1.5	2.9	0.0
RNNSearch	99.0	69.4	22.3	40.7	2.6
COPYNET	97.3	**93.7**	**98.3**	**68.2**	**77.5**

Table 1: The test accuracy (%) on synthetic data.

correct only when the generated sequence is exactly the same as the given one.

It is clear from Table 1 that COPYNET significantly outperforms the other two on all rule-types except "x $\rightarrow \emptyset$", indicating that COPYNET can effectively learn the patterns with variables and accurately replicate rather long subsequence of symbols at the proper places.This is hard to Enc-Dec due to the difficulty of representing a long sequence with very high fidelity. This difficulty can be alleviated with the attention mechanism. However attention alone seems inadequate for handling the case where strict replication is needed.

A closer look (see Figure 3 for example) reveals that the decoder is dominated by copy-mode when moving into the subsequence to replicate, and switch to generate-mode after leaving this area, showing COPYNET can achieve a rather precise coordination of the two modes.

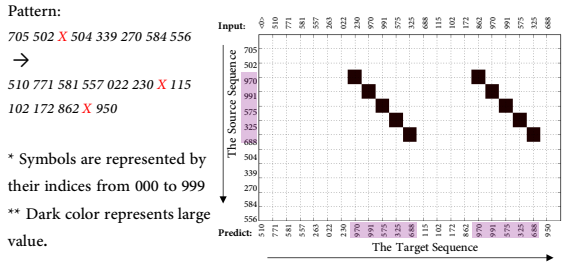


Figure 3: Example output of COPYNET on the synthetic dataset. The heatmap represents the activations of the copy-mode over the input sequence (left) during the decoding process (bottom).

5.2 Text Summarization

Automatic text summarization aims to find a condensed representation which can capture the core meaning of the original document. It has been recently formulated as a Seq2Seq learning problem in (Rush et al., 2015; Hu et al., 2015), which essentially gives *abstractive* summarization since the summary is generated based on a representation of the document. In contrast, *extractive* summarization extracts sentences or phrases from the original text to fuse them into the summaries, therefore making better use of the overall structure of the original document. In a sense, COPYNET for summarization lies somewhere between two categories, since part of output summary is actually extracted from the document (via the copying mechanism), which are fused together possibly with the words from the generate-mode.

Dataset: We evaluate our model on the recently published LCSTS dataset (Hu et al., 2015), a large scale dataset for short text summarization. The dataset is collected from the news medias on Sina Weibo[1] including pairs of (short news, summary) in Chinese. Shown in Table 2, PART II and III are manually rated for their quality from 1 to 5. Following the setting of (Hu et al., 2015) we use Part I as the training set and and the subset of Part III scored from 3 to 5 as the testing set.

Dataset	PART I	PART II	PART III
no. of pairs	2,400,591	10,666	1106
no. of score ≥ 3	-	8685	725

Table 2: Some statistics of the LCSTS dataset.

Experimental Setting: We try COPYNET that is based on character (+C) and word (+W). For the word-based variant the word-segmentation is obtained with jieba[2]. We set the vocabulary size to 3,000 (+C) and 10,000 (+W) respectively, which are much smaller than those for models in (Hu et al., 2015). For both variants we set the embedding dimension to 350 and the size of hidden layers to 500. Following (Hu et al., 2015), we evaluate the test performance with the commonly used ROUGE-1, ROUGE-2 and ROUGE-L (Lin, 2004), and compare it against the two models in (Hu et al., 2015), which are essentially canonical Encoder-Decoder and its variant with attention.

Models		ROUGE scores on LCSTS (%) R-1	R-2	R-L
RNN	+C	21.5	8.9	18.6
(Hu et al., 2015)	+W	17.7	8.5	15.8
RNN context	+C	29.9	17.4	27.2
(Hu et al., 2015)	+W	26.8	16.1	24.1
COPYNET	+C	**34.4**	**21.6**	**31.3**
	+W	**35.0**	**22.3**	**32.0**

Table 3: Testing performance of LCSTS, where "RNN" is canonical Enc-Dec, and "RNN context" its attentive variant.

It is clear from Table 3 that COPYNET beats the competitor models with big margin. Hu et al. (2015) reports that the performance of a word-based model is inferior to a character-based

[1] www.sina.com

[2] https://pypi.python.org/pypi/jieba

Input(1): 今天上午 9 点半， 复旦 投毒案 将 在 上海 二中院 公开审理 。 被害 学生 黄洋 的 亲属 已 从 四川 抵达 上海， 其父 称待 刑事 部分结束 后， 再 提 民事 赔偿， 黄洋 92 岁 的 奶奶 依然 不知情 。 今年 4 月， 在 复旦 上海医学院 读 研究生 的 黄洋 疑遭 室友 林森 浩 投毒， 不幸身亡 。 新民 网

Today 9:30, the Fudan poisoning case will be will on public trial at the Shanghai Second Intermediate Court. The relatives of the murdered student Huang Yang has arrived at Shanghai from Sichuan. His father said that they will start the lawsuit for civil compensation after the criminal section. HuangYang 92-year-old grandmother is still unaware of his death. In April, a graduate student at Fudan University Shanghai Medical College, Huang Yang is allegedly poisoned and killed by his roommate Lin Senhao. Reported by Xinmin

Golden: 林森 浩 投毒案 今日 开审 92 岁 奶奶 尚不知 情

the case of Lin Senhao poisoning is on trial today, his 92-year-old grandmother is still unaware of this

RNN context: 复旦投毒案： 黄洋疑遭室友投毒凶手已从四川飞往上海，父亲命案另有4人被通知家属不治？

CopyNet: 复旦 投毒案 今 在 沪 上 公开审理

the Fudan poisoning case is on public trial today in Shanghai

Input(2): 华谊 兄弟 （ 300027 ） 在 昨日 收盘 后 发布公告 称， 公司 拟 以 自有 资金 3.97 8亿元 收购 浙江 永乐 影视 股份 有限公司 若干 股东 持有 的 永乐 影视 51 % 的 股权 。 对于 此项 收购， 华谊 兄弟 董秘 胡明 昨日 表示： “ 和 永乐 影视 的 合并 是 对 华谊 兄弟 电视剧 业务 的 一个 加强 。

Huayi Brothers (300027) announced that the company intends to buy with its own fund 397.8 million 51% of Zhejiang Yongle Film LTD's stake owned by a number of shareholders of Yongle Film LTD. For this acquisition, the secretary of the board, Hu Ming, said yesterday: "the merging with Yongle Film is to strengthen Huayi Brothers on TV business".

Golden: 华谊 兄弟 拟 收购 永乐 影视 5 1 % 股权

Huayi Brothers intends to acquire 51% stake of Zhejiang Yongle Film

RNN context: 华谊兄弟收购永乐影视51%股权：与永乐影视合并为“和唐”影视合并的“UNK”和“UNK”的区别？

CopyNet: 华谊 兄弟 拟 3.978 亿 收购 永乐 影视 董秘 称 加强 电视剧 业务

Huayi Brothers is intended to 397.8 million acquisition of Yongle Film secretaries called to strengthen the TV business

Input(3): 工厂， 大门 紧锁， 约 20 名 工人 散 坐在 树荫下 。 “ 我们 就是 普通工人， 在 这里 等 工资 。 ” 其中 一人 说道 。 7 月 4 日 上午， 记者 抵达 深圳 龙华区 清 湖 路上 的 深圳 愿景 光电子 有限公司 。 正如 传言 一般， 愿景 光电子 倒闭 了， 大 股东 邢毅 不知 所踪 。

The door of factory is locked. About 20 workers are scattered to sit under the shade. "We are ordinary workers, waiting for our salary" one of them said. In the morning of July 4th, reporters arrived at Yuanjing Photoelectron Corporation located at Qinghu Road, Longhua District, Shenzhen. Just as the rumor, Yuanjing Photoelectron Corporation is closed down and the big shareholder Xing Yi is missing.

Golden: 深圳 亿元 级 LED 企业倒闭 烈日 下 工人 苦 等 老板

Hundred-million CNY worth LED enterprise is closed down and workers wait for the boss under the scorching sun

RNN context: 深圳 “<UNK>”：深圳<UNK><UNK>，<UNK>，<UNK>，<UNK>

CopyNet: 愿景 光电子 倒闭 20 名 工人 散 坐在 树荫下

Yuanjing Photoelectron Corporation is closed down, 20 workers are scattered to sit under the shade

Input(4): 截至 2012 年 10 月底， 全国 累计 报告 艾滋病 病毒感染者 和 病人 492191 例 。 卫生部 称， 性传播 已 成为 艾滋病 的 主要 传播 途径 。 至 2011 年 9 月， 艾滋病 感染者 和 病人数 累计 报告 数排 在 前 6 位 的 省份 依次 为 云南 、 广西 、 河南 、 四川 、 新疆 和 广东， 占 全国 的 75.8 %。 。

At the end of October 2012, the national total of reported HIV infected people and AIDS patients is 492,191 cases. The Health Ministry saids exual transmission has become the main route of transmission of AIDS. To September 2011, the six provinces with the most reported HIV infected people and AIDS patients were Yunnan, Guangxi, Henan,Sichuan, Xinjiang and Guangdong, accounting for 75.8% of the country.

Golden: 卫生部： 性传播 成 艾滋病 主要 传播 途径

Ministry of Health: Sexually transmission became the main route of transmission of AIDS

RNN context: 全国累计报告艾滋病患者和病人<UNK>例艾滋病患者占全国<UNK>%，性传播成艾滋病高发人群 ？

CopyNet: 卫生部： 性传播 已 成为 艾滋病 主要 传播 途径

Ministry of Health: Sexually transmission has become the main route of transmission of AIDS

Input(5): 中国 反垄断 调查 风暴 继续 席卷 汽车行业， 继 德国 车企 奥迪 和 美国 车企 克莱斯勒 “ 沦陷 ” 之后， 又 有 12 家 日本 汽车 企业 卷入漩涡 。 记者 从 业内人士 获悉， 丰田 旗下 的 雷克萨斯 近期 曾 被 发改委 约 谈 。

Chinese antitrust investigation continues to sweep the automotive industry. After Germany Audi car and the US Chrysler "fell", there are 12 Japanese car companies involved in the whirlpool. Reporters learned from the insiders that Toyota's Lexus has been asked to report to the Development and Reform Commission recently.

Golden: 发改委 公布 汽车 反垄断 进程： 丰田 雷克萨斯 近期 被 约 谈

the investigation by Development and Reform Commission: Toyota's Lexus has been asked to report

RNN context: 丰田雷克萨斯遭发改委约谈：曾被约谈丰田旗下的雷克萨斯遭发改委约谈负人被约谈

CopyNet: 中国 反垄断 继续 席卷 汽车行业 12 家 日本 汽车 企业 被 发改委 约 谈

Chinese antitrust investigation continues to sweep the automotive industry. 12 Japanese car companies are asked to report to he Development and Reform Commission

Input(6): 镁离子电池相比锂电池能量密度提升了近一倍，这意味着使用了镁电池的电动车，纯电续航也将有质的提升。但目前由于电解质等技术壁垒，要大规模量产并取代锂电池还为时过早。

The energy density of Magnesium ion batteries almost doubles that of lithium battery, which means that for the electric vehicles using of magnesium batteries will last longer even at pure electric power. But currently due to the technical barriers to the electrolyte, it is still too early for the mass production of it and replacing lithium batteries..

Golden: 锂电池或将被淘汰 能量密度更高的镁电池亦大势所趋

Lithium batteries will be phased out, magnesium battery with energy density higher will be the future trend

RNN context: <UNK>、<UNK>、<UNK>、<UNK>、<UNK>、<UNK>、<UNK>、<UNK>、<UNK>、<UNK>、<UNK>、<UNK>电 池 了

CopyNet: 镁离子电池问世：大规模量产取代锂电池

Magnesium ion battery is developed : mass production of it will replace lithium batteries

Input(7): 1 . 掌握 技巧 融会贯通 ； 2 . 学会 融资 ； 3 . 懂 法律 ； 4 . 保持 自信 ； 5 . 测试 + 尝试 ； 6 . 了解 客户 的 需求 ； 7 . 预测 + 衡量 + 确保 ； 8 . 做好 与 各种 小 bug 做 斗争 的 心态 ； 9 . 发现 机遇 保持 创业 激情 。

1 . master the skills; 2 Learn to finance ; 3 . understand the law; 4. Be confident; 5 . test+ trial; 6 . understand the need of customers; 7 forecast + measure + ensure; 8. mentally prepared to fight all kinds of small bugs ; 9 discover opportunities and keep the passion of start-up.

Golden: 初次 创业者 必知 的 10 个 技巧

The 10 tips for the first time start-ups

RNN context: 6个方法让你创业的6个<UNK>与<UNK>，你怎么看懂你的创业故事吗？（6家）

CopyNet: 创业 成功 的 9 个 技巧

The 9 tips for success in start-up

Input(8): 9 月 3 日， 总部 位于 日内瓦 的 世界 经济 论坛 发布 了 《 2014 - 2015 年 全球 竞争力 报告 》， 瑞士 连续 六年 位居 榜首， 成为 全球 最具 竞争力 的 国家， 新加坡 和 美国 分列 第二 位 和 第三位 。 中国 排名第 28 位， 在 金砖 国家 中 排名 最高 。

On September 3, the Geneva based World Economic Forum released " The Global Competitiveness Report 2014-2015". Switzerland topped the list for six consecutive years , becoming the world's most competitive country. Singapore and the United States are in the second and third place respectively. China is in the 28th place, ranking highest among the BRIC countries.

Golden: 全球 竞争力 排行榜 中国 居 28 位居 金砖 国家 首位

The Global competitiveness ranking list, China is in the 28th place, the highest among BRIC countries.

RNN context: 2014-2015年全球竞争力报告：瑞士连续6年居榜首中国居28位(首/3———访榜首)中国排名第28位

CopyNet: 2014 - 2015 年 全球 竞争力 报告： 瑞士 居首 中国 第 28

2014--2015 Global Competitiveness Report: Switzerland topped and China the 28th

Figure 4: Examples of COPYNET on LCSTS compared with RNN context. Word segmentation is applied on the input, where OOV words are underlined. The highlighted words (with different colors) are those words with copy-mode probability higher than the generate-mode. We also provide literal English translation for the document, the golden, and COPYNET, while omitting that for RNN context since the language is broken.

one. One possible explanation is that a word-based model, even with a much larger vocabulary (50,000 words in Hu et al. (2015)), still has a large proportion of OOVs due to the large number of entity names in the summary data and the mistakes in word segmentation. COPYNET, with its ability to handle the OOV words with the copying mechanism, performs however slightly better with the word-based variant.

5.2.1 Case Study

As shown in Figure 4, we make the following interesting observations about the summary from COPYNET: 1) most words are from copy-mode, but the summary is usually still fluent; 2) COPYNET tends to cover consecutive words in the original document, but it often puts together segments far away from each other, indicating a sophisticated coordination of content-based addressing and location-based addressing; 3) COPYNET handles OOV words really well: it can generate acceptable summary for document with many OOVs, and even the summary itself often contains many OOV words. In contrast, the canonical RNN-based approaches often fail in such cases.

It is quite intriguing that COPYNET can often find important parts of the document, a behavior with the characteristics of extractive summarization, while it often generate words to "connect" those words, showing its aspect of abstractive summarization.

5.3 Single-turn Dialogue

In this experiment we follow the work on neural dialogue model proposed in (Shang et al., 2015; Vinyals and Le, 2015; Sordoni et al., 2015), and test COPYNET on single-turn dialogue. Basically, the neural model learns to generate a response to user's input, from the given (input, response) pairs as training instances.

Dataset: We build a simple dialogue dataset based on the following three instructions:

1. Dialogue instances are collected from Baidu Tieba[3] with some coverage of conversations of real life e.g., greeting and sports, etc.
2. Patterns with slots like

   `hi, my name is` **x** $\rightarrow$ `hi,` **x**

   are mined from the set, with possibly multiple responding patterns to one input.
3. Similar with the synthetic dataset, we enlarge the dataset by filling the slots with suitable subsequence (e.g. name entities, dates, etc.)

To make the dataset close to the real conversations, we also maintain a certain proportion of instances with the response that 1) do not contain entities or 2) contain entities not in the input.

Experimental Setting: We create two datasets: DS-I and DS-II with slot filling on 173 collected patterns. The main difference between the two datasets is that the filled substrings for training and testing in DS-II have no overlaps, while in DS-I they are sampled from the same pool. For each dataset we use 6,500 instances for training and 1,500 for testing. We compare COPYNET with canonical RNNSearch, both character-based, with the same model configuration in Section 5.1.

	DS-I (%)		DS-II (%)	
Models	Top1	Top10	Top1	Top10
RNNSearch	44.1	57.7	13.5	15.9
COPYNET	**61.2**	**71.0**	**50.5**	**64.8**

Table 4: The decoding accuracy on the two testing sets. Decoding is admitted success only when the answer is found exactly in the Top-K outputs.

We compare COPYNET and RNNSearch on DS-I and DS-II in terms of top-1 and top-10 accuracy (shown in Table 4), estimating respectively the chance of the top-1 or one of top-10 (from beam search) matching the golden. Since there are often many good responses to an input, top-10 accuracy appears to be closer to the real world setting.

As shown in Table 4, COPYNET significantly outperforms RNNsearch, especially on DS-II. It suggests that introducing the copying mechanism helps the dialogue system master the patterns in dialogue and correctly identify the correct parts of input, often proper nouns, to replicate in the response. Since the filled substrings have no overlaps in DS-II, the performance of RNNSearch drops significantly as it cannot handle words unseen in training data. In contrast, the performance of COPYNET only drops slightly as it has learned to fill the slots with the copying mechanism and relies less on the representation of the words.

5.3.1 Case Study

As indicated by the examples in Figure 5, COPYNET accurately replicates the critical segments from the input with the copy-mode, and generates

[3] http://tieba.baidu.com

Figure 5: Examples from the testing set of DS-II shown as the input text and golden, with the outputs of RNNSearch and CopyNet. Words in red rectangles are unseen in the training set. The highlighted words (with different colors) are those words with copy-mode probability higher than the generate-mode. Green cirles (meaning correct) and red cross (meaning incorrect) are given based on human judgment on whether the response is appropriate.

the rest of the answers smoothly by the generate-mode. Note that in (2) and (3), the decoding sequence is not exactly the same with the standard one, yet still correct regarding to their meanings. In contrast, although RNNSearch usually generates answers in the right formats, it fails to catch the critical entities in all three cases because of the difficulty brought by the unseen words.

6 Related Work

Our work is partially inspired by the recent work of Pointer Networks (Vinyals et al., 2015a), in which a pointer mechanism (quite similar with the proposed copying mechanism) is used to predict the output sequence directly from the input. In addition to the difference with ours in application, (Vinyals et al., 2015a) cannot predict outside of the set of input sequence, while COPYNET can naturally combine generating and copying.

COPYNET is also related to the effort to solve the OOV problem in neural machine translation. Luong et al. (2015) introduced a heuristics to post-process the translated sentence using annotations on the source sentence. In contrast COPYNET addresses the OOV problem in a more systemic way with an end-to-end model. However, as COPYNET copies the exact source words as the output, it cannot be directly applied to machine translation. However, such copying mechanism can be naturally extended to any types of references except for the input sequence, which will help in applications with heterogeneous source and target sequences such as machine translation.

The copying mechanism can also be viewed as carrying information over to the next stage without any nonlinear transformation. Similar ideas are proposed for training very deep neural networks in (Srivastava et al., 2015; He et al., 2015) for classification tasks, where shortcuts are built between layers for the direct carrying of information.

Recently, we noticed some parallel efforts towards modeling mechanisms similar to or related to copying. Cheng and Lapata (2016) devised a neural summarization model with the ability to extract words/sentences from the source. Gulcehre et al. (2016) proposed a pointing method to handle the OOV words for summarization and MT. In contrast, COPYNET is more general, and not limited to a specific task or OOV words. Moreover, the softmaxCOPYNET is more flexible than gating in the related work in handling the mixture of two modes, due to its ability to adequately model the content of copied segment.

7 Conclusion and Future Work

We proposed COPYNET to incorporate copying into the sequence-to-sequence learning framework. For future work, we will extend this idea to the task where the source and target are in heterogeneous types, for example, machine translation.

Acknowledgments

This work is supported in part by the China National 973 Project 2014CB340301.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jianpeng Cheng and Mirella Lapata. 2016. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

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

Baotian Hu, Qingcai Chen, and Fangze Zhu. 2015. Lcsts: a large scale chinese short text summarization dataset. *arXiv preprint arXiv:1506.05865*.

Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. 2015. Neural random-access machines. *arXiv preprint arXiv:1511.06392*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In Stan Szpakowicz Marie-Francine Moens, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July. Association for Computational Linguistics.

Geoffrey J McLachlan and Kaye E Basford. 1988. Mixture models. inference and applications to clustering. *Statistics: Textbooks and Monographs, New York: Dekker, 1988*, 1.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*.

Shiqi Shen, Yong Cheng, Zhongjun He, Wei He, Hua Wu, Maosong Sun, and Yang Liu. 2015. Minimum risk training for neural machine translation. *CoRR*, abs/1512.02433.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *arXiv preprint arXiv:1505.00387*.

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

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2674–2682.

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

Cross-domain Text Classification with Multiple Domains and Disparate Label Sets

Himanshu S. Bhatt, Manjira Sinha and Shourya Roy
Xerox Research Centre India, Bangalore, INDIA
{Firstname.Lastname}@Xerox.Com

Abstract

Advances in transfer learning have let go the limitations of traditional supervised machine learning algorithms for being dependent on annotated training data for training new models for every new domain. However, several applications encounter scenarios where models need to transfer/adapt across domains when the label sets vary both in terms of count of labels as well as their connotations. This paper presents first-of-its-kind transfer learning algorithm for cross-domain classification with multiple source domains and disparate label sets. It starts with identifying transferable knowledge from across multiple domains that can be useful for learning the target domain task. This knowledge in the form of selective labeled instances from different domains is congregated to form an auxiliary training set which is used for learning the target domain task. Experimental results validate the efficacy of the proposed algorithm against strong baselines on a real world social media and the 20 Newsgroups datasets.

1 Introduction

A fundamental assumption in supervised statistical learning is that training and test data are independently and identically distributed (i.i.d.) samples drawn from a distribution. Otherwise, good performance on test data cannot be guaranteed even if the training error is low. On the other hand, transfer learning techniques allow domains, tasks, and distributions used in training and testing to be different, but related. It works in contrast to traditional supervised techniques on the principle of transferring learned knowledge across domains. Pan and Yang, in their survey paper (2010), described different transfer learning settings depending on if domains and tasks vary as well as labeled data is available in one/more/none of the domains. In this paper, we propose a generic solution for multi-source transfer learning where domains and tasks are different and no labeled data is available in the target domain. This is a relatively less chartered territory and arguably a more generic setting of transfer learning.

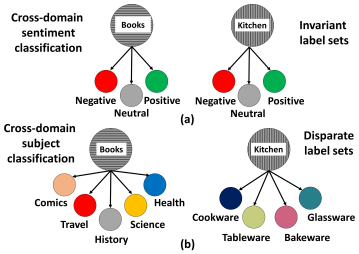


Figure 1: Cross-domain (a) sentiment classification and (b) subject classification. Illustrates (a) invariant and (b) disparate label sets.

Motivating example: Consider a social media consulting company helping brands to monitor their social media channels. Two problems typically of interest are: (i) sentiment classification (*is a post positive/negative/neutral?*) and (ii) subject classification (*what was the subject of a post?*). While sentiment classification attempts to classify a post based on its polarity, subject classification is towards identifying the subject (or topic) of the post, as illustrated in Figure 1. The company has been using standard classification techniques from an off-the-shelf machine learning toolbox. While machine learning toolkit helps them to create and apply statistical models efficiently, the same model can not be applied on a new collection due to variations in data distributions across collections[1]. It requires a few hundreds of manually labeled posts for every task on every collec-

[1]A collection comprises comments/posts pertaining to a particular client/product/services. Domain and collection are used interchangeably.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1641–1650,
Berlin, Germany, August 7-12, 2016.

tion. As social media are extremely high velocity and low retention channels, human labeling efforts act like that proverbial narrow bottleneck. Need of the hour was to reduce, if not eliminate, the human-intensive labeling stage while continue to use machine learning models for new collections.

Several transfer learning techniques exist in the literature which can reduce labeling efforts required for performing tasks in new collections. Tasks such as sentiment classification, named entity recognition (NER), part of speech (POS) tagging that have invariant label sets across domains, have shown to be greatly benefited from these works. On the other hand, tasks like subject classification that have disparate label sets across domains have not been able to gain at pace with the advances in transfer learning. Towards that we formulate the problem of *Cross-domain classification with disparate label sets* as learning an accurate model for the new unlabeled target domain given labeled data from multiple source domains where all domains have (possibly) different label sets.

Our contributions: To the best of our knowledge, this is the first work to explore the problem of cross-domain text classification with multiple source domains and disparate label sets. The other contributions of this work includes a simple yet efficient algorithm which starts with identifying transferable knowledge from across multiple source domains useful for learning the target domain task. Specifically, it identifies relevant class-labels from the source domains such that the instances in those classes can induce class-separability in the target domain. This transferable knowledge is accumulated as an auxiliary training set for an algorithm to learn the target domain classification task followed by suitable transformation of the auxiliary training instances.

Organization of the paper is as follows: Section 2 presents the preliminaries and notation, Section 3 summarizes the related work. Section 4 and 5 present the proposed algorithm and experimental results respectively. Section 6 concludes the paper.

2 Preliminaries and Notations

A domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ is characterized by two components: a feature space $\mathcal{X}$ and a marginal probability distribution $P(X)$, where $X = \{x_1, x_2, ...x_n\} \in \mathcal{X}$. A task $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$ also consists of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$.

In our settings for cross-domain classification with disparate label sets, we assume M source domains, denoted as $\mathcal{D}_{S_i}$, where $i = \{1, 2, ..M\}$. Each source domain has different marginal distribution i.e. $P(X_{S_i}) \neq P(X_{S_j})$ and different label space i.e. $\mathcal{Y}_{S_i} \neq \mathcal{Y}_{S_j}, \forall i, j \in M$. The label space across domains vary both in terms of count of class-labels as well as their connotations; however, a finite set of labeled instances are available from each source domain. The target domain ($\mathcal{D}_T$) consists of a finite set of unlabeled instances, denoted as t_i where $i = \{1, .., N\}$. Let $\mathcal{Y}_T$ be the target domain label space with K class-labels. We assume that the number of classes in the target domain i.e. K is known (analogous to clustering where the number of clusters is given).

3 Related Work

Table 1 summarizes different settings of transfer learning (Pan and Yang, 2010) and how this work differentiates from the existing literature[2]. The first scenario represents the ideal settings of traditional machine learning (Mitchell, 1997) where a model is trained on a fraction of labeled data and performs well for the same task on the future unseen instances from the same domain.

The second scenario where the domains vary while the tasks remain the same is referred to as transductive transfer learning. This is the most extensively studied settings in the transfer learning literature and can be broadly categorized as single and multi-source adaptation. Single source adaptation (Chen et al., 2009; Ando and Zhang, 2005; Daumé III, 2009) primarily aims at minimizing the divergence between the source and target domains either at instance or feature levels. The general idea being identifying a suitable low dimensional space where transformed source and target domains data follow similar distributions and hence, a standard supervised learning algorithm can be trained (Daumé III, 2009; Jiang and Zhai, 2007; Pan et al., 2011; Blitzer et al., 2007; Pan et al., 2010; Dai et al., 2007; Bhatt et al., 2015).

While several existing single source adaptation techniques can be extended to multi-source adaptation, the literature in multi-source adaptation can be broadly categorized as: 1) feature representation approaches (Chattopadhyay et al., 2012; Sun et al., 2011; Duan et al., 2009; Duan et al., 2012;

[2]This is not the complete view of the transfer learning literature; however, covers relevant work that helps motivate/differentiate the novel features of this paper.

Table 1: Summarizing the related work and differentiating the novel features of the proposed algorithm.

Scenario	Settings	Nature of Data	Learning Paradigm	Main Concepts	Our Differentiation
$\mathcal{D}_S = \mathcal{D}_T$, $\mathcal{T}_S = \mathcal{T}_T$	Traditional Machine learning	Labelled data in source domain(s) and unlabeled data in target domain	Source and target domains are exactly the same	Learn models on training set and test on future unseen data	Allows tasks across domains to be different;a more general setting
$\mathcal{D}_S \neq \mathcal{D}_T$, $\mathcal{T}_S = \mathcal{T}_T$	Transductive Transfer Learning	Labelled data in source domain(s) and unlabeled data from the target domain $\mathcal{P}(\mathcal{X}_S) \neq \mathcal{P}(\mathcal{X}_T)$	Single source domain adaptation	Learning common shared representation; instance weighing, parameter transfer	Exploits multiple sources each with disparate label sets.
			Multi-source adaptation	Classifier combination; efficient combination of information from multiple sources; Feature representation	Intelligent selection of transferable knowledge from multiple sources for adaptation.
No conditions on $\mathcal{D}_S$ & $\mathcal{D}_T$, but, $\mathcal{T}_S \neq \mathcal{T}_T$	Inductive Transfer Learning	Unlabeled data in source domain(s) and labeled data in target domain	Self-taught learning	Extracts higher level representations from unlabeled auxiliary data to learn instance-to-label mapping with labeled target instances	Learns instance-to-label mapping in the unlabeled target domain using multiple labeled source domains having different data distributions and label spaces.
		Labeled data is available in all domains	Multi-task learning	Simultaneously learns multiple tasks within (or across) domain(s) by exploiting the common feature subspace shared across the tasks	Learns the optimal class distribution in an unlabeled target domain by minimizing the differences with multiple labeled source domains.
$\mathcal{D}_S \neq \mathcal{D}_T$, $\mathcal{T}_S \neq \mathcal{T}_T$	Kim *et al.* (2015)	Labeled data in source and target domains	Transfer learning with disparate label set	Disparate fine grained label sets across domains, however, same coarse grained labels set can be invoked across domains	No coarse-to-fine label mapping due to heterogeneity of label sets, Assumes no labelled data in target domain.

Bollegala et al., 2013; Crammer et al., 2008; Mansour et al., 2009; Ben-David et al., 2010; Bhatt et al., 2016) and 2) combining pre-trained classifiers (Schweikert and Widmer, 2008; Sun and Shi, 2013; Yang et al., 2007; Xu and Sun, 2012; Sun et al., 2013). Our work differentiates in intelligently exploiting selective transferable knowledge from multiple sources unlike existing approaches where multiple sources contribute in a brute-force manner.

The third scenario where the tasks differ irrespective of the relationship among domains is referred to as inductive transfer learning. Self-taught learning (Raina et al., 2007) and multi-task (Jiang, 2009; Maurer et al., 2012; Xu et al., 2015; Kumar and Daume III, 2012) learning are the two main learning paradigms in this scenario and Table 1 differentiates our work from these.

This work closely relates to the fourth scenario where we allow domains to vary in the marginal probability distributions and the tasks to vary due to different label spaces[3]. The closest prior work by Kim *et al.* (2015) address a sequential labeling problem in NLU where the fine grained label sets across domains differ. However, they assume that there exists a bijective mapping between the coarse and fine-grained label sets across domains. They learn this mapping using labeled instances from the target domain to reduce the problem to a standard domain adaptation problem (Scenario 2).

[3]This work do not consider scenario when domains vary in feature spaces and tasks vary in the objective predictive functions.

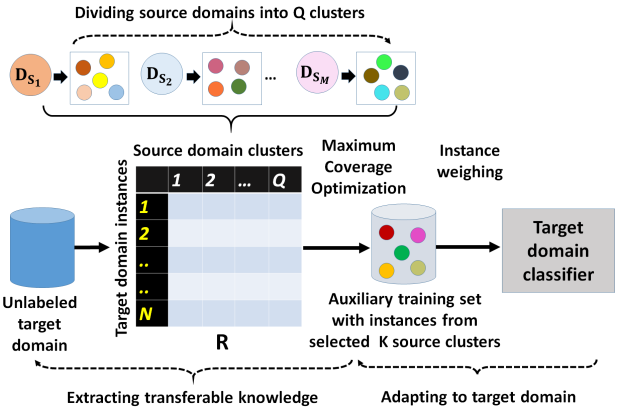


Figure 2: Illustrates different stages of the proposed algorithm.

However, this paper caters to multiple source domains with disparate label sets without assuming availability of any labeled data from the target domain or fine-to-coarse label mappings across domains.

4 Cross-domain Classification with Disparate Label Set

The underlying philosophy of the proposed algorithm is to learn the target domain task by using the available information from multiple source domains. To accomplish this, we have developed an algorithm to identify and extract partial transferable knowledge from multiple sources. This knowledge is then suitably transformed to induce classes in the target domain using the class separation from the source domains. Different stages of the proposed algorithm, as shown in Figure 2, are

elaborated in the next sections.

4.1 Exploiting Multiple Domains

If we had the mappings between the source and target domain label sets, we could have leveraged existing transfer learning approaches. However, heterogeneity of label sets across domains and the unlabeled data from the target domain exacerbate the problem. Our objective is to leverage the knowledge from multiple source domains to induce class-separability in the target domain. Inducing class-separability refers to segregating the target domain into K classes using labeled instances from selective K source domain classes.

Towards this, the proposed algorithm divides each source domain into clusters/groups based on the class-labels such that instances with the same label are grouped in one cluster. All source domains are divided into Q clusters where $Q = \sum_{m=1}^{M} ||Y_m||$ represents the count of class-labels across all sources. $||Y_m||$ being the count of class-labels in the m^{th} source domain. C_q denotes the q^{th} cluster and μ_q denotes its centroid computed as the average of all the members in the cluster. We assert that the target domain instances that have high similarity to a particular source domain cluster can be grouped together. Given N target domain instances and Q source domain clusters, a matrix R (dimension $N \times Q$) is computed based on the similarity of the target domain instances with the source clusters. The i^{th} row of the matrix captures the similarity of the i^{th} target domain instance (t_i) with all the source domain clusters. It captures how different source domain class-labels are associated with the target domain instances and hence, can induce class-separability in the target domain.

4.2 Extracting Transferable Knowledge

The similarity matrix R associates target domain instances to the source domain clusters in proportion to their similarity. However, the objective is to select the optimal K source domain clusters that fit the maximum number of target domain instances. This problem is similar to the well-known combinatorial optimization problem of Maximum Coverage (Vazirani, 2003) where given a collection of P sets, we need to select A sets ($A < P$) such that the size of the union of the selected sets is maximized. In this paper, we are given Q source domain clusters and need to select K clusters such that the corresponding number of associated target domain instances is maximized. As the Maximum Coverage problem is **NP**-hard, we implement a greedy algorithm for selecting the k source domain clusters, as illustrated in Algorithm 1.

Algorithm 1 Selecting K Source Clusters

Input: A matrix R, K = number target domain classes, l= number of selected cluster.
Initialize: $l = 0$, Normalize R such that each row sums up to 1.
repeat:
1: Pick the column in R which has maximum sum of similarity scores for uncovered target domain instances.
2: Mark elements in the chosen column as covered.
3: $l = l + 1$
until: $l = K$
Output: K source domain clusters.

A source domain contributes partially in terms of zero or more class-labels (clusters) identified using the Algorithm 1. Therefore, we refer to the labeled instances from the selected clusters of a source domain as the partial transferable knowledge from that domain. This partial transferable knowledge from across multiple source domains is congregated to form an auxiliary training set, referred to as (AUX).

4.3 Adapting to the Target Domain

The auxiliary training set comprises labeled instances from selected K source domain clusters[4]. Since, the auxiliary set is pulled out from multiple source domains, it follows different data distribution as compared to the target domain. For a classifier, trained on the K-class auxiliary training set, the distributional variations have to be normalized so that it can generalize well on the target domain.

In this research, we proposed to use an instance weighting technique (Jiang and Zhai, 2007) to minimize the distributional variations by deferentially weighting instances in the auxiliary set. Intuitively, the auxiliary training instances similar to the target domain are assigned higher weights while training the classifier and vice versa. The weight for the i^{th} instance in the auxiliary set should be proportional to the ratio $\frac{(P_t(x_i))}{(P_a(x_i))}$. However, since the actual probability distributions

[4]The K classes in auxiliary set induce class-separability in the target domain, however, the actual class-labels across these two may not have any sort of coarse-to-fine mapping.

Algorithm 2 Cross-domain Classification with Disparate Label Sets

Input: M source domains, target domain instances (t_i), $i = (1, ..., N)$, K = number of target domain classes.

Process: Divide M sources into Q clusters *s.t.* $Q = \sum_{q=1}^{M} |Y_m|$. C_q be the q^{th} cluster & μ_q be its centroid computed as shown in Eq 1.

A: Exploiting Multiple Sources:
for $i = 0$: till N **do**
  for $q = 0$: till Q **do**
    $R[i, q] = Sim(\mu_q, t_i)$
  end for
end for

B: Extracting partial knowledge:
1: Pick K columns from R using Algorithm 1.
2: Construct AUX by congregating instances from the selected K source domain class-labels.

C: Adapting to target domain:
1: Minimize distributional variations using instance weighing technique.
2: Train a K-class classifier using AUX.

Output: K-class target domain classifier.

($P_a(x)$ and $P_t(x)$ for the auxiliary set and target domain respectively) are unknown, the instance difference is approximated as $\frac{(P_t(x_i|d=target))}{(P_a(x_i|d=auxilliary))}$, where d is a random variable used to represent whether x_i came from the auxiliary set or the target domain. To calculate this ratio, a binary classifier is trained using the auxiliary set and target domain data with labels {-1} and {+1} respectively. The predicted probabilities from the classifier are used to estimate the ratio as the weight for the i^{th} auxiliary instance x_i. Finally, a K-class classifier is trained on the weighted auxiliary training set to perform classification on the target domain data.

4.4 Algorithm

As shown in Figure 2, the step-by-step flow of the proposed algorithm is summarized below:

1. Divide M source domains into Q clusters, each represented as C_q, $q = \{1, 2, .., Q\}$.

2. Compute centroid of each cluster as the average of the cluster members, as shown in Eq. 1.

$$\mu_q = \frac{1}{||C_q||} \sum_{(i=1;\mathbf{x}_i \in C_q)}^{||C_q||} \mathbf{x}_i \qquad (1)$$

where μ_q is the centroid, $||C_q||$ is the membership count and $\mathbf{x}_i$ is the i^{th} member of C_q.

3. For target instances t_i $\forall i \in N$, compute cosine similarity with all the source domain cluster centroids to form the matrix R (dimensions: $N \times Q$), as shown in Eq. 2

$$R[i, q] = Sim(\mu_q, \mathbf{t}_i) = \frac{\mu_q \cdot \mathbf{t}_i}{||\mu_q||\ ||\mathbf{t}_i||} \qquad (2)$$

4. Run Algorithm 1 on R to select K optimal source clusters (i.e. columns of R).

5. Congregate labeled instances from the selected source domain clusters to form the K-class auxiliary training set.

6. Minimize the divergence between the auxiliary set and target domain using the instance weighing technique, described in Section 4.3.

7. Finally, train a K-class classifier on deferentially weighted auxiliary training instances to perform classification in the target domain.

The K-class classifier trained on the auxiliary training set is an SVM classifier (Chih-Wei Hsu and Lin, 2003) with $L2 - loss$ from the LIBLINEAR library (Fan et al., 2008). The classifier used in the instance weighing technique is again an SVM classifier with RBF kernel. The proposed algorithm uses distributional embedding i.e. Doc2Vec (Le and Mikolov, 2014) to represent instances from the multiple source and target domains. We used an open-source implementation of Doc2Vec (Le and Mikolov, 2014) for learning 400 dimensional vector representation using DBoW.

5 Experimental Evaluation

Comprehensive experiments are performed to evaluate the efficacy of the proposed algorithm for cross-domain classification with disparate label sets across domains on two datasets.

5.1 Datasets

The first dataset is a real-world Online Social Media (OSM) dataset which consists of 74 collections. Each collection comprises comments/tweets that are collected based on user-defined keywords. These keywords are fed to a listening engine which crawls the social media (i.e. Twitter.com) and fetches comments matching the keywords. The task is to classify the comments in a collection

Table 2: Illustrates variability in label sets across some collections from the OSM dataset.

Apple iPhone 6	Apple iOS 8	Apple iPad mini3
Camera	Locking apps & features	Release date & Features
Design	Extensibility features	Apple play & NFC
Review link	General features related marketing	Apple sim card
Apple Play/NFC	Camera features	Touch ID
Comparison to Android	Password with touch integration	iPad mini3 - disappoints
Price	Health & fitness app	
Apple watch	Location & Maps	
	Firmware updates	

Table 3: Table illustrates the collections from the EMPATH database used in this research.

Collection ID	Domain	#Categories
Coll 1	Huwaei	5
Coll 2	Healthcare	9
Coll 3	Whattsapp	8
Coll 4	Apple iOS 8	8
Coll 5	Apple iPhone 6	7

into user-defined categories. These user-defined categories may vary across collections in terms of count as well as their connotations. Table 2 shows an example of the user-defined categories for a few collections related to "Apple" products. In the experiments, one collection is used as unlabeled target collection and the remaining collections are used as the labeled source collections. We randomly selected 5 target collections to report the performance, as described in Table 3.

The second dataset is the 20 Newsgroups (NG) (Lang, 1995) dataset which comprises $20,000$ news articles organized into 6 groups with different sub-groups both in terms of count as well as connotations, as shown in Figure 3(a). Two different experiments are performed on this dataset. In the first experiment ("**Exp-1**"), one group is considered as the target domain and the remaining 5 groups as the source domains. In the second experiment ("**Exp-2**"), one sub-group from each of the first five groups[5] is randomly selected to synthesize a target domain while all the groups (with the remaining sub-groups) are used as source domains. Figure 3(b) shows an example on how to synthesize target domains in "Exp-2". There are 720 possible target domains in this experiment and we report the average performance across all possible target domains, referred to as "Grp 7". The task in both the experiments is to categorize the target domain into its K categories (sub-groups) using labeled data from multiple source domains.

[5]Group-6 has only 1 sub-group, therefore, it is considered for synthesizing target domain in the experiments.

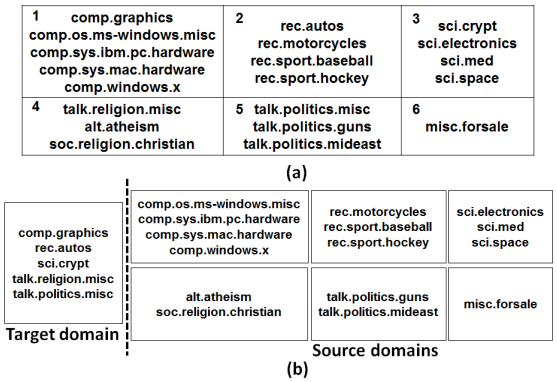


Figure 3: Illustrates (a) different groups (b) target domain synthesis ("EXP 2") on the NG dataset.

5.2 Evaluation Metric

The performance is reported in terms of classification accuracy on the target domain. There is no definite mapping between the actual class-labels in the target domain and the K categories (i.e. induced categories) in the auxiliary training set. Therefore, we sequentially evaluate all possible one-to-one mappings between the K categories in the auxiliary training set and target domain to report results for the best performing mapping.

5.3 Experimental Protocol

The performance of the proposed algorithm is skylined by the in-domain performance (**Gold**), i.e. a classifier trained and tested on the labeled target domain data. We also compared the performance with spherical K-means clustering (Dhillon and Modha, 2001) used to group the target domain data into K categories against the ground truth, referred to **CL**. Spherical K-means clustering is based on cosine similarity and performs better for high-dimensional sparse data such as text.

To compare with a baseline and an existing adaptation algorithm, we selected the most similar source domain[6] with exactly K number of class-labels and report the performance on the best possible mapping, as described in Section 5.2. To compute the baseline (**BL**), a classifier trained on the source domain is used to categorize the target domain. A widely used domain adaptation algorithm, namely structural correspondence learning (**SCL**) (Blitzer et al., 2007) is also applied using the selected source domain.

[6]The most similar source domain is selected using proxy-$\mathcal{A}$ distance (Blitzer et al., 2007) which has good correlation with domain adaptation performance.

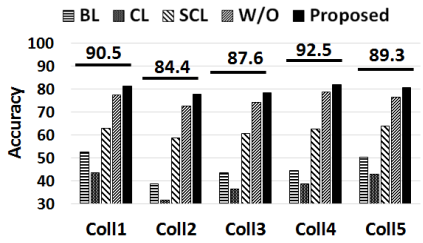


Figure 4: Compares the performance of different techniques on the OSM dataset.

Table 4: Summarizes the performance of the proposed algorithm on the OSM dataset.

Coll ID (#)	BL	CL	SCL	W/O	Proposed	Gold
Coll 1 (5)	52.6	43.7	62.8	77.4	81.4	90.5
Coll 2 (9)	38.6	31.8	58.8	72.5	77.6	84.4
Coll 3 (8)	43.6	36.4	60.7	74.2	78.5	87.6
Coll 4 (8)	44.7	38.8	62.5	78.8	82.1	92.5
Coll 5 (7)	50.5	42.8	64.4	76.6	80.5	89.3

5.4 Results and Analysis

Key observations and analysis from the experimental evaluations are summarized below:

5.4.1 Results on the OSM Dataset

Results in Figure 4 and Table 4 show the efficacy of the proposed algorithm for cross-domain classification with disparate label sets as it outperforms other approaches by at least 15%. Coll ID(#) refers to the target collection and the corresponding count of class-labels. Results in Table 4 also compare the performance of the proposed technique without the distributional normalization of the auxiliary training set, referred to as "W/O". Results suggest that suitably weighing instances from the auxiliary training set mitigates the distributional variations and enhances the cross-domain performance by at least 3.3%.

5.4.2 Results on the 20Newsgroups Dataset

Results in Table 5 show that the proposed algorithm outperforms other techniques for both the experiments by at least 15 % and 18% respectively on the 20 Newsgroups dataset. In Table 5, "-" refers to the cases where a single source domain with the same number of class-labels as in the target domain is not available. In "Exp-1" where the source and target categories vary in terms of counts as well as their connotations, the proposed algorithm efficiently induces the classes in the unlabeled target domain using the partial transferable knowledge from multiple sources. For "Exp-2", it is observed that the performance of the proposed algorithm is better than the performance in "Exp-1" as the target categories have closely related categories (from the same group) in the source domains. Table 5 reports the average performance across all the 720 possible combinations of target domains with a standard deviation of 2.6.

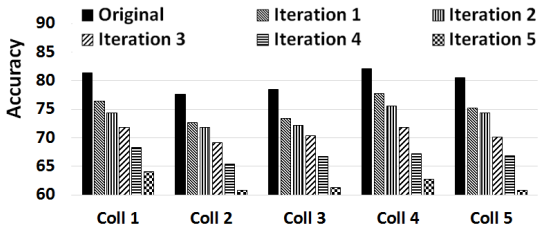


Figure 5: Effects of selected source collections on the OSM dataset.

Table 5: Summarizes the performance of the proposed algorithm on the 20Newsgroups dataset.

Target(#)	BL	CL	SCL	W/O	Proposed	Gold
Grp 1 (5)	-	48.6	-	79.4	80.8	85.6
Grp 2 (4)	62.7	50.2	62.7	78.3	83.6	89.2
Grp 3 (4)	64.3	54.8	64.4	81.6	85.3	90.4
Grp 4 (3)	69.6	55.6	67.3	82.2	86.4	92.5
Grp 5 (3)	69.7	56.4	70.3	83.6	85.3	91.2
Grp 7 (5)	-	52.8	-	84.6	88.4	93.8

5.4.3 Effect of Multiple Source Domains

Table 6 validates our assertion that multiple sources are necessary to induce class-separability in the target domain as a single source is not sufficient to cater to the heterogeneity of class-labels across domains. It also suggests that the proposed algorithm can learn class-separability in the target domain by using arbitrary diverse class-labels from different sources and does not necessarily require class-labels to follow any sort of coarse-to-fine mapping across domains.

To evaluate the effects of using multiple sources, further experiments were performed by varying the number of available source domains. For the OSM dataset, we varied the number of available source collections from 1 to 73 starting with the most similar source collection and repeatedly adding the next most similar collection in the pool of available collections. We observe that even the most similar collection was not independently sufficient to induce classes in the target collection and it was favorable to exploit multiple collections. Moreover, adding collections based on similarity to the target collection had a better likelihood of achieving higher performance as compared to adding random collections.

In another experiment, we first identified the source collections which contributed to learning the target task. We removed these collections and applied the proposed algorithm on the remaining source collections. Figure 5 shows the perfor-

Table 6: Actual target domain class-labels and the corresponding source domain class clusters used to build the auxiliary training set.

Target Collection: Apple iOS 8	Associated Class-labels from multiple source collections
Locking apps & security	Anti Theft Features (**Coll ID:** 776 **on Apple iOS 6 plus**)
Extensibility features	Application update (**Coll ID:**720 **on Apple iOS Features**)
General features related marketing	General press (**Coll ID:** 163 **on XBOX Issues**)
Camera features	Camera (**Coll ID:** 775 **on Apple iPhone 6**)
Password with touch integration	Touch ID (**Coll ID:** 803 **on Apple iPad mini3**)
Health & fitness app	Reproductive health issues (**Coll ID:** 289 **on Healthcare**)
Location & Maps	Events (**Coll ID:** 502 **on L'Oreal**)
Firmware updates	Updates & patches (**Coll ID:** 478 **on Riot Game Support v2**)

mance of the proposed algorithm on 5 such iterations of removing the contributing source collections from the previous iteration. We observed a significant drop in the performance with each iteration which signifies the effectiveness of the proposed algorithm in extracting highly discriminating transferable knowledge from multiple sources.

5.4.4 Comparing with Domain Adaptation

We applied domain adaptation techniques considering the auxiliary training set to be a single source domain with the same number of classes as that in the target domain. We applied two of the widely used domain adaptation techniques, namely SCL (Blitzer et al., 2007) and SFA (Pan et al., 2010) referred to as "AuxSCL" and "AuxSFA" respectively. Results in Table 7 suggest that the proposed algorithm significantly outperforms "AuxSCL" and "AuxSFA" on the two datasets. Generally, existing domain adaptation techniques are built on the co-occurrences of the common features with the domain specific features and hence, capture how domain specific features in one domain behaves w.r.t to the domain specific features in the other domain. They assume homogeneous labels and expect the aligned features across domains to behave similarly for the prediction task. However, these features are misaligned when the label set across domains vary in terms of their connotations.

5.4.5 Effect of Different Representations

The proposed algorithm uses Doc2Vec (Le and Mikolov, 2014) for representing instances from multiple domains. However, the proposed algorithm can build on different representations and hence, we compare its performance with traditional TF-IDF representation (including unigrams and bigrams) and a dense representation using TF-IDF+PCA (reduced to a dimension such that it covers 90% of the variance). We observe that Doc2Vec representation clearly outperforms the other two representations as it addresses the drawbacks of bag-of n-gram models in terms of implicitly inheriting the semantics of the words in a document and offering a more generalizable concise vector representation.

Table 7: Comparing the proposed algorithm with existing domain adaptation algorithms.

Dataset	Target	SCL	SFA	Proposed
OSM	Coll 1	66.2	64.7	81.4
	Coll 2	63.8	62.6	77.6
	Coll 3	64,1	63.4	78.5
	Coll 4	64.2	65.2	82.1
	Coll 5	64.0	63.7	80.5
NG Exp-1	Grp 1	65.2	64.2	80.8
	Grp 2	68.2	65.3	83.6
	Grp 3	69.4	68.4	85.3
	Grp 4	70.3	69.2	86.4
	Grp 5	69.0	68.8	85.3
NG Exp-2	Grp 7	72.6	70.2	88.4

Table 8: Comparing different representations.

Dataset	Target	TF-IDF	TF-IDF +PCA	Doc2Vec
OSM	Coll 1	70.6	76.8	81.4
	Coll 2	69.5	74.2	77.6
	Coll 3	70.2	75.5	78.5
	Coll 4	71.6	77.9	82.1
	Coll 5	70.8	76.8	80.5
NG Exp-1	Grp 1	71.8	75.6	80.8
	Grp 2	73.6	77.5	83.6
	Grp 3	77.4	81.1	85.3
	Grp 4	76.6	82.5	86.4
	Grp 5	75.5	81.4	85.3
NG Exp-2	Grp 7	76.2	83.6	88.4

6 Conclusions

This paper presented the first study on cross-domain text classification in presence of multiple domains with disparate label sets and proposed a novel algorithm for the same. It proposed to extract partial transferable knowledge from across multiple source domains which was beneficial for inducing class-separability in the target domain. The transferable knowledge was assimilated in terms of selective labeled instances from different source domain to form a K-class auxiliary training set. Finally, a classifier was trained using this auxiliary training set, following a distribution normalizing instance weighing technique, to perform the classification task in the target domain. The efficacy of the proposed algorithm for cross-domain classification across disparate label sets will expand the horizon for ML-based algorithms to be more widely applicable in more general and practically observed scenarios.

References

Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *Proceedings of Association for Computational Linguistics*, pages 1–9.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

Himanshu Sharad Bhatt, Deepali Semwal, and Shourya Roy. 2015. An iterative similarity based adaptation technique for cross-domain text classification. In *Proceedings of Conference on Natural Language Learning*, pages 52–61.

Himanshu Sharad Bhatt, Arun Rajkumar, and Shourya Roy. 2016. Multi-source iterative adaptation for cross-domain classification. In *In Proceedings of International Joint Conference on Artificial Intelligence*.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of Association for Computational Linguistics*, pages 440–447.

Danushka Bollegala, David Weir, and John Carroll. 2013. Cross-domain sentiment classification using a sentiment sensitive thesaurus. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1719–1731.

Rita Chattopadhyay, Qian Sun, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. 2012. Multisource domain adaptation and its application to early detection of fatigue. *ACM Transactions on Knowledge Discovery from Data*, 6(4):1–26.

Bo Chen, Wai Lam, Ivor Tsang, and Tak-Lam Wong. 2009. Extracting discriminative concepts for domain adaptation in text mining. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pages 179–188.

Chih-Chung Chang Chih-Wei Hsu and Chih-Jen Lin. 2003. A practical guide to support vector classification. Technical report, Department of Computer Science, National Taiwan University.

Koby Crammer, Michael Kearns, and Jennifer Wortman. 2008. Learning from multiple sources. *Journal of Machine Learning Research*, 9(1):1757–1774.

Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. 2007. Co-clustering based classification for out-of-domain documents. In *Proceedings of International Conference on Knowledge Discovery and Data Mining*, pages 210–219.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*.

Inderjit S. Dhillon and Dharmendra S. Modha. 2001. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175.

Lixin Duan, Ivor W. Tsang, Dong Xu, and Tat-Seng Chua. 2009. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings International Conference on Machine Learning*, pages 289–296.

Lixin Duan, Dong Xu, and Ivor Wai-Hung Tsang. 2012. Domain adaptation from multiple sources: a domaindependent regularization approach. *IEEE Transactions on Neural Networks and Learning Systems*, 23(3):504518.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: a library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

J. Jiang and C. Zhai. 2007. Instance weighting for domain adaptation in NLP. In *Proceedings of Association for Computational Linguistics*, volume 7, pages 264–271.

Jing Jiang. 2009. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1012–1020. Association for Computational Linguistics.

Young-Bum Kim, Karl Stratos, Ruhi Sarikaya, and Minwoo Jeong. 2015. New transfer learning techniques for disparate label sets. In *Proceedings of Association for Computational Linguistics*.

Abhishek Kumar and Hal Daume III. 2012. Learning task grouping and overlap in multi-task learning. *arXiv preprint arXiv:1206.6417*.

Ken Lang. 1995. NewsWeeder: Learning to filter netnews. In *Proceedings of International Conference on Machine Learning*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. 2009. Domain adaptation with multiple sources. In *Advances in Neural Information Processing Systems*, pages 1041–1048.

Andreas Maurer, Massimiliano Pontil, and Bernardino Romera-Paredes. 2012. Sparse coding for multitask and transfer learning. *arXiv preprint arXiv:1209.0738*.

Thomas M. Mitchell. 1997. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

Sinno Jialin Pan and Qiang Yang. 2010. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Sinno Jialin Pan, Xiaochuan Ni, Jian-Tao Sun, Qiang Yang, and Zheng Chen. 2010. Cross-domain sentiment classification via spectral feature alignment. In *Proceedings of International Conference on World Wide Web*, pages 751–760.

Sinno Jialin Pan, , Ivor W. Tsang, James T. Kwok, and Qiang Yang. 2011. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: Transfer learning from unlabeled data. In *International Conference on Machine Learning*, pages 759–766.

Gabriele Schweikert and Christian Widmer. 2008. An empirical analysis of domain adaptation algorithms for genomic sequence analysis. In *Advances in Neural Information Processing Systems*, pages 1433–1440.

Shi-Liang Sun and Hong-Lei Shi. 2013. Bayesian multi-source domain adaptations. In *International Conference on Machine Learning and Cybernetics*, pages 24–28.

Qian Sun, Rita Chattopadhyay, Sethuraman Panchanathan, and Jieping Ye. 2011. A two-stage weighting framework for multi-source domain adaptation. In *Advances in Neural Information Processing Systems*, pages 505–513.

Shiliang Sun, Zhijie Xu, and Mo Yang. 2013. Transfer learning with part-based ensembles. In *Multiple Classifier Systems*, volume 7872, pages 271–282.

Vijay V. Vazirani. 2003. *Approximation Algorithms*. Springer-Verlag Berlin Heidelberg.

Zhijie Xu and Shiliang Sun. 2012. Multi-source transfer learning with multi-view adaboost. In *Proceedings of International Conference on Neural Information Processing*, pages 332–339.

Linli Xu, Aiqing Huang, Jianhui Chen, and Enhong Chen. 2015. Exploiting task-feature co-clusters in multi-task learning. In *Proceedings of Association for the Advancement of Artificial Intelligence*, pages 1931–1937.

Jun Yang, Rong Yan, and Alexander G. Hauptmann. 2007. Cross-domain video concept detection using adaptive svms. In *Proceedings of International Conference on Multimedia*, pages 188–197.

Morphological Smoothing and Extrapolation of Word Embeddings

Ryan Cotterell
Department of Computer Science
Johns Hopkins University, USA
ryan.cotterell@jhu.edu

Hinrich Schütze
CIS
LMU Munich, Germany
inquiries@cis.lmu.org

Jason Eisner
Department of Computer Science
Johns Hopkins University, USA
jason@cs.jhu.edu

Abstract

Languages with rich inflectional morphology exhibit lexical data sparsity, since the word used to express a given concept will vary with the syntactic context. For instance, each count noun in Czech has 12 forms (where English uses only singular and plural). Even in large corpora, we are unlikely to observe all inflections of a given lemma. This reduces the vocabulary coverage of methods that induce continuous representations for words from distributional corpus information. We solve this problem by exploiting existing morphological resources that can enumerate a word's component morphemes. We present a latent-variable Gaussian graphical model that allows us to extrapolate continuous representations for words not observed in the training corpus, as well as smoothing the representations provided for the observed words. The latent variables represent embeddings of morphemes, which combine to create embeddings of words. Over several languages and training sizes, our model improves the embeddings for words, when evaluated on an analogy task, skip-gram predictive accuracy, and word similarity.

1 Introduction

Representations of words as high-dimensional real vectors have been shown to benefit a wide variety of NLP tasks. Because of this demonstrated utility, many aspects of vector representations have been explored recently in the literature. One of the most interesting discoveries is that these representations capture meaningful morpho-syntactic and semantic properties through very simple linear relations: in a semantic vector space, we observe that

$$v_{talked} - v_{talk} \approx v_{drank} - v_{drink}. \quad (1)$$

That this equation approximately holds across many morphologically related 4-tuples indicates that the learned embeddings capture a feature of English morphology—adding the past tense feature roughly corresponds to adding a certain vector. Moreover, manipulating this equation yields what we will call the **vector offset method** (Mikolov et al., 2013c) for approximating other vectors. For instance, if we only know the vectors for the Spanish words *comieron* (*ate*), *comemos* (*eat*) and *bebieron* (*drank*), we can produce an approximation of the vector for *bebemos* (*drink*), as shown in Figure 1.

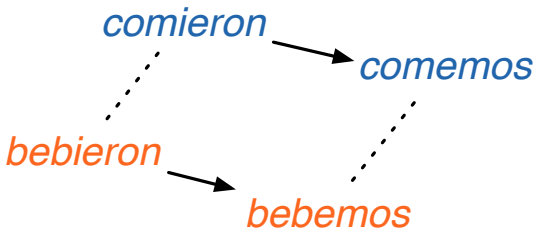


Figure 1: A visual depiction of the vector offset method for morpho-syntactic analogies in $\mathbb{R}^2$. We expect *bebieron* and *bebemos* to have the same relation (vector offset shown as solid vector) as *comieron* and *comemos*.

Many languages exhibit much richer morphology than English. While English nouns commonly take two forms – singular and plural—Czech nouns take 12 and Turkish nouns take over 30. This increase in word forms per lemma creates considerable data sparsity. Fortunately, for many languages there exist large morphological lexicons, or better yet, morphological tools that can analyze *any* word form—meaning that we have analyses (usually accurate) for forms that were unobserved or rare in our training corpus.

Our proposed method runs as a fast post-processor (taking under a minute to process 100-dimensional embeddings of a million observed word types) on the output of any existing tool that constructs word embeddings, such as WORD2VEC.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1651–1660,
Berlin, Germany, August 7-12, 2016.

	Indicative		Subjunctive	
	Sg	Pl	Sg	Pl
1	bebo	bebemos	beba	bebamos
2	bebes	bebéis	bebas	bebáis
3	bebe	beben	beba	beban

Table 1: The paradigm of the Spanish verb BEBER (to drink). The paradigm actually consists of > 40 word forms; only the present tense portion is shown here.

In this output, some embeddings are noisy or missing, due to sparse training data. We correct these problems by using a Gaussian graphical model that jointly models the embeddings of morphologically related words. Inference under this model can smooth the noisy embeddings that were observed in the WORD2VEC output. In the limiting case of a word for which no embedding was observed (equivalent to infinite noise), inference can extrapolate one based on the observed embeddings of related words—a kind of global version of the vector offset method. The *structure* of our graphical model is defined using morphological lexicons, which supply analyses for each word form.

We conduct a comprehensive study of our ability to modify and generate vectors across five languages. Our model also dramatically improves performance on the morphological analogy task in many cases: e.g., accuracy at selecting the nominative plural forms of Czech nouns is 89%, ten times better than the standard analogy approach.

2 Background: Inflectional Morphology

Many languages require every verb token to be inflected for certain properties, such as person, number, tense, and mood. A verbal **paradigm** such as Table 1 lists all the inflected forms of a given verb. We may refer to this verb in the abstract by its **lemma**, BEBER—but when using it in a sentence, we must instead select from its paradigm the word type, such as *bebéis*, that expresses the contextually appropriate properties. Noun tokens in a language may similarly be required to be inflected for properties such as case, gender, and number.

A content word is chosen by specifying a lemma (which selects a particular paradigm) together with some inflectional attributes (which select a particular slot within that paradigm). For example, [Lemma=EAT, Person=3, Number=SINGULAR, Tense=PRESENT] is a bundle of attribute-value pairs that would be jointly expressed in English by the word form *eats* (Sylak-Glassman et al., 2015).

The regularities observed by Mikolov et al. (2013c) hold between words with similar attribute-value pairs. In Spanish, the word *beben* "they drink" (Table 1) can be analyzed as expressing the bundle [Lemma=BEBER, Person=3, Number=PLURAL, Tense=PRESENT]. Its vector similarity to *bebemos* "we drink" is due to the fact that both word forms have the same lemma BEBER. Likewise, the vector similarity of *beben* to *comieron* "they ate" is due to the conceptual similarity of their lemmas, BEBER "drink" and COMER "eat". Conversely, that *beben* is similar to *preguntan* "they ask" is caused by shared inflectional attributes [Person=3, Number=PLURAL, Tense=PRESENT]. Under cosine similarity, the most similar words are often related on both axes at once: e.g., one of the word forms closest to *beben* typically is *comen* "they eat".

3 Approach

Following this intuition, we fit a directed Gaussian graphical model (GGM) that simultaneously considers (i) each word's embedding (obtained from an embedding model like WORD2VEC) and (ii) its morphological analysis (obtained from a lexical resource). We then use this model to smooth the provided embeddings, and to generate embeddings for unseen inflections. For a lemma covered by the resource, the GGM can produce embeddings for all its forms (if at least one of these forms has a known embedding); this can be extended to words not covered using a guesser like MORFESSOR (Creutz and Lagus, 2007) or CHIPMUNK (Cotterell et al., 2015a).

A major difference of our approach from related techniques is that our model uses existing morphological resources (e.g., morphological lexicons or finite-state analyzers) rather than semantic resources (e.g., WordNet (Miller et al., 1990) and PPDB (Ganitkevitch et al., 2013)). The former tend to be larger: we often can analyze more words than we have semantic representations for.

It would be possible to *integrate* our GGM into the training procedure for a word embedding system, making that system sensitive to morphological attributes. However, the *postprocessing* approach in our present paper lets us use any existing word embedding system as a black box. It is simple to implement, and turns out to get excellent results, which will presumably improve further as

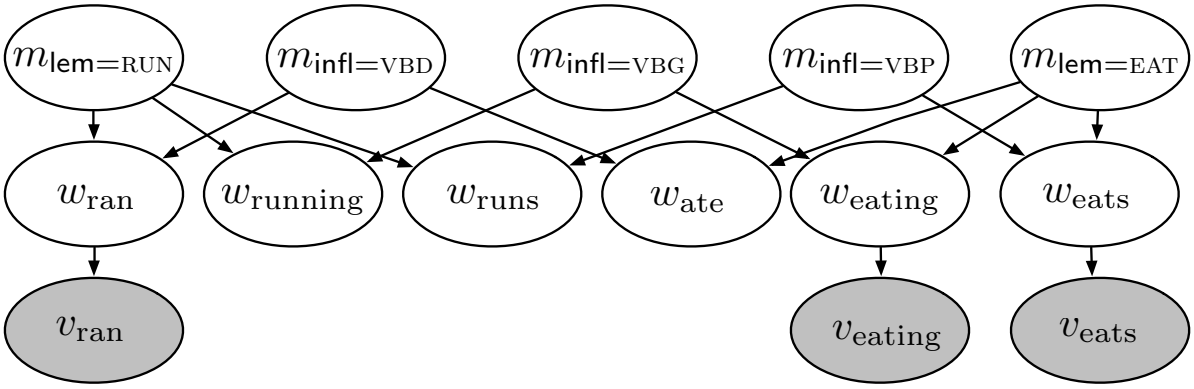


Figure 2: A depiction of our directed Gaussian graphical model (GGM) for the English verbal paradigm. Each variable represents a vector in $\mathbb{R}^n$; thus, this is not the traditional presentation of a GGM in which each node would be a *single real-valued random variable*, but each node represents a *real-valued random vector*. The shaded nodes v_i at the bottom are observed word embeddings. The nodes w_i at the middle layer are smoothed or extrapolated word embeddings. The nodes m_k at the top are latent embeddings of morphemes.

better black boxes become available.

4 A Generative Model

Figure 2 draws our GGM's structure as a Bayes net. In this paper, we loosely use the term "morpheme" to refer to an attribute-value pair (possibly of the form Lemma=...). Let $\mathcal{M}$ be the set of all morphemes. In our model, each morpheme $k \in \mathcal{M}$ has its *own* latent embedding $m_k \in \mathbb{R}^n$. These random variables are shown as the top layer of Figure 2. We impose an IID spherical Gaussian prior on them (similar to L_2 regularization with strength $\lambda > 0$):

$$m_k \sim \mathcal{N}(\mathbf{0}, \lambda^{-1} I), \ \forall k \tag{2}$$

Let $\mathcal{L}$ be the lexicon of all word types that appear in our lexical resource. (The noun and verb senses of *bat* are separate entries in $\mathcal{L}$.) In our model, each word $i \in \mathcal{L}$ has a latent embedding $w_i \in \mathbb{R}^n$. These random variables are shown as the middle layer of Figure 2. We assume that each w_i is simply a *sum* of the m_k for its component morphemes $M_i \subseteq \mathcal{M}$ (shown in Figure 2 as w_i's parents), *plus* a Gaussian perturbation:

$$w_i \sim \mathcal{N}(\sum_{k \in M_i} m_k, \Sigma_i), \ \forall i \tag{3}$$

This perturbation models idiosyncratic usage of word i that is not predictable from its morphemes. The covariance matrix Σ_i is shared for all words i with the same coarse POS (e.g., VERB).

Our system's *output* will be a guess of all of the w_i. Our system's *input* consists of noisy estimates v_i for some of the w_i, as provided by a black-box word embedding system run on some large corpus $\mathcal{C}$. (Current systems estimate the same vector for both senses of *bat*.) These observed random variables are shown as the bottom layer of Figure 2. We assume that the black-box system would have recovered the "true" w_i if given enough data, but instead it gives a noisy small-sample estimate

$$v_i \sim \mathcal{N}(w_i, \tfrac{1}{n_i}\Sigma_i'), \ \forall i \tag{4}$$

where n_i is the count of word i in training corpus $\mathcal{C}$.

This formula is inspired by the central limit theorem, which guarantees that v_i's distribution would approach (4) (as $n_i \to \infty$) if it were estimated by averaging a set of n_i noisy vectors drawn IID from *any* distribution with mean w_i (the truth) and covariance matrix Σ_i'. A system like WORD2VEC does not precisely do that, but it does choose v_i by aggregating (if not averaging) the influences from the contexts of the n_i tokens.

The parameters $\lambda, \Sigma_i, \Sigma_i'$ now have likelihood

$$p(\boldsymbol{v}) = \int p(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{m}) \, d\boldsymbol{w} \, d\boldsymbol{m}, \text{ where} \tag{5}$$

$$p(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{m}) = \prod_{k \in \mathcal{M}} p(m_k) \cdot \prod_{i \in \mathcal{L}} p(w_i \,|\, m_k : k \in M_i) \cdot p(v_i \,|\, w_i) \tag{6}$$

Here $\boldsymbol{m} = \{m_k : k \in \mathcal{M}\}$ represents the collection of all latent morpheme embeddings, and similarly $\boldsymbol{w} = \{w_i : i \in \mathcal{L}\}$ and $\boldsymbol{v} = \{v_i : i \in \mathcal{L}\}$. We take $p(v_i \mid w_i) = 1$ if no observation v_i exists.

How does the model behave qualitatively? If $\hat{w}_i$ is the MAP estimate of w_i, then $\hat{w}_i \to v_i$ as $n_i \to \infty$, but $\hat{w}_i \to \sum_{k \in M_i} m_k$ as $n_i \to 0$. This

is because (3) and (4) are in tension; when n_i is small, (4) is weaker and we get more smoothing. The morpheme embeddings m_k are largely determined from the observed embeddings v_i of the frequent words (since m_k aims via (2)–(3) to explain w_i, which $\approx v_i$ when i is frequent). That determines the compositional embedding $\sum_{k \in M_i} m_k$ toward which the w_i of a rarer word is smoothed (away from v_i). If v_i is not observed or if $n_i = 0$, then $\hat{w}_i = \sum_{k \in M_i} m_k$ exactly.

5 Inference

Suppose first that the model parameters are known, and we want to reconstruct the latent vectors w_i. Because the joint density $p(\boldsymbol{v}, \boldsymbol{w}, \boldsymbol{m})$ in (6) is a product of (sometimes degenerate) Gaussian densities, it is itself a highly multivariate Gaussian density over all elements of all vectors.[1] Thus, the posterior marginal distribution of each w_i is Gaussian as well. A good deal is known about how to exactly compute these marginal distributions of a Gaussian graphical model (e.g., by matrix inversion) or at least their means (e.g., by belief propagation) (Koller and Friedman, 2009).

For this paper, we adopt a simpler method—MAP estimation of all latent vectors. That is, we seek the $\boldsymbol{w}, \boldsymbol{m}$ that jointly maximize (6). This is equivalent to minimizing

$$\sum_k \lambda ||m_k||_2^2 + \sum_i ||w_i - \sum_{k \in M_i} m_k||_{\Sigma_i}^2 + \sum_i ||v_i - w_i||_{\Sigma_i'/n_i}^2, \quad (7)$$

which is a simple convex optimization problem.[2] We apply block coordinate descent until numerical convergence, in turn optimizing each vector m_k or w_i with all other vectors held fixed. This finds the global minimum (convex objective) and is extremely fast even when we have over a hundred million real variables. Specifically, we update

$$m_k \leftarrow \big(\lambda I + \sum_{i \in W_k} \mathfrak{T}_i\big)^{-1} \sum_{i \in W_k} \mathfrak{T}_i (w_i - \sum_{j \in M_i, j \neq k} m_j),$$

where $\mathfrak{T} \stackrel{\text{def}}{=} \Sigma^{-1}$ is the inverse covariance matrix and $W_k \stackrel{\text{def}}{=} \{i : k \in M_i\}$. This updates m_k so the partial derivatives of (7) with respect to the components of m_k are 0. In effect, this updates m_k to a *weighted average* of several vectors. Morpheme k participates in words $i \in W_k$, so its vector m_k is updated to the average of the contributions $(w_i - \sum_{j \in M_i, j \neq k} m_j)$ that m_k would *ideally* make to the embeddings w_i of those words. The contribution of w_i is "weighted" by the inverse covariance matrix $\mathfrak{T}_i$. Because of prior (2), $\mathbf{0}$ is also included in the average, "weighted" by λI.

Similarly, the update rule for w_i is

$$w_i \leftarrow (n_i \mathfrak{T}_i' + \mathfrak{T}_i)^{-1} \big(n_i \mathfrak{T}_i' v_i + \mathfrak{T}_i \sum_{k \in M_i} m_k\big),$$

which can similarly be regarded as a weighted average of the observed and compositional representations.[3] See Appendix C for the derivations.

6 Parameter Learning

We wish to optimize the model parameters $\lambda, \Sigma_i, \Sigma_i'$ by empirical Bayes. That is, we do not have a prior on these parameters, but simply do maximum likelihood estimation. A standard approach is the Expectation-Maximization or EM algorithm (Dempster et al., 1977) to *locally* maximize the likelihood. This alternates between reconstructing the latent vectors given the parameters (E step) and optimizing the parameters given the latent vectors (M step). In this paper, we use the Viterbi approximation to the E step, that is, MAP inference as described in section 5. Thus, our overall method is Viterbi EM.

As all conditional probabilities in the model are Gaussian, the M step has closed form. MLE estimation of a covariance matrix is a standard result—in our setting the update to Σ_i takes the form:

$$\Sigma_c \leftarrow \frac{1}{N_c} \sum_{i : c \in C(i)} (w_i - \sum_{k \in M_i} m_k)(w_i - \sum_{k \in M_i} m_k)^T,$$

where $C(i)$ are i's POS tags, $N_c = |\{i | c \in C(i)\}|$ and Σ_c is the matrix for the c^{th} POS tag (the matrices are tied by POS). In this paper we simply fix $\Sigma_i' = I$ rather than fitting it.[4] Also, we tune the hyperparameter λ on a development set, using grid search over the values $\{0.1, 0.5, 1.0\}$.

[1] Its inverse covariance matrix is highly sparse: its pattern of non-zeros is related to the graph structure of Figure 2. (Since the graphical model in Figure 2 is *directed*, the inverse covariance matrix has a sparse Cholesky decomposition that is even more directly related to the graph structure.)

[2] By definition, $||\boldsymbol{x}||_A^2 \stackrel{\text{def}}{=} \boldsymbol{x}^T A \boldsymbol{x}$.

[3] If v_i is not observed, take $n_i = 0$. In fact it is not necessary to represent this w_i during optimization. Simply omit i from all W_k. *After* convergence, set $w_i \leftarrow \sum_{k \in M_i} m_k$.

[4] Note that it is not necessary to define it as $\lambda' I$, introducing a new scale parameter λ', since doubling λ' would have the same effect on the MAP update rules as halving λ and Σ_i.

Viterbi EM can be regarded as block coordinate descent on the negative log-likelihood function, with E and M steps both improving this common objective along different variables. We update the parameters (M step above) after each 10 passes of updating the latent vectors (section 5's E step).

7 Related Work

Our postprocessing strategy is inspired by Faruqui et al. (2015), who designed a retrofitting procedure to modify pre-trained vectors such that their relations match those found in semantic lexicons. We focus on morphological resources, rather than semantic lexicons, and employ a generative model. More importantly, in addition to modifying vectors of observed words, our model can generate vectors for forms not observed in the training data.

Wieting et al. (2015) compute compositional embeddings of *phrases*, with their simplest method being additive (like ours) over the phrase's words. Their embeddings are tuned to fit observed phrase similarity scores from PPDB (Ganitkevitch et al., 2013), which allows them to smooth and extend PPDB just as we do to WORD2VEC output.

Using morphological resources to enhance embeddings at *training* time has been examined by numerous authors. Luong et al. (2013) used MORFESSOR (Creutz and Lagus, 2007), an unsupervised morphological induction algorithm, to segment the training corpus. They then trained a recursive neural network (Goller and Kuchler, 1996; Socher, 2014) to generate compositional word embeddings. Our model is much simpler and faster to train. Their evaluation was limited to English and focused on rare English words. dos Santos and Zadrozny (2014) introduced a neural tagging architecture (Collobert et al., 2011) with a character-level convolutional layer. Qiu et al. (2014) and Botha and Blunsom (2014) both use MORFESSOR segmentations to augment WORD2VEC and a log-bilinear (LBL) language model (Mnih and Hinton, 2007), respectively. Similar to us, they have an additive model of the semantics of morphemes, i.e., the embedding of the word form is the *sum* of the embeddings of its constituents. In contrast to us, however, both include the word form itself in the sum. Finally, Cotterell and Schütze (2015) jointly trained an LBL language model and a morphological tagger (Hajič, 2000) to encourage the embeddings to encode rich morphology. With the exception of (Cotterell and Schütze, 2015), all of the above methods use *unsupervised* methods to infuse word embeddings with morphology. Our approach is supervised in that we use a morphological lexicon, i.e., a manually built resource.

Our model is also related to other generative models of real vectors common in machine learning. The simplest of them is probabilistic principal component analysis (Roweis, 1998; Tipping and Bishop, 1999), a generative model of matrix factorization that explains a set of vectors via latent low-dimensional vectors. Probabilistic canonical correlation analysis similarly explains a set of *pairs* of vectors (Bach and Jordan, 2005).

Figure 2 has the same topology as our graphical model in (Cotterell et al., 2015b). In that work, the random variables were *strings* rather than vectors. Morphemes were combined into words by concatenating strings rather than adding vectors, and then applying a stochastic edit process (modeling phonology) rather than adding Gaussian noise.

8 Experiments

We perform three experiments to test the ability of our model to improve on WORD2VEC. To reiterate, our approach does not generate or analyze a word's spelling. Rather, it uses an *existing* morphological analysis of a word's spelling (constructed manually or by a rule-based or statistical system) as a resource to improve its embedding.

In our first experiment, we attempt to identify a corpus word that expresses a given set of morphological attributes. In our second experiment, we attempt to use a word's embedding to predict the words that appear in its context, i.e., the skip-gram objective of Mikolov et al. (2013a). Our third example attempts to use word embeddings to predict human similarity judgments.

We experiment on 5 languages: Czech, English, German, Spanish and Turkish. For each language, our corpus data consists of the full Wikipedia text. Table 5 in Appendix A reports the number of types and tokens and their ratio. The lexicons we use are characterized in Table 6: MorfFlex CZ for Czech (Hajič and Hlaváčová, 2013), CELEX for English and German (Baayen et al., 1993) and lexicons for Spanish and Turkish that were scraped from Wiktionary by Sylak-Glassman et al. (2015).

Given a finite training corpus $\mathcal{C}$ and a lexicon $\mathcal{L}$,[5] we generate embeddings v_i for all word

[5] $\mathcal{L}$ is finite in our experiments. It could be infinite (though still incomplete) if a morphological guesser were used.

	Nom Sg	Nom Pl	Gen Sg	Gen Pl	Dat Sg	Dat Pl	Acc Sg	Acc Pl	Ins Sg	Ins Pl	Voc Sg	Voc Pl
GGM	48	89	81	43	30	33	36	87	33	48	49	89
Nom Sg		4.4	2.6	1.7	3.9	0.33	2.1	1.6	3.5	0.59	0	5.2
Nom Pl	4.6		1.7	3.4	1.6	2.2	2.3	10	1.9	7	0	0
Gen Sg	2	1.7		2.9	0.67	0.5	0.98	0.5	1.3	1.6	0	0.33
Gen Pl	1.8	4.9	4.8		4.7	2.5	3.3	7.2	3.1	0.92	1.7	3.7
Dat Sg	2	2	0	3.3		0	1.3	0.61	2.8	0	0	2
Dat Pl	3.8	4.8	2.4	8	0.86		0.33	7.2	1	2.7	0	5
Acc Sg	3.9	2.7	0.74	2.1	1.7	0		2.4	0.58	2.5	5	2.2
Acc Pl	1.3	15	1.2	6.2	2.6	2.6	2.2		2.8	2.7	0	0
Ins Sg	4	1.6	1.6	2.3	2.7	0.36	1.6	2.4		0.18	0.83	2.9
Ins Pl	2	6	3	2.5	1.7	4.3	3.6	7.7	0.69		0	4.6
Voc Sg	0	0	0	0	0	5	5	0	0.83	0		0
Voc Pl	4.7	0	0	3.5	1.3	5.5	2.2	0	2.1	3.7	0	

	1ps Ps	2ps Ps	3ps Ps	1pp Ps	2pp Ps	3pp Ps	1ps Pt	2ps Pt	3ps Pt	1pp Pt	2pp Pt	3pp Pt
GGM	8.8	4.5	30	14	10	49	9.4	8.1	41	9.5	0	43
1ps Ps		0	0.79	2.1	0	0.33	0.83	0	3.4	1	0	3
2ps Ps	0		14	0.5	5	0.091	0.83	0	0.17	0	0	0.28
3ps Ps	1.2	9.3		1.2	0	29	1.7	3.3	13	2.5	0	8.2
1pp Ps	0.24	0	1.9		0	2.7	2.1	0	2.6	0	0	2.2
2pp Ps	0	0	0	0		0	0	0	0	0	0	0
3pp Ps	1.6	0	30	4	0		0.24	0	8	4.4	0	14
1ps Pt	0.14	0.5	2	3.2	0	0.33		0	1.4	0.83	0	1.7
2ps Pt	0	0	0	0	0	0	0		0	0	0	0
3ps Pt	7.6	0	13	1.3	0	7.2	0.86	0		3.5	0	36
1pp Pt	1.7	0	3.3	0	0	2.8	1.7	0	3.4		0	2
2pp Pt	0	0	0	0	0	0	0	0	0	0		0
3pp Pt	7.3	1.3	7.2	3.4	0	13	1.4	0	37	1.8	0	

Table 2: The two tables show how the Gaussian graphical model (GGM) compares to various analogies on Czech nouns (left) and Spanish verbs (right).The numbers in each cell represent the accuracy (larger is better). The columns represent the inflection of the word i to be predicted. Our GGM model is the top row. The other rows subdivide the baseline analogy results according to the inflection of source word a. Abbreviations: in the Czech noun table (left), the first word indicates the case and the second the number, e.g., Dat Sg = Dative Singular. In the Spanish verb table (right), the first word is the person and number and the second the tense, e.g., 3pp Pt = 3rd-person plural past.

types $i \in \mathcal{C}$, using the GENSIM implementation (Řehůřek and Sojka, 2010) of the WORD2VEC hierarchical softmax skip-gram model (Mikolov et al., 2013a), with a context size of 5. We set the dimension n to 100 for all experiments.[6]

We then apply our GGM to generate smoothed embeddings w_i for all word types $i \in \mathcal{C} \cap \mathcal{L}$. (Recall that the noun and verb sense of *bats* are separate types in $\mathcal{L}$, even if conflated in $\mathcal{C}$, and get separate embeddings.) How do we handle other word types? For an out-of-vocabulary (OOV) test word $i \notin \mathcal{C}$, we will extrapolate $w_i \leftarrow \sum_{k \in M_i} m_k$ on demand, as the GGM predicts, provided $i \in \mathcal{L}$. If any of these morphemes m_k were themselves never seen in $\mathcal{C}$, we back off to the mode of the prior to take $m_k = \mathbf{0}$.[7] Our experiments also encounter out-of-lexicon (OOL) test words $i \notin \mathcal{L}$, for which we have no morphological analysis; here we take $w_i = v_i$ (unsmoothed) if $i \in \mathcal{C}$ and $w_i = \mathbf{0}$ otherwise.

8.1 Experiment 1: Extrapolation vs. Analogy

Our first set of experiments uses embeddings for word selection. Our prediction task is to identify the unique word $i \in \mathcal{C}$ that expresses the morphological attributes M_i. To do this, we predict a target embedding x, and choose the most similar unsmoothed word by cosine distance, $\hat{\imath} = \text{argmax}_{j \in \mathcal{C}}\, v_j \cdot x$. We are scored correct if $\hat{\imath} = i$. Our experimental design ensures that $i \notin \mathcal{L}$, since if it were, we could trivially find i simply by consulting $\mathcal{L}$. The task is to identify *missing* lexical entries, by exploiting the distributional properties in $\mathcal{C}$.[8] Given the input bundle M_i, our method predicts the embedding $x = \sum_{k \in M_i} m_k$, and so looks for a word $j \in \mathcal{C}$ whose unsmoothed embedding $v_j \approx x$. The GGM's role here is to predict that the bundle M_i will be realized by something like x.

The baseline method is the analogy method of equation (1). This predicts the embedding x via the vector-offset formula $v_a + (v_b - v_c)$, where $a, b, c \in \mathcal{C} \cap \mathcal{L}$ are three other words sharing i's coarse part of speech such that M_i can be expressed as $M_a + (M_b - M_c)$.[9] Specifically, the baseline chooses a, b, c uniformly at random from all possibilities. (This is not too inefficient: given a, at most one choice of (b, c) is possible.) Note that the baseline extrapolates from the unsmoothed embeddings of 3 other words, whereas the GGM considers *all* words in $\mathcal{C} \cap \mathcal{L}$ that share i's morphemes.

[6] An additional important hyperparameter is the number of epochs. The default value in the GENSIM package is 5, which is suitable for larger corpora. We use this value for Experiments 1 and 3. Experiment 2 involves training on smaller corpora and we found it necessary to set the number of epochs to 10.

[7] One could in principle learn "backoff morphemes." For instance, if *borogoves* is analyzed as [Lemma=OOV_NOUN, Num=PL], we might want $m_{\text{Lemma=OOV\_NOUN}} \neq \mathbf{0}$ to represent novel *nouns*.

[8] The argmax selection rule does not exploit the fact that the entry is missing: it is free to incorrectly return some $\hat{\imath} \in \mathcal{L}$.

[9] More formally, $\overline{M_i} = \overline{M_a} + (\overline{M_b} - \overline{M_c})$, if we define $\overline{M}$ by $(\overline{M})_k = \mathbb{I}(k \in M)$ for all morphemes $k \in \mathcal{M}$. This converts morpheme bundle M to a $\{0, 1\}$ indicator vector $\overline{M}$ over $\mathcal{M}$.

	analogy	GGM		analogy	GGM		analogy	GGM		analogy	GGM		analogy	GGM
Ins^P	.067	.343*	Acc^S	.131	.582*	VBD	.052	.202*	Masc^S Part	.178	.340*	Loc^S	.002	.036*
Gen^S	.059	.632*	Nom^S	.136	.571*	VBG	.06	.102*	Fem^P Part	.230	.286	Abl^S	.001	.019*
Gen^P	.078	.242*	Dat^P	.074	.447*	NN	.051	.080*	Fem^S Part	.242	.235	Dat^S	.001	.037*
Nom^P	.076	.764*	Acc^P	.075	.682*	VBN	.052	.218*	Adj^S	.201	.286*	Acc^S	.001	.023*
Nom^S	.066	.290*	Gen^P	.075	.666*	NNS	.052	.114*	Gerund^S	.186	.449*	Ins^S	.001	.004
Voc^P	.063	.789*	Nom^P	.064	.665*	VB	.056	.275*	Gerund^P	.172	.311*	Nom^S	.001	.023*
Czech nouns			German nouns			English nouns & verbs			Spanish nouns, verbs & adj's			Turkish nouns		

Table 3: Test results for Experiment 1. The rows indicate the inflection of the test word i to be predicted (superscript ^{P} indicates plural, superscript ^{S} singular). The columns indicate the prediction method. Each number is an average over 10 training-test splits. Improvements marked with a $\star$ are statistically significant ($p < 0.05$) under a paired permutation test over these 10 runs.

Experimental Setup: A lexical resource consists of pairs (word form i, analysis M_i). For each language, we take a random 80% of these pairs to serve as the training lexicon $\mathcal{L}$ that is seen by the GGM. The remaining pairs are used to construct our prediction problems (given M_i, predict i), with a random 10% each as dev and test examples. We compare our method against the baseline method on ten such random training-test splits. We are releasing all splits for future research.

For some dev and test examples, the baseline method has no choice of the triple a, b, c. Rather than score these examples as incorrect, our baseline results do not consider them at all (which inflates performance). For each remaining example, to reduce variance, the baseline method reports the average performance on up to 100 a, b, c triples sampled uniformly without replacement.

The automatically created analogy problems ($a, b, c \rightarrow i$) solved by the baseline are similar to those of Mikolov et al. (2013c). However, most previous analogy evaluation sets evaluate only on 4-tuples of frequent words (Nicolai et al., 2015), to escape the need for smoothing, while ours also include infrequent words. Previous evaluation sets also tend to be translations of the original English datasets—leaving them impoverished as they therefore *only* test morpho-syntactic properties found *in English*. E.g., the German analogy problems of Köper et al. (2015) do not explore the four cases and two numbers in the German adjectival system. Thus our baseline analogy results are useful as a more comprehensive study *of the vector offset method for randomly sampled words*.

Results: Overall results for 5 languages are shown in Table 3. Additional rows break down performance by the inflection of the target word i. (The inflections shown are the ones for which the *baseline* method is most accurate.)

For almost all target inflections, GGM is significantly better than the analogy baseline. An extreme case is the vocative plural in Czech, for which GGM predicts vectors better by more than 70%. In other cases, the margin is slimmer; but GGM loses only on predicting the Spanish feminine singular participle. For Czech, German, English and Spanish the results are clear—GGM yields better predictions. This is not surprising as our method incorporates information from *multiple* morphologically related forms.

More detailed results for two languages are given in Table 2. Here, each row constrains the source word a to have a certain inflectional tag; again we average over up to 100 analogies, now chosen under this constraint, and again we discard a test example i from the test set if no such analogy exists. The GGM row considers all test examples.

Past work on morphosyntactic analogies has generally constrained a to be the unmarked (lemma) form (Nicolai et al., 2015). However, we observe that it is easier to predict one word form from another starting from a form that is "closer" in morphological space. For instance, it is easier to predict Czech forms inflected in the genitive plural from forms in nominative plural, rather than the nominative singular. Likewise, it is easier to predict a singular form from another singular form rather than from a plural form. It also is easier to predict partially syncretic forms, i.e., two inflected forms that share the same orthographic string; e.g., in Czech the nominative plural and the accusative plural are identical for inanimate nouns.

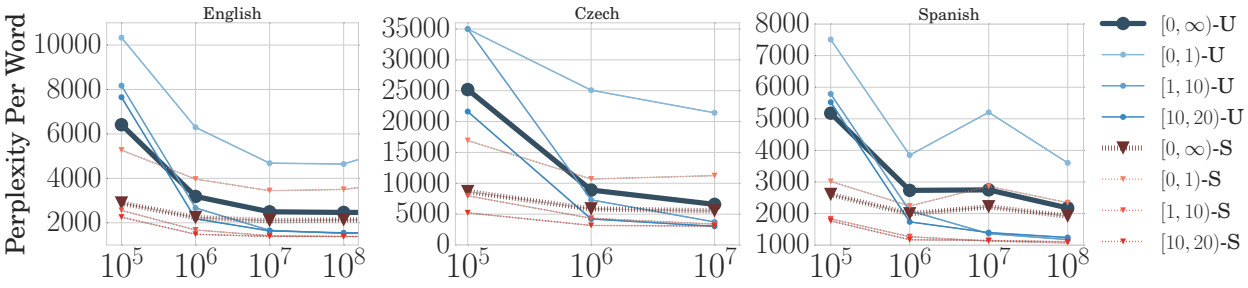


Figure 3: Results for the WORD2VEC skip-gram objective score (perplexity per predicted context word) on a held-out test corpus. The x-axis measures the size in tokens of the training corpus used to generate the model. We plot the held-out perplexity for the skip-gram model with Unsmoothed observed vectors $\boldsymbol{v}$ (solid ●) and Smoothed vectors $\boldsymbol{w}$ (barred ▼). The thickest, darkest curves show aggregate performance. The thinner, lighter versions show a breakdown according to whether the *predicting* word's frequency in the *smallest* training corpus falls in the range $[0, 1)$, $[1, 10)$, or $[10, 20)$ (from lightest to darkest and roughly from top to bottom). (These are the words whose representations we smooth; footnote 10 explains why we do not smooth the predicted context word.) We do not show $[20, \infty)$ since WORD2VEC randomly removes some tokens of high-frequency words ("subsampling"), similar in spirit to removing stop words. See Appendix B for more graphs.

8.2 Experiment 2: Held-Out Evaluation

We now evaluate the smoothed and extrapolated representations w_i. Fundamentally, we want to know if our approach improves the embeddings of the entire vocabulary, as if we had seen more evidence. But we cannot simply compare our smoothed vectors to "gold" vectors trained on much more data, since two different runs of WORD2VEC will produce incomparable embedding schemes. We must ask whether our embeddings improve results on a downstream task.

To avoid choosing a downstream task with a narrow application, we evaluate our embedding using the WORD2VEC skip-gram objective on *held-out* data—as one would evaluate a language model. If we believe that a better score on the WORD2VEC objective indicates generally more useful embeddings—which indeed we do as we optimize for it—then improving this score indicates that our smoothed vectors are superior. Concretely, the objective is

$$\sum_s \sum_t \sum_{j \in [t-5, t+5], j \neq t} \log_2 p_{\text{word2vec}}(T_{sj} \mid T_{st}), \quad (8)$$

where T_s is the s^{th} sentence in the *test* corpus, t indexes its tokens, and j indexes tokens near t. The probability model p_{word2vec} is defined in Eq. (3) of (Mikolov et al., 2013b). It relies on an embedding of the word form T_{st}.[10] Our baseline approach simply uses WORD2VEC's embeddings (or **0** for OOV words $T_{st} \notin \mathcal{C}$). Our GGM approach substitutes "better" embeddings when T_{st} appears in the lexicon $\mathcal{L}$ (if T_{st} is ambiguous, we use the mean w_i vector from all $i \in \mathcal{L}$ with spelling T_{st}).

Note that (8) is itself a kind of task of predicting words in context, resembling language modeling or a "cloze" task. Also, Taddy (2015) showed how to use this objective for document classification.

Experimental Setup: We evaluate GGM on the same 5 languages, but now hold out part of the corpus instead of part of the lexicon. We take the training corpus $\mathcal{C}$ to be the initial portion of Wikipedia of size 10^5, 10^6, 10^7 or 10^8. (We skip the 10^8 case for the smaller datasets: Czech and Turkish). The 10^7 tokens after that are the dev corpus; the next 10^7 tokens are the test corpus.

Results: We report results on three languages in Figure 3 and all languages in Appendix B. Smoothing from v_i to w_i helps a lot, reducing perplexity by up to 48% (Czech) with 10^5 training tokens and up to 10% (Spanish) even with 10^8 training tokens. This roughly halves the perplexity, which in the case of 10^5 training tokens, is equivalent to $8\times$ more training data. This is a clear win for lower-resource languages. We get larger gains from smoothing the rarer predicting words, but even words with frequency $\geq 10^{-4}$ benefit. (The exception is Turkish, where the large gains are confined to rare predicting words.) See Appendix B for more analysis.

[10] In the hierarchical softmax version, it also relies on a *separate* embedding for a variable-length bit-string encoding of the context word T_{sj}. Unfortunately, we do *not* currently know of a way to smooth these bit-string encodings (also found by WORD2VEC). However, it might be possible to directly incorporate morphology into the construction of the vocabulary tree that defines the bit-strings.

	English	German	Spanish
Forms / Lemma	1.8	6.3	8.1
Skip-Gram	**58.9**	36.2	37.8
GGM	**58.9**	**37.6**	**40.3**

Table 4: Word similarity results (correlations) using the WS-353 dataset in the three languages, in which it is available. Since all the words in WS-353 are lemmata, we report the average inflected form to lemma ratio for forms *appearing in* the datasets.

8.3 Experiment 3: Word Similarity

As a third and final experiment, we consider word similarity using the WS-353 data set (Finkelstein et al., 2001), translated into Spanish (Hassan and Mihalcea, 2009) and German (Leviant, 2016).[11] The datasets are composed of 353 pairs of words. Multiple native speakers were then asked to give an integral value between 1 and 10 indicating the similarity of that pair, and those values were then averaged. In each case, we train the GGM on the *whole* Wikipedia corpus for the language. Since in each language every word in the WS-353 set is in fact a lemma, we use the latent embedding our GGM learns in the experiment. In Spanish, for example, we use the learned latent morpheme embedding for the lemma BEBER (recall this takes information from every element in the paradigm, e.g., *bebemos* and *beben*), rather than the embedding for the infinitival form *beber*. In highly inflected languages we expect this to improve performance, because to get the embedding of a lemma, it leverages the distributional signal from *all* inflected forms of that lemma, not just a single one. Note that unlike previous retrofitting approaches, we do not introduce new semantic information into the model, but rather simply allow the model to better exploit the distributional properties already in the text, by considering words with related lemmata together. In essence, our approach embeds a lemma as the average of all words containing that lemma, *after* "correcting" those forms by subtracting off their other morphemes (e.g., inflectional affixes).

Results: As is standard in the literature, we report Spearman's correlation cofficient ρ between the averaged human scores and the cosine distance between the embeddings. We report results in Table 4. We additionally report the average number of forms per lemma. We find that we improve performance on the Spanish and German datasets over the original skip-gram vectors, but only equal the performance on English. This is not surprising as German and Spanish have roughly 3 and 4 times more forms per lemma than English. We speculate that cross-linguistically the GGM will improve word similarity scores more for languages with richer morphology.

[11] This dataset has yet to be translated into Czech or Turkish, nor are there any comparable resources in these languages.

9 Conclusion and Future Work

For morphologically rich languages, we generally will *not* observe, even in a large corpus, a high proportion of the word forms that exist in lexical resources. We have presented a Gaussian graphical model that exploits lexical relations documented in existing morphological resources to smooth vectors for observed words and extrapolate vectors for new words. We show that our method achieves large improvements over strong baselines for the tasks of morpho-syntactic analogies and predicting words in context. Future work will consider the role of derivational morphology in embeddings as well as noncompositional cases of inflectional morphology.

Acknowledgments

The first author was funded by a DAAD Long-Term Research Grant. This work was also partially supported by Deutsche Forschungsgemeinschaft (grat SCHU-2246/2-2 WordGraph) and by the U.S. National Science Foundation under Grant No. 1423276.

References

R Harald Baayen, Richard Piepenbrock, and Rijn van H. 1993. The CELEX lexical data base on CD-ROM.

Francis R Bach and Michael I Jordan. 2005. A probabilistic interpretation of canonical correlation analysis. Technical report, UC Berkeley.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *ICML*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12.

Ryan Cotterell and Hinrich Schütze. 2015. Morphological word embeddings. In *NAACL*.

Ryan Cotterell, Thomas Müller, Alexander Fraser, and Hinrich Schütze. 2015a. Labeled morphological segmentation with semi-Markov models. In *CoNLL*.

Ryan Cotterell, Nanyun Peng, and Jason Eisner. 2015b. Modeling word forms using latent underlying morphs and phonology. *TACL*, 3.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1):3.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, pages 1–38.

Cıcero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *ICML*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *NAACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: The concept revisited. In *WWW*. ACM.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *NAACL*.

Christoph Goller and Andreas Kuchler. 1996. Learning task-dependent distributed representations by backpropagation through structure. *Neural Networks*.

Jan Hajič and Jaroslava Hlaváčová. 2013. MorfFlex CZ. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Jan Hajič. 2000. Morphological tagging: Data vs. dictionaries. In *NAACL*.

Samer Hassan and Rada Mihalcea. 2009. Cross-lingual semantic relatedness using encyclopedic knowledge. In *EMNLP*.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. MIT press.

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. 2015. Multilingual reliability and semantic structure of continuous word spaces. In *IWCS*.

Ira Leviant. 2016. *Separated by an Un-common Language: Towards Judgment Language Informed Vector Space Modeling*. Ph.D. thesis, Technion.

Minh-Thang Luong, Richard Socher, and C Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *NAACL*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–244.

Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *ICML*.

Garrett Nicolai, Colin Cherry, and Grzegorz Kondrak. 2015. Morpho-syntactic regularities in continuous word representations: A multilingual study. In *Proceedings of Workshop on Vector Space Modeling for NLP*, pages 129–134.

Siyu Qiu, Qing Cui, Jiang Bian, Bin Gao, and Liu Tie-Yan. 2014. Co-learning of word representations and morpheme representations. In *COLING*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Sam Roweis. 1998. EM algorithms for PCA and SPCA. In *NIPS*.

Richard Socher. 2014. *Recursive Deep Learning for Natural Language Processing and Computer Vision*. Ph.D. thesis, Stanford University.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015. A language-independent feature schema for inflectional morphology. In *ACL*.

Matt Taddy. 2015. Document classification by inversion of distributed language representations. In *ACL*.

Michael E Tipping and Christopher M Bishop. 1999. Probabilistic principal component analysis. *Journal of the Royal Statistical Society B*, 61(3):611–622.

John Wieting, Mohit Bansal, Kevin Gimpel, Karen Livescu, and Dan Roth. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*.

Cross-lingual Models of Word Embeddings: An Empirical Comparison

Shyam Upadhyay[1] **Manaal Faruqui**[2] **Chris Dyer**[2] **Dan Roth**[1]
[1] Department of Computer Science, University of Illinois, Urbana-Champaign, IL, USA
[2] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
upadhya3@illinois.edu, mfaruqui@cs.cmu.edu
cdyer@cs.cmu.edu, danr@illinois.edu

Abstract

Despite interest in using cross-lingual knowledge to learn word embeddings for various tasks, a systematic comparison of the possible approaches is lacking in the literature. We perform an extensive evaluation of four popular approaches of inducing cross-lingual embeddings, each requiring a different form of supervision, on four typologically different language pairs. Our evaluation setup spans four different tasks, including intrinsic evaluation on mono-lingual and cross-lingual similarity, and extrinsic evaluation on downstream semantic and syntactic applications. We show that models which require expensive cross-lingual knowledge almost always perform better, but cheaply supervised models often prove competitive on certain tasks.

1 Introduction

Learning word vector representations using monolingual distributional information is now a ubiquitous technique in NLP. The quality of these word vectors can be significantly improved by incorporating cross-lingual distributional information (Klementiev et al., 2012; Zou et al., 2013; Vulić and Moens, 2013b; Mikolov et al., 2013b; Faruqui and Dyer, 2014; Hermann and Blunsom, 2014; Chandar et al., 2014, *inter alia*), with improvements observed both on monolingual (Faruqui and Dyer, 2014; Rastogi et al., 2015) and cross-lingual tasks (Guo et al., 2015; Søgaard et al., 2015; Guo et al., 2016).

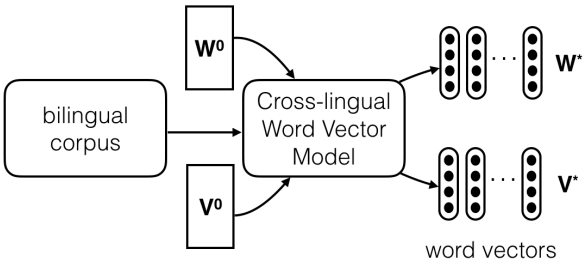
Several models for inducing cross-lingual embeddings have been proposed, each requiring a different form of cross-lingual supervision – some can use document-level alignments (Vulić and Moens, 2015), others need alignments at the sentence (Hermann and Blunsom, 2014; Gouws et al., 2015) or word level (Faruqui and Dyer, 2014; Gouws and Søgaard, 2015), while some require both sentence and word alignments (Luong et al., 2015). However, a systematic comparison of these models is missing from the literature, making it difficult to analyze which approach is suitable for a particular NLP task. In this paper, we fill this void by empirically comparing four cross-lingual word embedding models each of which require different form of alignment(s) as supervision, across several dimensions. To this end, we train these models on four different language pairs, and evaluate them on both monolingual and cross-lingual tasks.[1]

First, we show that different models can be viewed as instances of a more general framework for inducing cross-lingual word embeddings. Then, we evaluate these models on both extrinsic and intrinsic tasks. Our intrinsic evaluation assesses the quality of the vectors on monolingual (§4.2) and cross-lingual (§4.3) word similarity tasks, while our extrinsic evaluation spans semantic (cross-lingual document classification §4.4) and syntactic tasks (cross-lingual dependency parsing §4.5).

Our experiments show that word vectors trained using expensive cross-lingual supervision (word alignments or sentence alignments) perform the best on semantic tasks. On the other hand, for syntactic tasks like cross-lingual dependency parsing, models requiring weaker form of cross-lingual supervision (such as context agnostic translation dictionary) are competitive to models requiring expensive supervision. We also show qualitatively how the nature of cross-lingual supervision used to train word vectors affects the proximity of translation pairs across languages, and of words with similar meaning in the same language in the vector-space.

[1] Instructions and code to reproduce the experiments available at http://cogcomp.cs.illinois.edu/page/publication_view/794

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1661–1670,
Berlin, Germany, August 7-12, 2016.



Algorithm 1 General Algorithm

1: Initialize $\mathbf{W} \leftarrow \mathbf{W}^0$,$\mathbf{V} \leftarrow \mathbf{V}^0$
2: $(\mathbf{W}^*, \mathbf{V}^*) \leftarrow \arg\min \alpha A(\mathbf{W}) + \beta B(\mathbf{V}) + C(\mathbf{W}, \mathbf{V})$

Figure 1: (**Above**) A general schema for induction of cross-lingual word vector representations. The word vector model generates embeddings which incorporates distributional information *cross-lingually*. (**Below**) A general algorithm for inducing bilingual word embeddings, where $\alpha, \beta, \mathbf{W}^0, \mathbf{V}^0$ are parameters and A, B, C are suitably defined losses.

2 Bilingual Embeddings

A general schema for inducing bilingual embeddings is shown in Figure 1. Our comparison focuses on dense, fixed-length distributed embeddings which are obtained using some form of cross-lingual supervision. We briefly describe the embedding induction procedure for each of the selected bilingual word vector models, with the aim to provide a unified algorithmic perspective for all methods, and to facilitate better understanding and comparison. Our choice of models spans across different forms of supervision required for inducing the embeddings, illustrated in Figure 2.

Notation. Let $W = \{w_1, w_2, \ldots, w_{|W|}\}$ be the vocabulary of a language l_1 with $|W|$ words, and $\mathbf{W} \in \mathbb{R}^{|W| \times l}$ be the corresponding word embeddings of length l. Let $V = \{v_1, v_2, \ldots, v_{|V|}\}$ be the vocabulary of another language l_2 with $|V|$ words, and $\mathbf{V} \in \mathbb{R}^{|V| \times m}$ the corresponding word embeddings of length m. We denote the word vector for a word w by $\mathbf{w}$.

2.1 Bilingual Skip-Gram Model (BiSkip)

Luong et al. (2015) proposed Bilingual Skip-Gram, a simple extension of the monolingual skip-gram model, which learns bilingual embeddings by using a parallel corpus along with word alignments (both sentence and word level alignments).

The learning objective is a simple extension of the skip-gram model, where the context of a word is expanded to include bilingual links obtained from word alignments, so that the model is trained to predict words cross-lingually. In particular, given a word alignment link from word $v \in V$ in language l_2 to $w \in W$ in language l_1, the model predicts the context words of w using v and vice-versa. Formally, the cross lingual part of the objective is,

$$D_{12}(\mathbf{W}, \mathbf{V}) = -\sum_{(v,w) \in Q} \sum_{w_c \in \text{NBR}_1(w)} \log P(w_c \mid v) \quad (1)$$

where $\text{NBR}_1(w)$ is the context of w in language l_1, Q is the set of word alignments, and $P(w_c \mid v) \propto \exp(\mathbf{w}_c^T \mathbf{v})$. Another similar term D_{21} models the objective for v and $\text{NBR}_2(v)$. The objective can be cast into Algorithm 1 as,

$$C(\mathbf{W}, \mathbf{V}) = D_{12}(\mathbf{W}, \mathbf{V}) + D_{21}(\mathbf{W}, \mathbf{V}) \quad (2)$$

$$A(\mathbf{W}) = -\sum_{w \in W} \sum_{w_c \in \text{NBR}_1(w)} \log P(w_c \mid w) \quad (3)$$

$$B(\mathbf{V}) = -\sum_{v \in V} \sum_{v_c \in \text{NBR}_2(v)} \log P(v_c \mid v) \quad (4)$$

where $A(\mathbf{W})$ and $B(\mathbf{V})$ are the familiar skip-gram formulation of the monolingual part of the objective. α and β are chosen hyper-parameters which set the relative importance of the monolingual terms.

2.2 Bilingual Compositional Model (BiCVM)

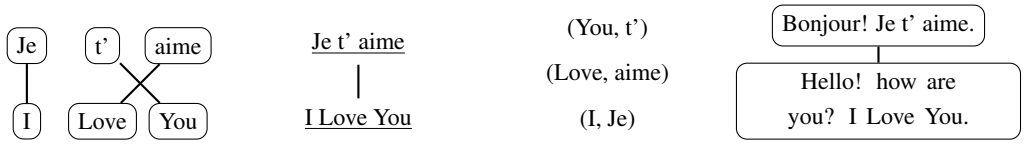
Hermann and Blunsom (2014) present a method that learns bilingual word vectors from a sentence aligned corpus. Their model leverages the fact that aligned sentences have equivalent meaning, thus their sentence representations should be similar.

We denote two aligned sentences, $\vec{v} = \langle \boldsymbol{x}_1, \ldots, \rangle$ and $\vec{w} = \langle \boldsymbol{y}_1, \ldots \rangle$, where $\boldsymbol{x}_i \in \mathbf{V}, \boldsymbol{y}_i \in \mathbf{W}$, are vectors corresponding to the words in the sentences. Let functions $f : \vec{v} \rightarrow \mathbb{R}^n$ and $g : \vec{w} \rightarrow \mathbb{R}^n$, map sentences to their semantic representations in $\mathbb{R}^n$. BiCVM generates word vectors by minimizing the squared ℓ_2 norm between the sentence representations of aligned sentences. In order to prevent the degeneracy arising from directly minimizing the ℓ_2 norm, they use a noise-contrastive large-margin update, with randomly drawn sentence pairs $(\vec{v}, \vec{w}^n)$ as negative samples. The loss for the sentence pairs $(\vec{v}, \vec{w})$ and $(\vec{v}, \vec{w}^n)$ can be written as,

$$E(\vec{v}, \vec{w}, \vec{w}^n) = \max\left(\delta + \Delta E(\vec{v}, \vec{w}, \vec{w}^n), 0\right) \quad (5)$$

where,

$$E(\vec{v}, \vec{w}) = \|f(\vec{v}) - g(\vec{w})\|^2 \quad (6)$$



(a) **BiSkip** (b) **BiCVM** (c) **BiCCA** (d) **BiVCD**

Figure 2: Forms of supervision required by the four models compared in this paper. From left to right, the cost of the supervision required varies from expensive (BiSkip) to cheap (BiVCD). BiSkip requires a parallel corpus annotated with word alignments (Fig. 2a), BiCVM requires a sentence-aligned corpus (Fig. 2b), BiCCA only requires a bilingual lexicon (Fig. 2c) and BiVCD requires comparable documents (Fig. 2d).

and,

$$\Delta E(\vec{v}, \vec{w}, \vec{w}^n) = E(\vec{v}, \vec{w}) - E(\vec{v}, \vec{w}^n) \quad (7)$$

This can be cast into Algorithm 1 by,

$$C(\mathbf{W}, \mathbf{V}) = \sum_{\substack{\text{aligned } (\vec{v},\vec{w}) \\ \text{random } \vec{w}^n}} E(\vec{v}, \vec{w}, \vec{w}^n) \quad (8)$$

$$A(\mathbf{W}) = \|\mathbf{W}\|^2 \quad B(\mathbf{V}) = \|\mathbf{V}\|^2 \quad (9)$$

with $A(\mathbf{W})$ and $B(\mathbf{V})$ being regularizers, with $\alpha = \beta$.

2.3 Bilingual Correlation Based Embeddings (BiCCA)

The BiCCA model, proposed by Faruqui and Dyer (2014), showed that when (independently trained) monolingual vector matrices $\mathbf{W}, \mathbf{V}$ are projected using CCA (Hotelling, 1936) to respect a translation lexicon, their performance improves on word similarity and word analogy tasks. They first construct $\mathbf{W}' \subseteq \mathbf{W}, \mathbf{V}' \subseteq \mathbf{V}$ such that $|\mathbf{W}'| = |\mathbf{V}'|$ and the corresponding words (w_i, v_i) in the matrices are translations of each other. The projection is then computed as:

$$\mathbf{P}_W, \mathbf{P}_V = \text{CCA}(\mathbf{W}', \mathbf{V}') \quad (10)$$

$$\mathbf{W}^* = \mathbf{W}\mathbf{P}_W \quad \mathbf{V}^* = \mathbf{V}\mathbf{P}_V \quad (11)$$

where, $\mathbf{P}_V \in \mathbb{R}^{l \times d}, \mathbf{P}_W \in \mathbb{R}^{m \times d}$ are the projection matrices with $d \leq \min(l, m)$ and the $\mathbf{V}^* \in \mathbb{R}^{|V| \times d}, \mathbf{W}^* \in \mathbb{R}^{|W| \times d}$ are the word vectors that have been "enriched" using bilingual knowledge.

The BiCCA objective can be viewed[2] as the following instantiation of Algorithm 1:

$$\mathbf{W}^0 = \mathbf{W}', \mathbf{V}^0 = \mathbf{V}' \quad (12)$$

$$C(\mathbf{W}, \mathbf{V}) = \|\mathbf{W} - \mathbf{V}\|^2 + \gamma \left(\mathbf{V}^T \mathbf{W}\right) \quad (13)$$

$$A(\mathbf{W}) = \|\mathbf{W}\|^2 - 1 \quad B(\mathbf{V}) = \|\mathbf{V}\|^2 - 1 \quad (14)$$

where $\mathbf{W} = \mathbf{W}^0 \mathbf{P}_W$ and $\mathbf{V} = \mathbf{V}^0 \mathbf{P}_V$, where we set $\alpha = \beta = \gamma = \infty$ to set hard constraints.

[2] described in Section 6.5 of (Hardoon et al., 2004)

2.4 Bilingual Vectors from Comparable Data (BiVCD)

Another approach of inducing bilingual word vectors, which we refer to as BiVCD, was proposed by Vulić and Moens (2015). Their approach is designed to use *comparable* corpus between the source and target language pair to induce cross-lingual vectors.

Let d_e and d_f denote a pair of comparable documents with length in words p and q respectively (assume $p > q$). BiVCD first merges these two comparable documents into a single pseudo-bilingual document using a deterministic strategy based on length ratio of two documents $R = \lfloor \frac{p}{q} \rfloor$. Every R^{th} word of the merged pseudo-bilingual document is picked sequentially from d_f. Finally, a skip-gram model is trained on the corpus of pseudo-bilingual documents, to generate vectors for all words in $\mathbf{W}^* \cup \mathbf{V}^*$. The vectors constituting $\mathbf{W}^*$ and $\mathbf{V}^*$ can then be easily identified.

Instantiating BiVCD in the general algorithm is obvious: $C(\mathbf{W}, \mathbf{V})$ assumes the familiar word2vec skip-gram objective over the pseudo-bilingual document,

$$C(\mathbf{W}, \mathbf{V}) = -\sum_{s \in W \cup V} \sum_{t \in \text{NBR}(s)} \log P(t \mid s) \quad (15)$$

where $\text{NBR}(s)$ is defined by the pseudo-bilingual document and $P(t \mid s) \propto \exp(\mathbf{t}^T \mathbf{s})$. Note that $t, s \in W \cup V$.

Although BiVCD is designed to use comparable corpus, we provide it with parallel data in our experiments (to ensure comparability), and treat two aligned sentences as comparable.

3 Data

We train cross-lingual embeddings for 4 language pairs: English-German (en-de), English-French (en-fr), English-Swedish (en-sv) and English-Chinese (en-zh). For en-de and en-sv we use the

l_1	l_2	#sent	#l_1-words	#l_2-words
en	de	1.9	53	51
	fr	2.0	55	61
	sv	1.7	46	42
	zh	2.0	58	50

Table 1: The size of parallel corpora (in millions) of different language pairs used for training cross-lingual word vectors.

Europarl v7 parallel corpus[3] (Koehn, 2005). For en-fr, we use Europarl combined with the news-commentary and UN-corpus dataset from WMT 2015.[4] For en-zh, we use the FBIS parallel corpus from the news domain (LDC2003E14). We use the Stanford Chinese Segmenter (Tseng et al., 2005) to preprocess the en-zh parallel corpus. Corpus statistics for all languages is shown in Table 1.

4 Evaluation

We measure the quality of the induced cross-lingual word embeddings in terms of their performance, when used as features in the following tasks:

- monolingual word similarity for English
- Cross-lingual dictionary induction
- Cross-lingual document classification
- Cross-lingual syntactic dependency parsing

The first two tasks intrinsically measure how much can monolingual and cross-lingual similarity benefit from cross-lingual training. The last two tasks measure the ability of cross-lingually trained vectors to extrinsically facilitate model transfer across languages, for semantic and syntactic applications respectively. These tasks have been used in previous works (Klementiev et al., 2012; Luong et al., 2015; Vulić and Moens, 2013a; Guo et al., 2015) for evaluating cross-lingual embeddings, but no comparison exists which uses them in conjunction.

To ensure fair comparison, all models are trained with embeddings of size 200. We provide all models with parallel corpora, irrespective of their requirements. Whenever possible, we also report statistical significance of our results.

[3] `www.statmt.org/europarl/v7/{de, sv}-en.tgz`

[4] `www.statmt.org/wmt15/translation-task.html`

4.1 Parameter Selection

We follow the **BestAvg** parameter selection strategy from Lu et al. (2015): we selected the parameters for all models by tuning on a set of values (described below) and picking the parameter setting which did best on an average across all tasks.

BiSkip. All models were trained using a window size of 10 (tuned over $\{5, 10, 20\}$), and 30 negative samples (tuned over $\{10, 20, 30\}$). The cross-lingual weight was set to 4 (tuned over $\{1, 2, 4, 8\}$). The word alignments for training the model (available at `github.com/lmthang/bivec`) were generated using `fast_align` (Dyer et al., 2013). The number of training iterations was set to 5 (no tuning) and we set $\alpha = 1$ and $\beta = 1$ (no tuning).

BiCVM. We use the tool (available at `github.com/karlmoritz/bicvm`) released by Hermann and Blunsom (2014) to train all embeddings. We train an additive model (that is, $f(\vec{x}) = g(\vec{x}) = \sum_i x_i$) with hinge loss margin set to 200 (no tuning), batch size of 50 (tuned over $50, 100, 1000$) and noise parameter of 10 (tuned over $\{10, 20, 30\}$). All models are trained for 100 iterations (no tuning).

BiCCA. First, monolingual word vectors are trained using the skip-gram model[5] with negative sampling (Mikolov et al., 2013a) with window of size 5 (tuned over $\{5, 10, 20\}$). To generate a cross-lingual dictionary, word alignments are generated using cdec from the parallel corpus. Then, word pairs $(a, b), a \in l_1, b \in l_2$ are selected such that a is aligned to b the most number of times and vice versa. This way, we obtained dictionaries of approximately 36k, 35k, 30k and 28k word pairs for en-de, en-fr, en-sv and en-zh respectively.

The monolingual vectors are aligned using the above dictionaries with the tool (available at `github.com/mfaruqui/eacl14-cca`) released by Faruqui and Dyer (2014) to generate the cross-lingual word embeddings. We use $k = 0.5$ as the number of canonical components (tuned over $\{0.2, 0.3, 0.5, 1.0\}$). Note that this results in a embedding of size 100 after performing CCA.

BiVCD. We use word2vec's skip gram model for training our embeddings, with a window size of 5 (tuned on $\{5, 10, 20, 30\}$) and negative sampling parameter set to 5 (tuned on $\{5, 10, 25\}$). Every pair of parallel sentences is treated as a

[5] `code.google.com/p/word2vec`

pair of comparable documents, and merging is performed using the sentence length ratio strategy described earlier.[6]

4.2 Monolingual Evaluation

We first evaluate if the inclusion of cross-lingual knowledge improves the quality of English embeddings.

Word Similarity. Word similarity datasets contain word pairs which are assigned similarity ratings by humans. The task evaluates how well the notion of word similarity according to humans is emulated in the vector space. Evaluation is based on the Spearman's rank correlation coefficient (Myers and Well, 1995) between human rankings and rankings produced by computing cosine similarity between the vectors of two words.

We use the SimLex dataset for English (Hill et al., 2014) which contains 999 pairs of English words, with a balanced set of noun, adjective and verb pairs. SimLex is claimed to capture word similarity exclusively instead of WordSim-353 (Finkelstein et al., 2001) which captures both word similarity and relatedness. We declare significant improvement if $p < 0.1$ according to Steiger's method (Steiger, 1980) for calculating the statistical significant differences between two dependent correlation coefficients.

Table 2 shows the performance of English embeddings induced by all the models by training on different language pairs on the SimLex word similarity task. The score obtained by monolingual English embeddings trained on the respective English side of each language is shown in column marked Mono. In all cases (except BiCCA on en-sv), the bilingually trained vectors achieve better scores than the mono-lingually trained vectors.

Overall, across all language pairs, BiCVM is the best performing model in terms of Spearman's correlation, but its improvement over BiSkip and BiVCD is often insignificant. It is notable that 2 of the 3 top performing models, BiCVM and BiVCD, need sentence aligned and document-aligned corpus only, which are easier to obtain than parallel data with word alignments required by BiSkip.

QVEC. Tsvetkov et al. (2015) proposed an intrinsic evaluation metric for estimating the quality of English word vectors. The score produced by QVEC measures how well a given set of word vectors is able to quantify linguistic properties of words, with higher being better. The metric is shown to have strong correlation with performance on downstream semantic applications. As it can be currently only used for English, we use it to evaluate the English vectors obtained using cross-lingual training of different models. Table 3 shows that on average across language pairs, BiSkip achieves the best score, followed by Mono (mono-lingually trained English vectors), BiVCD and BiCCA. A possible explanation for why Mono scores are better than those obtained by some of the cross-lingual models is that QVEC measures monolingual semantic content based on a linguistic oracle made for English. Cross-lingual training might affect these semantic properties arbitrarily.

Interestingly, BiCVM which was the best model according to SimLex, ranks last according to QVEC. The fact that the best models according to QVEC and word similarities are different reinforces observations made in previous work that performance on word similarity tasks alone does not reflect quantification of linguistic properties of words (Tsvetkov et al., 2015; Schnabel et al., 2015).

[6]We implemented the code for performing the merging as we could not find a tool provided by the authors.

pair	Mono	BiSkip	BiCVM	BiCCA	BiVCD
en-de	0.29	0.34	**0.37**	0.30	0.32
en-fr	0.30	0.35	**0.39**	0.31	0.36
en-sv	0.28	0.32	**0.34**	0.27	0.32
en-zh	0.28	0.34	**0.39**	0.30	0.31
avg.	0.29	0.34	**0.37**	0.30	0.33

Table 2: Word similarity score measured in Spearman's correlation ratio for English on SimLex-999. The best score for each language pair is shown in **bold**. Scores which are significantly better (per Steiger's Method with $p < 0.1$) than the next lower score are underlined. For example, for en-zh, BiCVM is significantly better than BiSkip, which in turn is significantly better than BiVCD.

pair	Mono	BiSkip	BiCVM	BiCCA	BiVCD
en-de	0.39	**0.40**	0.31	0.33	0.37
en-fr	0.39	**0.40**	0.31	0.33	0.38
en-sv	0.39	**0.39**	0.31	0.32	0.37
en-zh	0.40	**0.40**	0.32	0.33	0.38
avg.	0.39	**0.40**	0.31	0.33	0.38

Table 3: Intrinsic evaluation of English word vectors measured in terms of QVEC score across models. Best scores for each language pair is shown in **bold**.

l_1	l_2	BiSkip	BiCVM	BiCCA	BiVCD
en	de	**79.7**	74.5	72.4	62.5
	fr	**78.9**	72.9	70.1	68.8
	sv	**77.1**	76.7	74.2	56.9
	zh	**69.4**	66.0	59.6	53.2
avg.		**76.3**	72.5	69.1	60.4

Table 4: Cross-lingual dictionary induction results (top-10 accuracy). The same trend was also observed across models when computing MRR (mean reciprocal rank).

l_1	l_2	BiSkip	BiCVM	BiCCA	BiVCD
en	de	**85.2**	85.0	79.1	79.9
	fr	**77.7**	71.7	70.7	72.0
	sv	**72.3**	69.1	65.3	59.9
	zh	**75.5**	73.6	69.4	73.0
de	en	**74.9**	71.1	64.9	74.1
fr		**80.4**	73.7	75.5	77.6
sv		73.4	67.7	67.0	**78.2**
zh		**81.1**	76.4	77.3	80.9
avg.		**77.6**	73.5	71.2	74.5

Table 5: Cross-lingual document classification accuracy when trained on language l_1, and evaluated on language l_2. The best score for each language is shown in **bold**. Scores which are significantly better (per McNemar's Test with $p < 0.05$) than the next lower score are underlined. For example, for sv→en, BiVCD is significantly better than BiSkip, which in turn is significantly better than BiCVM.

4.3 Cross-lingual Dictionary Induction

The task of cross-lingual dictionary induction (Vulić and Moens, 2013a; Gouws et al., 2015; Mikolov et al., 2013b) judges how good cross-lingual embeddings are at detecting word pairs that are semantically similar across languages. We follow the setup of Vulić and Moens (2013a), but instead of manually creating a gold cross-lingual dictionary, we derived our gold dictionaries using the Open Multilingual WordNet data released by Bond and Foster (2013). The data includes synset alignments across 26 languages with over 90% accuracy. First, we prune out words from each synset whose frequency count is less than 1000 in the vocabulary of the training data from §3. Then, for each pair of aligned synsets $s_1 = \{k_1, k_2, \cdots\}$ $s_2 = \{g_1, g_2, \cdots\}$, we include all elements from the set $\{(k, g) \mid k \in s_1, g \in s_2\}$ into the gold dictionary, where k and g are the lemmas. Using this approach we generated dictionaries of sizes 1.5k, 1.4k, 1.0k and 1.6k pairs for en-fr, en-de, en-sv and en-zh respectively.

We report top-10 accuracy, which is the fraction of the entries (e, f) in the gold dictionary, for which f belongs to the list of top-10 neighbors of the word vector of e, according to the induced cross-lingual embeddings. From the results (Table 4), it can be seen that for dictionary induction, the performance improves with the quality of supervision. As we move from cheaply supervised methods (eg. BiVCD) to more expensive supervision (eg. BiSkip), the accuracy improves. This suggests that for cross lingual similarity tasks, the more expensive the cross-lingual knowledge available, the better. Models using weak supervision like BiVCD perform poorly in comparison to models like BiSkip and BiCVM, with performance gaps upwards of 10 pts on an average.

4.4 Cross-lingual Document Classification

We follow the cross-lingual document classification (CLDC) setup of Klementiev et al. (2012), but extend it to cover all of our language pairs. We use the RCV2 Reuters multilingual corpus[7] for our experiments. In this task, for a language pair (l_1, l_2), a document classifier is trained using the document representations derived from word embeddings in language l_1, and then the trained model is tested on documents from language l_2 (and vice-versa). By using supervised training data in one language and evaluating without further supervision in another, CLDC assesses whether the learned cross-lingual representations are semantically coherent across multiple languages.

All embeddings are learned on the data described in §3, and we only use the RCV2 data to learn document classification models. Following previous work, we compute document representation by taking the tf-idf weighted average of vectors of the words present in it.[8] A multi-class classifier is trained using an averaged perceptron (Freund and Schapire, 1999) for 10 iterations, using the document vectors of language l_1 as features[9]. Majority baselines for $en \rightarrow l_2$ and $l_1 \rightarrow en$ are 49.7% and 46.7% respectively, for all languages. Table 5 shows the performance of different models across different language pairs. We computed confidence values using the McNemar test (McNe-

[7] http://trec.nist.gov/data/reuters/reuters.html

[8] tf-idf (Salton and Buckley, 1988) was computed using all documents for that language in RCV2.

[9] We use the implementation of Klementiev et al. (2012).

mar, 1947) and declare significant improvement if $p < 0.05$.

Table 5 shows that in almost all cases, BiSkip performs significantly better than the remaining models. For transferring semantic knowledge across languages via embeddings, sentence and word level alignment proves superior to sentence or word level alignment alone. This observation is consistent with the trend in cross-lingual dictionary induction, where too the most expensive form of supervision performed the best.

4.5 Cross-lingual Dependency Parsing

Using cross lingual similarity for direct-transfer of dependency parsers was first shown in Täckström et al. (2012). The idea behind direct-transfer is to train a dependency parsing model using embeddings for language l_1 and then test the trained model on language l_2, replacing embeddings for language l_1 with those of l_2. The transfer relies on coherence of the embeddings across languages arising from the cross lingual training. For our experiments, we use the cross lingual transfer setup of Guo et al. (2015).[10] Their framework trains a transition-based dependency parser using nonlinear activation function, with the source-side embeddings as lexical features. These embeddings can be replaced by target-side embeddings at test time.

All models are trained for 5000 iterations with fixed word embeddings during training. Since our goal is to determine the utility of word embeddings in dependency parsing, we turn off other features that can capture distributional information like brown clusters, which were originally used in Guo et al. (2015). We use the universal dependency treebank (McDonald et al., 2013) version-2.0 for our evaluation. For Chinese, we use the treebank released as part of the CoNLL-X shared task (Buchholz and Marsi, 2006).

We first evaluate how useful the word embeddings are in cross-lingual model transfer of dependency parsers (Table 6). On an average, BiCCA does better than other models. BiSkip is a close second, with an average performance gap of less than 1 point. BiSkip outperforms BiCVM on German and French (over 2 point improvement), owing to word alignment information BiSkip's model uses during training. It is not surprising that English-Chinese transfer scores are low, due to the significant difference in syntactic structure of the two languages. Surprisingly, unlike the semantic tasks considered earlier, the models with expensive supervision requirements like BiSkip and BiCVM could not outperform a cheaply supervised BiCCA.

[10] github.com/jiangfeng1124/acl15-clnndep

l_1	l_2	BiSkip	BiCVM	BiCCA	BiVCD
en	de	49.8	47.5	**51.3**	49.0
	fr	65.8	63.2	**65.9**	60.7
	sv	56.9	56.7	**59.4**	54.6
	zh	**6.4**	6.1	**6.4**	6.0
de	en	49.7	45.0	**50.3**	43.6
fr		53.3	50.6	**54.2**	49.5
sv		48.2	49.0	**49.9**	44.6
zh		**0.17**	0.12	**0.17**	0.15
avg.		41.3	39.8	**42.2**	38.5

Table 6: Labeled attachment score (LAS) for cross-lingual dependency parsing when trained on language l_1, and evaluated on language l_2. The best score for each language is shown in **bold**.

We also evaluate whether using cross-lingually trained vectors for learning dependency parsers is better than using mono-lingually trained vectors in Table 7. We compare against parsing models trained using mono-lingually trained word vectors (column marked Mono in Table 7). These vectors are the same used as input to the BiCCA model. All other settings remain the same. On an average across language pairs, improvement over the monolingual embeddings was obtained with the BiSkip and BiCCA models, while BiCVM and BiVCD consistently performed worse. A possible reason for this is that BiCVM and BiVCD operate on sentence level contexts to learn the embeddings, which only captures the semantic meaning of the sentences and ignores the internal syntactic structure. As a result, embedding trained using BiCVM and BiVCD are not informative for syntactic tasks. On the other hand, BiSkip and BiCCA both utilize the word alignment information to train their embeddings and thus do better in capturing some notion of syntax.

5 Qualitative Analysis

Figure 3 shows the PCA projection of some of the most frequent words in the English-French corpus. It is clear that BiSkip and BiCVM produce cross-lingual vectors which are the most comparable, the English and French words which are translations of each other are represented by almost the same point in the vector-space. In BiCCA and BiVCD

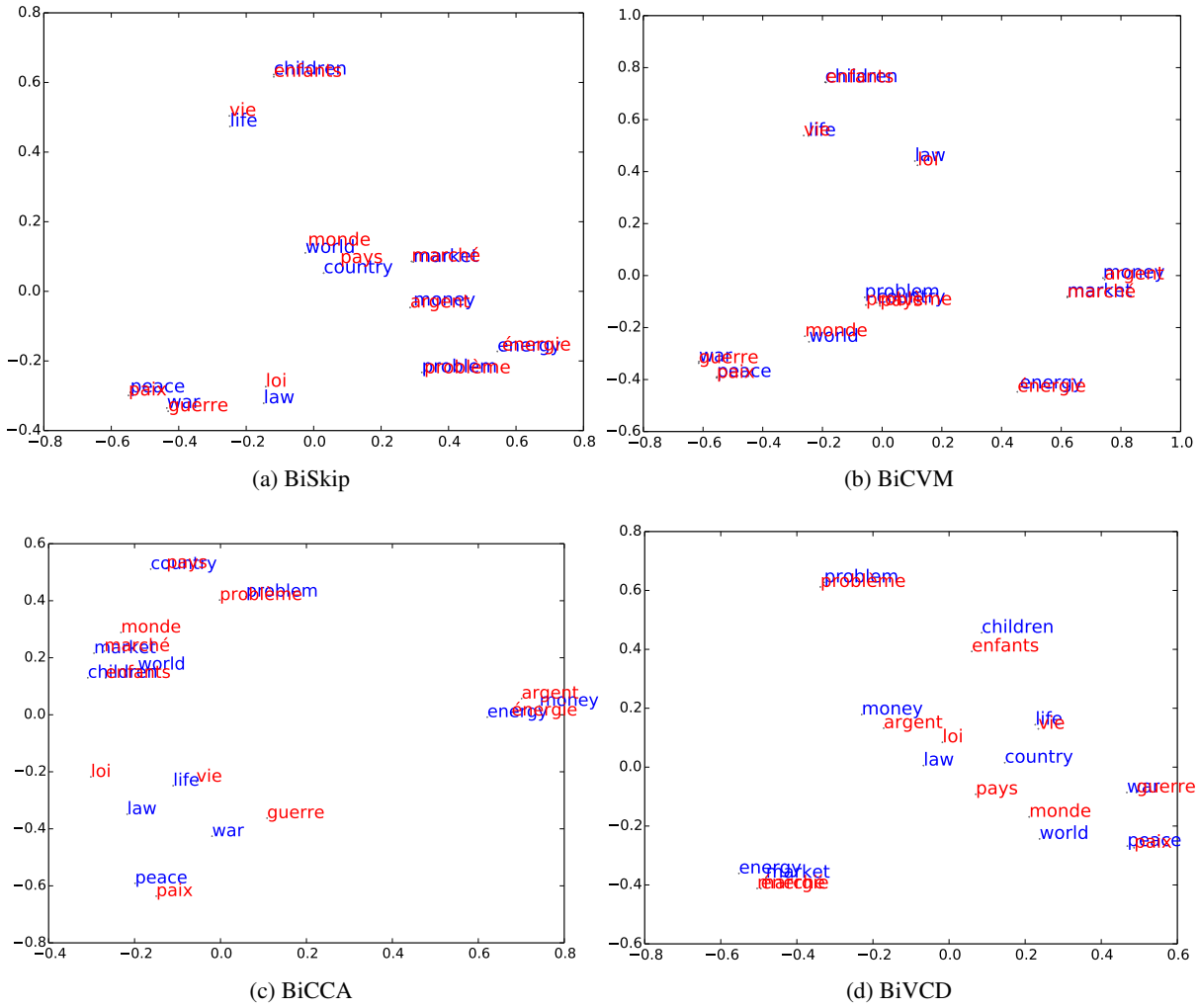


Figure 3: PCA projection of word embeddings of some frequent words present in English-French corpus. English and French words are shown in blue and red respectively.

l	Mono	BiSkip	BiCVM	BiCCA	BiVCD
de	71.1	**72.0**	60.4	**71.4**	58.9
fr	78.9	**80.4**	73.7	**80.2**	69.5
sv	75.5	**78.2**	70.5	**79.0**	64.5
zh	73.8	73.1	65.8	71.7	67.0
avg.	74.8	**75.9**	67.6	**75.6**	66.8

Table 7: Labeled attachment score (LAS) for dependency parsing when trained and tested on language l. Mono refers to parser trained with mono-lingually induced embeddings. Scores in **bold** are better than the Mono scores for each language, showing improvement from cross-lingual training.

the translated words are more distant than BiSkip and BiCVM. This is not surprising because BiSkip and BiCVM require more expensive supervision at the sentence level in contrast to the other two models.

An interesting observation is that BiCCA and BiVCD are better at separating antonyms. The words *peace* and *war*, (and their French translations *paix* and *guerre*) are well separated in BiCCA and BiVCD. However, in BiSkip and BiCVM these pairs are very close together. This can be attributed to the fact that BiSkip and BiCVM are trained on parallel sentences, and if two antonyms are present in the same sentence in English, they will also be present together in its French translation. However, BiCCA uses bilingual dictionary and BiVCD use comparable sentence context, which helps in pulling apart the synonyms and antonyms.

6 Discussion

The goal of this paper was to formulate the task of learning cross-lingual word vector representations in a unified framework, and conduct experiments to compare the performance of existing

models in a unbiased manner. We chose existing cross-lingual word vector models that can be trained on two languages at a given time. In recent work, Ammar et al. (2016) train multilingual word vectors using more than two languages; our comparison does not cover this setting. It is also worth noting that we compare here different cross-lingual word embeddings, which are not to be confused with a collection of monolingual word embeddings trained for different languages individually (Al-Rfou et al., 2013).

The paper does not cover *all* approaches that generate cross-lingual word embeddings. Some methods do not have publicly available code (Coulmance et al., 2015; Zou et al., 2013); for others, like BilBOWA (Gouws et al., 2015), we identified problems in the available code, which caused it to consistently produced results that are inferior even to mono-lingually trained vectors.[11] However, the models that we included for comparison in our survey are representative of other cross-lingual models in terms of the form of cross-lingual supervision required by them. For example, BilBOWA (Gouws et al., 2015) and cross-lingual Auto-encoder (Chandar et al., 2014) are similar to BiCVM in this respect. Multi-view CCA (Rastogi et al., 2015) and deep CCA (Lu et al., 2015) can be viewed as extensions of BiCCA. Our choice of models was motivated to compare different forms of supervision, and therefore, adding these models, would not provide additional insight.

7 Conclusion

We presented the first systematic comparative evaluation of cross-lingual embedding methods on several downstream NLP tasks, both intrinsic and extrinsic. We provided a unified representation for all approaches, showing them as instances of a general algorithm. Our choice of methods spans a diverse range of approaches, in that each requires a different form of supervision.

Our experiments reveal interesting trends. When evaluating on intrinsic tasks such as monolingual word similarity, models relying on cheaper forms of supervision (such as BiVCD) perform almost on par with models requiring expensive supervision. On the other hand, for cross-lingual semantic tasks, like cross-lingual document classification and dictionary induction, the model with the most informative supervision performs best overall. In contrast, for the syntactic task of dependency parsing, models that are supervised at a word alignment level perform slightly better. Overall this suggests that semantic tasks can benefit more from richer cross-lingual supervision, as compared to syntactic tasks.

[11] We contacted the authors of the papers and were unable to resolve the issues in the toolkit.

Acknowledgement

This material is based on research sponsored by DARPA under agreement number FA8750-13-2-0008 and Contract HR0011-15-2-0025. Approved for Public Release, Distribution Unlimited. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual nlp. In *Proc. of CoNLL*.

Waleed Ammar, George Mulcaire, Yulia Tsvetkov, Guillaume Lample, Chris Dyer, and Noah A Smith. 2016. Massively multilingual word embeddings. *arXiv preprint arXiv:1602.01925*.

Francis Bond and Ryan Foster. 2013. Linking and extending an open multilingual wordnet. In *Proc. of ACL*.

Sabine Buchholz and Erwin Marsi. 2006. Conll-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. 2014. An autoencoder approach to learning bilingual word representations. In *Proc. of NIPS*.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Trans-gram, fast cross-lingual word-embeddings. In *Proc. of EMNLP*.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of ibm model 2. In *Proc. of NAACL*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of EACL*.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. In *Proc. of WWW*.

Yoav Freund and Robert E Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine learning*, 37(3):277–296.

Stephan Gouws and Anders Søgaard. 2015. Simple task-specific bilingual word embeddings. In *Proc. of NAACL*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proc. of ICML*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proc. of ACL*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2016. A representation learning framework for multi-source transfer parsing. In *Proc. of AAAI*.

David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Models for Compositional Distributional Semantics. In *Proc. of ACL*.

Felix Hill, Roi Reichart, and Anna Korhonen. 2014. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *arXiv preprint arXiv:1408.3456*.

Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. In *Proc. of COLING*.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proc. of MT Summit*.

Ang Lu, Weiran Wang, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. Deep multilingual correlation for improved word embeddings. In *Proc. of NAACL*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual word representations with monolingual quality in mind. In *Proc. of the Workshop on Vector Space Modeling for NLP*.

Ryan McDonald, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee. 2013. Universal dependency annotation for multilingual parsing. In *Proc. of ACL*.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

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

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013b. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*.

Jerome L. Myers and Arnold D. Well. 1995. *Research Design & Statistical Analysis*. Routledge.

Pushpendre Rastogi, Benjamin Van Durme, and Raman Arora. 2015. Multiview LSA: Representation learning via generalized CCA. In *Proceedings of NAACL*.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. 2015. Evaluation methods for unsupervised word embeddings. In *Proc. of EMNLP*.

Anders Søgaard, Željko Agić, Héctor Martínez Alonso, Barbara Plank, Bernd Bohnet, and Anders Johannsen. 2015. Inverted indexing for cross-lingual nlp. In *Proc. of ACL*.

James H Steiger. 1980. Tests for comparing elements of a correlation matrix. *Psychological bulletin*, 87(2):245.

O. Täckström, R. McDonald, and J. Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *NAACL*.

Huihsin Tseng, Pichuan Chang, Galen Andrew, Daniel Jurafsky, and Christopher Manning. 2005. A conditional random field word segmenter for sighan bakeoff 2005. In *Proc. of SIGHAN*.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proc. of EMNLP*.

Ivan Vulić and Marie-Francine Moens. 2013a. Cross-lingual semantic similarity of words as the similarity of their semantic word responses. In *Proc. of NAACL*.

Ivan Vulić and Marie-Francine Moens. 2013b. A study on bootstrapping bilingual vector spaces from non-parallel data (and nothing else). In *Proc. of EMNLP*.

Ivan Vulić and Marie-Francine Moens. 2015. Bilingual word embeddings from non-parallel document-aligned data applied to bilingual lexicon induction. In *Proc. of ACL*.

Will Y. Zou, Richard Socher, Daniel Cer, and Christopher D. Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proc. of EMNLP*.

Take and *Took*, *Gaggle* and *Goose*, *Book* and *Read*: Evaluating the Utility of Vector Differences for Lexical Relation Learning

Ekaterina Vylomova,[1] **Laura Rimell,**[2] **Trevor Cohn,**[1] and **Timothy Baldwin**[1]
[1]Department of Computing and Information Systems, University of Melbourne
[2]Computer Laboratory, University of Cambridge
evylomova@gmail.com laura.rimell@cl.cam.ac.uk {tcohn,tbaldwin}@unimelb.edu.au

Abstract

Recent work has shown that simple vector subtraction over word embeddings is surprisingly effective at capturing different lexical relations, despite lacking explicit supervision. Prior work has evaluated this intriguing result using a word analogy prediction formulation and hand-selected relations, but the generality of the finding over a broader range of lexical relation types and different learning settings has not been evaluated. In this paper, we carry out such an evaluation in two learning settings: (1) spectral clustering to induce word relations, and (2) supervised learning to classify vector differences into relation types. We find that word embeddings capture a surprising amount of information, and that, under suitable supervised training, vector subtraction generalises well to a broad range of relations, including over unseen lexical items.

1 Introduction

Learning to identify lexical relations is a fundamental task in natural language processing ("NLP"), and can contribute to many NLP applications including paraphrasing and generation, machine translation, and ontology building (Banko et al., 2007; Hendrickx et al., 2010).

Recently, attention has been focused on identifying lexical relations using word embeddings, which are dense, low-dimensional vectors obtained either from a "predict-based" neural network trained to predict word contexts, or a "count-based" traditional distributional similarity method combined with dimensionality reduction. The skip-gram model of Mikolov et al. (2013a) and other similar language models have been shown to perform well on an analogy completion task (Mikolov et al., 2013b; Mikolov et al., 2013c; Levy and Goldberg, 2014a), in the space of *relational similarity* prediction (Turney, 2006), where the task is to predict the missing word in analogies such as *A*:*B* :: *C*:*–?–*. A well-known example involves predicting the vector $\mathbf{queen}$ from the vector combination $\mathbf{king} - \mathbf{man} + \mathbf{woman}$, where linear operations on word vectors appear to capture the lexical relation governing the analogy, in this case OPPOSITE-GENDER. The results extend to several semantic relations such as CAPITAL-OF ($\mathbf{paris} - \mathbf{france} + \mathbf{poland} \approx \mathbf{warsaw}$) and morphosyntactic relations such as PLURALISATION ($\mathbf{cars} - \mathbf{car} + \mathbf{apple} \approx \mathbf{apples}$). Remarkably, since the model is not trained for this task, the relational structure of the vector space appears to be an emergent property.

The key operation in these models is *vector difference*, or *vector offset*. For example, the $\mathbf{paris} - \mathbf{france}$ vector appears to encode CAPITAL-OF, presumably by cancelling out the features of $\mathbf{paris}$ that are France-specific, and retaining the features that distinguish a capital city (Levy and Goldberg, 2014a). The success of the simple offset method on analogy completion suggests that the difference vectors ("DIFFVEC" hereafter) must themselves be meaningful: their direction and/or magnitude encodes a lexical relation.

Previous analogy completion tasks used with word embeddings have limited coverage of lexical relation types. Moreover, the task does not explore the full implications of DIFFVECs as meaningful vector space objects in their own right, because it only looks for a one-best answer to the particular lexical analogies in the test set. In this paper, we introduce a new, larger dataset covering many well-known lexical relation types from the linguistics and cognitive science literature. We then apply DIFFVECs to two new tasks: unsupervised and supervised relation extraction. First, we cluster the DIFFVECs to test whether the clusters map onto true lexical relations. We find that the clustering

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1671–1682,
Berlin, Germany, August 7-12, 2016.

works remarkably well, although syntactic relations are captured better than semantic ones.

Second, we perform classification over the DIFFVECs and obtain remarkably high accuracy in a closed-world setting (over a predefined set of word pairs, each of which corresponds to a lexical relation in the training data). When we move to an open-world setting including random word pairs — many of which do not correspond to any lexical relation in the training data — the results are poor. We then investigate methods for better attuning the learned class representation to the lexical relations, focusing on methods for automatically synthesising negative instances. We find that this improves the model performance substantially.

We also find that hyper-parameter optimised count-based methods are competitive with predict-based methods under both clustering and supervised relation classification, in line with the findings of Levy et al. (2015a).

2 Background and Related Work

A lexical relation is a binary relation r holding between a word pair (w_i, w_j); for example, the pair $(cart, wheel)$ stands in the WHOLE-PART relation. Relation learning in NLP includes relation extraction, relation classification, and relational similarity prediction. In relation extraction, related word pairs in a corpus and the relevant relation are identified. Given a word pair, the relation classification task involves assigning a word pair to the correct relation from a pre-defined set. In the Open Information Extraction paradigm (Banko et al., 2007; Weikum and Theobald, 2010), also known as unsupervised relation extraction, the relations themselves are also learned from the text (e.g. in the form of text labels). On the other hand, relational similarity prediction involves assessing the degree to which a word pair (A, B) stands in the same relation as another pair (C, D), or to complete an analogy $A{:}B :: C{:}{-?-}$. Relation learning is an important and long-standing task in NLP and has been the focus of a number of shared tasks (Girju et al., 2007; Hendrickx et al., 2010; Jurgens et al., 2012).

Recently, attention has turned to using vector space models of words for relation classification and relational similarity prediction. Distributional word vectors have been used for detection of relations such as hypernymy (Geffet and Dagan, 2005; Kotlerman et al., 2010; Lenci and Benotto, 2012; Weeds et al., 2014; Rimell, 2014; Santus et al., 2014) and qualia structure (Yamada et al., 2009). An exciting development, and the inspiration for this paper, has been the demonstration that vector difference over word embeddings (Mikolov et al., 2013c) can be used to model word analogy tasks. This has given rise to a series of papers exploring the DIFFVEC idea in different contexts. The original analogy dataset has been used to evaluate predict-based language models by Mnih and Kavukcuoglu (2013) and also Zhila et al. (2013), who combine a neural language model with a pattern-based classifier. Kim and de Marneffe (2013) use word embeddings to derive representations of adjective scales, e.g. *hot—warm—cool—cold*. Fu et al. (2014) similarly use embeddings to predict hypernym relations, in this case clustering words by topic to show that hypernym DIFFVECs can be broken down into more fine-grained relations. Neural networks have also been developed for joint learning of lexical and relational similarity, making use of the WordNet relation hierarchy (Bordes et al., 2013; Socher et al., 2013; Xu et al., 2014; Yu and Dredze, 2014; Faruqui et al., 2015; Fried and Duh, 2015).

Another strand of work responding to the vector difference approach has analysed the structure of predict-based embedding models in order to help explain their success on the analogy and other tasks (Levy and Goldberg, 2014a; Levy and Goldberg, 2014b; Arora et al., 2015). However, there has been no systematic investigation of the range of relations for which the vector difference method is most effective, although there have been some smaller-scale investigations in this direction. Makrai et al. (2013) divide antonym pairs into semantic classes such as quality, time, gender, and distance, finding that for about two-thirds of antonym classes, DIFFVECs are significantly more correlated than random. Necşulescu et al. (2015) train a classifier on word pairs, using word embeddings to predict coordinates, hypernyms, and meronyms. Roller and Erk (2016) analyse the performance of vector concatenation and difference on the task of predicting lexical entailment and show that vector concatenation overwhelmingly learns to detect Hearst patterns (e.g., *including*, *such as*). Köper et al. (2015) undertake a systematic study of morphosyntactic and semantic relations on word embeddings produced with `word2vec` ("`w2v`" hereafter; see §3.1) for English and German. They test a variety of relations including word similarity, antonyms, synonyms, hypernyms, and meronyms, in a novel analogy task. Although the set of relations tested by

Köper et al. (2015) is somewhat more constrained than the set we use, there is a good deal of overlap. However, their evaluation is performed in the context of relational similarity, and they do not perform clustering or classification on the DIFFVECs.

3 General Approach and Resources

We define the task of lexical relation learning to take a set of (ordered) word pairs $\{(w_i, w_j)\}$ and a set of binary lexical relations $R = \{r_k\}$, and map each word pair (w_i, w_j) as follows: (a) $(w_i, w_j) \mapsto r_k \in R$, i.e. the "closed-world" setting, where we assume that all word pairs can be uniquely classified according to a relation in R; or (b) $(w_i, w_j) \mapsto r_k \in R \cup \{\phi\}$ where ϕ signifies the fact that none of the relations in R apply to the word pair in question, i.e. the "open-world" setting.

Our starting point for lexical relation learning is the assumption that important information about various types of relations is implicitly embedded in the offset vectors. While a range of methods have been proposed for composing word vectors (Baroni et al., 2012; Weeds et al., 2014; Roller et al., 2014), in this research we focus exclusively on DIFFVEC (i.e. $\mathbf{w_2} - \mathbf{w_1}$). A second assumption is that there exist dimensions, or directions, in the embedding vector spaces responsible for a particular lexical relation. Such dimensions could be identified and exploited as part of a clustering or classification method, in the context of identifying relations between word pairs or classes of DIFFVECs.

In order to test the generalisability of the DIFFVEC method, we require: (1) word embeddings, and (2) a set of lexical relations to evaluate against. As the focus of this paper is not the word embedding pre-training approaches so much as the utility of the DIFFVECs for lexical relation learning, we take a selection of four pre-trained word embeddings with strong currency in the literature, as detailed in §3.1. We also include the state-of-the-art count-based approach of Levy et al. (2015a), to test the generalisability of DIFFVECs to count-based word embeddings.

For the lexical relations, we want a range of relations that is representative of the types of relational learning tasks targeted in the literature, and where there is availability of annotated data. To this end, we construct a dataset from a variety of sources, focusing on lexical semantic relations (which are less well represented in the analogy dataset of Mikolov et al. (2013c)), but also including morphosyntactic and morphosemantic relations (see §3.2).

Name	Dimensions	Training data
w2v	300	100×10^9
GloVe	200	6×10^9
SENNA	100	37×10^6
HLBL	200	37×10^6
w2v_{wiki}	300	50×10^6
$\text{GloVe}_{\text{wiki}}$	300	50×10^6
SVD_{wiki}	300	50×10^6

Table 1: The pre-trained word embeddings used in our experiments, with the number of dimensions and size of the training data (in word tokens). The models trained on English Wikipedia ("wiki") are in the lower half of the table.

3.1 Word Embeddings

We consider four highly successful word embedding models in our experiments: w2v (Mikolov et al., 2013a; Mikolov et al., 2013b), GloVe (Pennington et al., 2014), SENNA (Collobert and Weston, 2008), and HLBL (Mnih and Hinton, 2009), as detailed below. We also include SVD (Levy et al., 2015a), a count-based model which factorises a positive PMI (PPMI) matrix. For consistency of comparison, we train SVD as well as a version of w2v and GloVe (which we call w2v_{wiki} and $\text{GloVe}_{\text{wiki}}$, respectively) on the English Wikipedia corpus (comparable in size to the training data of SENNA and HLBL), and apply the preprocessing of Levy et al. (2015a). We additionally normalise the w2v_{wiki} and SVD_{wiki} vectors to unit length; $\text{GloVe}_{\text{wiki}}$ is natively normalised by column.[1]

w2v CBOW (Continuous Bag-Of-Words; Mikolov et al. (2013a)) predicts a word from its context using a model with the objective:

$$J = \frac{1}{T}\sum_{i=1}^{T} \log \frac{\exp\left(\mathbf{w}_i^\top \sum\limits_{j\in[-c,+c],j\neq 0} \tilde{\mathbf{w}}_{i+j}\right)}{\sum_{k=1}^{V} \exp\left(\mathbf{w}_k^\top \sum\limits_{j\in[-c,+c],j\neq 0} \tilde{\mathbf{w}}_{i+j}\right)}$$

where $\mathbf{w}_i$ and $\tilde{\mathbf{w}}_i$ are the vector representations for the ith word (as a focus or context word, respectively), V is the vocabulary size, T is the number of tokens in the corpus, and c is the context window size.[2] Google News data was used

[1] We ran a series of experiments on normalised and unnormalised w2v models, and found that normalisation tends to boost results over most of our relations (with the exception of $\text{LEXSEM}_{\text{Event}}$ and $\text{NOUN}_{\text{Coll}}$). We leave a more detailed investigation of normalisation to future work.

[2] In a slight abuse of notation, the subscripts of $\mathbf{w}$ do double

Relation	Description	Pairs	Source	Example
LEXSEM$_{\text{Hyper}}$	hypernym	1173	SemEval'12 + BLESS	(*animal*, *dog*)
LEXSEM$_{\text{Mero}}$	meronym	2825	SemEval'12 + BLESS	(*airplane*, *cockpit*)
LEXSEM$_{\text{Attr}}$	characteristic quality, action	71	SemEval'12	(*cloud*, *rain*)
LEXSEM$_{\text{Cause}}$	cause, purpose, or goal	249	SemEval'12	(*cook*, *eat*)
LEXSEM$_{\text{Space}}$	location or time association	235	SemEval'12	(*aquarium*, *fish*)
LEXSEM$_{\text{Ref}}$	expression or representation	187	SemEval'12	(*song*, *emotion*)
LEXSEM$_{\text{Event}}$	object's action	3583	BLESS	(*zip*, *coat*)
NOUN$_{\text{SP}}$	plural form of a noun	100	MSR	(*year*, *years*)
VERB$_3$	first to third person verb present-tense form	99	MSR	(*accept*, *accepts*)
VERB$_{\text{Past}}$	present-tense to past-tense verb form	100	MSR	(*know*, *knew*)
VERB$_{\text{3Past}}$	third person present-tense to past-tense verb form	100	MSR	(*creates*, *created*)
LVC	light verb construction	58	Tan et al. (2006b)	(*give*, *approval*)
VERBNOUN	nominalisation of a verb	3303	WordNet	(*approve*, *approval*)
PREFIX	prefixing with *re* morpheme	118	Wiktionary	(*vote*, *revote*)
NOUN$_{\text{Coll}}$	collective noun	257	Web source	(*army*, *ants*)

Table 2: Description of the 15 lexical relations.

to train the model. We use the focus word vectors, $W = \{\mathbf{w}_k\}_{k=1}^{V}$, normalised such that each $\|\mathbf{w}_k\| = 1$.

The GloVe model (Pennington et al., 2014) is based on a similar bilinear formulation, framed as a low-rank decomposition of the matrix of corpus co-occurrence frequencies:

$$J = \frac{1}{2}\sum_{i,j=1}^{V} f(P_{ij})(\mathbf{w}_i^\top \tilde{\mathbf{w}}_j - \log P_{ij})^2 \,,$$

where w_i is a vector for the left context, w_j is a vector for the right context, P_{ij} is the relative frequency of word j in the context of word i, and f is a heuristic weighting function to balance the influence of high versus low term frequencies. The model was trained on English Wikipedia and the English Gigaword corpus version 5.

The SVD model (Levy et al., 2015a) uses positive pointwise mutual information (PMI) matrix defined as:

$$\text{PPMI}(w, c) = \max(\log \frac{\hat{P}(w, c)}{\hat{P}(w)\hat{P}(c)}, 0) \,,$$

where $\hat{P}(w, c)$ is the joint probability of word w and context c, and $\hat{P}(w)$ and $\hat{P}(c)$ are their marginal probabilities. The matrix is factorised by singular value decomposition.

HLBL (Mnih and Hinton, 2009) is a log-bilinear formulation of an n-gram language model, which predicts the ith word based on context words $(i - n, \ldots, i-2, i-1)$. This leads to the following training objective:

$$J = \frac{1}{T}\sum_{i=1}^{T} \frac{\exp(\tilde{\mathbf{w}}_i^\top \mathbf{w}_i + b_i)}{\sum_{k=1}^{V} \exp(\tilde{\mathbf{w}}_i^\top \mathbf{w}_k + b_k)} \,,$$

where $\tilde{\mathbf{w}}_i = \sum_{j=1}^{n-1} C_j \mathbf{w}_{i-j}$ is the context embedding, C_j is a scaling matrix, and b_* is a bias term.

The final model, SENNA (Collobert and Weston, 2008), was initially proposed for multi-task training of several language processing tasks, from language modelling through to semantic role labelling. Here we focus on the statistical language modelling component, which has a pairwise ranking objective to maximise the relative score of each word in its local context:

$$J = \frac{1}{T}\sum_{i=1}^{T}\sum_{k=1}^{V} \max\big[0, 1 - f(\mathbf{w}_{i-c}, \ldots, \mathbf{w}_{i-1}, \mathbf{w}_i) + f(\mathbf{w}_{i-c}, \ldots, \mathbf{w}_{i-1}, \mathbf{w}_k)\big] \,,$$

where the last $c-1$ words are used as context, and $f(x)$ is a non-linear function of the input, defined as a multi-layer perceptron.

For HLBL and SENNA, we use the pre-trained embeddings from Turian et al. (2010), trained on the Reuters English newswire corpus. In both cases, the embeddings were scaled by the global standard deviation over the word-embedding matrix, $W_{\text{scaled}} = 0.1 \times \frac{W}{\sigma(W)}$.

For w2v$_{\text{wiki}}$, GloVe$_{\text{wiki}}$ and SVD$_{\text{wiki}}$ we used English Wikipedia. We followed the same preprocessing procedure described in Levy et al. (2015a),[3] i.e., lower-cased all words and removed non-textual elements. During the training phase, for each model we set a word frequency threshold of 5. For the SVD model, we followed the recommendations of Levy et al. (2015a) in setting the context window size to 2, negative sampling parameter to 1, eigenvalue weighting to 0.5, and context distribution smoothing to 0.75; other parameters were assigned

duty, denoting either the embedding for the ith token, $\mathbf{w}_i$, or kth word type, $\mathbf{w}_k$.

[3] Although the w2v model trained without preprocessing performed marginally better, we used preprocessing throughout for consistency.

their default values. For the other models we used the following parameter values: for w2v, context window = 8, negative samples = 25, hs = 0, sample = 1e-4, and iterations = 15; and for GloVe, context window = 15, x_max = 10, and iterations = 15.

3.2 Lexical Relations

In order to evaluate the applicability of the DIFFVEC approach to relations of different types, we assembled a set of lexical relations in three broad categories: lexical semantic relations, morphosyntactic paradigm relations, and morphosemantic relations. We constrained the relations to be binary and to have fixed directionality.[4] Consequently we excluded symmetric lexical relations such as synonymy. We additionally constrained the dataset to the words occurring in all embedding sets. There is some overlap between our relations and those included in the analogy task of Mikolov et al. (2013c), but we include a much wider range of lexical semantic relations, especially those standardly evaluated in the relation classification literature. We manually filtered the data to remove duplicates (e.g., as part of merging the two sources of LEXSEM$_{\text{Hyper}}$ intances), and normalise directionality.

The final dataset consists of 12,458 triples $\langle \text{relation}, \text{word}_1, \text{word}_2 \rangle$, comprising 15 relation types, extracted from SemEval'12 (Jurgens et al., 2012), BLESS (Baroni and Lenci, 2011), the MSR analogy dataset (Mikolov et al., 2013c), the light verb dataset of Tan et al. (2006a), Princeton WordNet (Fellbaum, 1998), Wiktionary,[5] and a web lexicon of collective nouns,[6] as listed in Table 2.[7]

4 Clustering

Assuming DIFFVECs are capable of capturing all lexical relations equally, we would expect clustering to be able to identify sets of word pairs with high relational similarity, or equivalently clusters of similar offset vectors. Under the additional assumption that a given word pair corresponds to a unique lexical relation (in line with our definition of the lexical relation learning task in §3), a hard clustering approach is appropriate. In order to test these assumptions, we cluster our 15-relation closed-world dataset in the first instance, and evaluate against the lexical resources in §3.2.

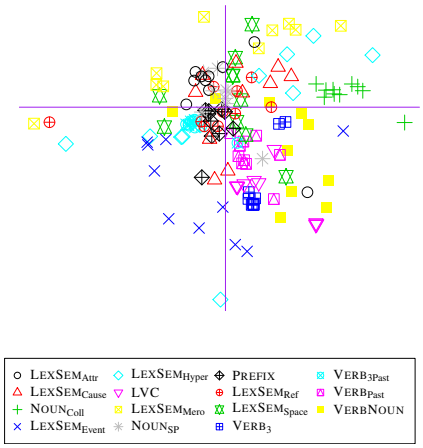


Figure 1: t-SNE projection (Van der Maaten and Hinton, 2008) of DIFFVECs for 10 sample word pairs of each relation type, based on w2v. The intersection of the two axes identify the projection of the zero vector. Best viewed in colour.

As further motivation, we projected the DIFFVEC space for a small number of samples of each class using t-SNE (Van der Maaten and Hinton, 2008), and found that many of the morphosyntactic relations (VERB$_3$, VERB$_{\text{Past}}$, VERB$_{3\text{Past}}$, NOUN$_{\text{SP}}$) form tight clusters (Figure 1).

We cluster the DIFFVECs between all word pairs in our dataset using spectral clustering (Von Luxburg, 2007). Spectral clustering has two hyperparameters: the number of clusters, and the pairwise similarity measure for comparing DIFFVECs. We tune the hyperparameters over development data, in the form of 15% of the data obtained by random sampling, selecting the configuration that maximises the V-Measure (Rosenberg and Hirschberg, 2007). Figure 2 presents V-Measure values over the test data for each of the four word embedding models. We show results for different numbers of clusters, from $N = 10$ in steps of 10, up to $N = 80$ (beyond which the clustering quality diminishes).[8] Observe that w2v achieves the best results, with a V-Measure value of around 0.36,[9] which is relatively constant over varying numbers of clusters. GloVe and SVD mirror this result, but are consistently below w2v at a V-Measure of around 0.31. HLBL and SENNA performed very

[4] Word similarity is not included; it is not easily captured by DIFFVEC since there is no homogeneous "content" to the lexical relation which could be captured by the direction and magnitude of a difference vector (other than that it should be small).

[5] http://en.wiktionary.org

[6] http://www.rinkworks.com/words/collective.shtml

[7] The dataset is available at http://github.com/ivri/DiffVec

[8] Although 80 clusters $\gg$ our 15 relation types, the SemEval'12 classes each contain numerous subclasses, so the larger number may be more realistic.

[9] V-Measure returns a value in the range $[0, 1]$, with 1 indicating perfect homogeneity and completeness.

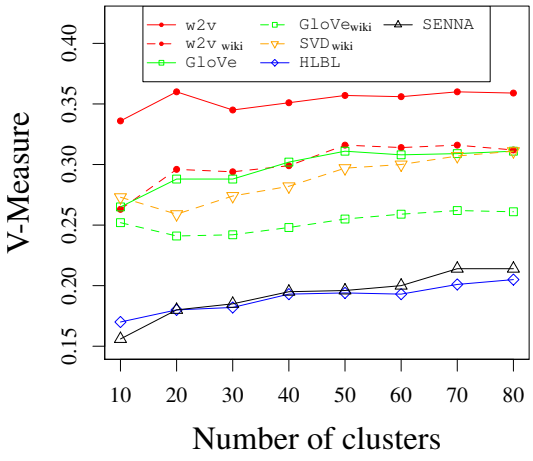


Figure 2: Spectral clustering results, comparing cluster quality (V-Measure) and the number of clusters. DIFFVECs are clustered and compared to the known relation types. Each line shows a different source of word embeddings.

	w2v	GloVe	HLBL	SENNA
LEXSEM$_{\text{Attr}}$	**0.49**	0.54	0.62	0.63
LEXSEM$_{\text{Cause}}$	**0.47**	0.53	0.56	0.57
LEXSEM$_{\text{Space}}$	**0.49**	0.55	0.54	0.58
LEXSEM$_{\text{Ref}}$	**0.44**	0.50	0.54	0.56
LEXSEM$_{\text{Hyper}}$	0.44	0.50	**0.43**	0.45
LEXSEM$_{\text{Event}}$	**0.46**	0.47	0.47	0.48
LEXSEM$_{\text{Mero}}$	**0.40**	0.42	0.42	0.43
NOUN$_{\text{SP}}$	**0.07**	0.14	0.22	0.29
VERB$_3$	**0.05**	0.06	0.49	0.44
VERB$_{\text{Past}}$	**0.09**	0.14	0.38	0.35
VERB$_{\text{3Past}}$	0.07	**0.05**	0.49	0.52
LVC	**0.28**	0.55	0.32	0.30
VERBNOUN	**0.31**	0.33	0.35	0.36
PREFIX	0.32	**0.30**	0.55	0.58
NOUN$_{\text{Coll}}$	**0.21**	0.27	0.46	0.44

Table 3: The entropy for each lexical relation over the clustering output for each set of pre-trained word embeddings.

similarly, at a substantially lower V-Measure than w2v or GloVe, closer to 0.21. As a crude calibration for these results, over the related clustering task of word sense induction, the best-performing systems in SemEval-2010 Task 4 (Manandhar et al., 2010) achieved a V-Measure of under 0.2.

The lower V-measure for w2v$_{\text{wiki}}$ and GloVe$_{\text{wiki}}$ (as compared to w2v and GloVe, respectively) indicates that the volume of training data plays a role in the clustering results. However, both methods still perform well above SENNA and HLBL, and w2v has a clear empirical advantage over GloVe. We note that SVD$_{\text{wiki}}$ performs almost as well as w2v$_{\text{wiki}}$, consistent with the results of Levy et al. (2015a).

We additionally calculated the entropy for each lexical relation, based on the distribution of instances belonging to a given relation across the different clusters (and simple MLE). For each embedding method, we present the entropy for the cluster size where V-measure was maximised over the development data. Since the samples are distributed nonuniformly, we normalise entropy results for each method by $\log(n)$ where n is the number of samples in a particular relation. The results are in Table 3, with the lowest entropy (purest clustering) for each relation indicated in bold.

Looking across the different lexical relation types, the morphosyntactic paradigm relations (NOUN$_{\text{SP}}$ and the three VERB relations) are by far the easiest to capture. The lexical semantic relations, on the other hand, are the hardest to capture for all embeddings.

Considering w2v embeddings, for VERB$_3$ there was a single cluster consisting of around 90% of VERB$_3$ word pairs. Most errors resulted from POS ambiguity, leading to confusion with VERB-NOUN in particular. Example VERB$_3$ pairs incorrectly clustered are: (*study*, *studies*), (*run*, *runs*), and (*like*, *likes*). This polysemy results in the distance represented in the DIFFVEC for such pairs being above average for VERB$_3$, and consequently clustered with other cross-POS relations.

For VERB$_{\text{Past}}$, a single relatively pure cluster was generated, with minor contamination due to pairs such as (*hurt*, *saw*), (*utensil*, *saw*), and (*wipe*, *saw*). Here, the noun *saw* is ambiguous with a high-frequency past-tense verb; *hurt* and *wipe* also have ambigous POS.

A related phenomenon was observed for NOUN$_{\text{Coll}}$, where the instances were assigned to a large mixed cluster containing word pairs where the second word referred to an animal, reflecting the fact that most of the collective nouns in our dataset relate to animals, e.g. (*stand*, *horse*), (*ambush*, *tigers*), (*antibiotics*, *bacteria*). This is interesting from a DIFFVEC point of view, since it shows that the lexical semantics of one word in the pair can overwhelm the semantic content of the DIFFVEC (something that we return to investigate in §5.4). LEXSEM$_{\text{Mero}}$ was also split into multiple clusters along topical lines, with separate clusters for weapons, dwellings, vehicles, etc.

Given the encouraging results from our clustering experiment, we next evaluate DIFFVECs in a supervised relation classification setting.

5 Classification

A natural question is whether we can accurately characterise lexical relations through supervised learning over the DIFFVECs. For these experiments we use the w2v, w2v_{wiki}, and SVD_{wiki} embeddings exclusively (based on their superior performance in the clustering experiment), and a subset of the relations which is both representative of the breadth of the full relation set, and for which we have sufficient data for supervised training and evaluation, namely: $\text{NOUN}_{\text{Coll}}$, $\text{LEXSEM}_{\text{Event}}$, $\text{LEXSEM}_{\text{Hyper}}$, $\text{LEXSEM}_{\text{Mero}}$, NOUN_{SP}, PREFIX, VERB_{3}, $\text{VERB}_{3\text{Past}}$, and $\text{VERB}_{\text{Past}}$ (see Table 2).

We consider two applications: (1) a CLOSED-WORLD setting similar to the unsupervised evaluation, in which the classifier only encounters word pairs which correspond to one of the nine relations; and (2) a more challenging OPEN-WORLD setting where random word pairs — which may or may not correspond to one of our relations — are included in the evaluation. For both settings, we further investigate whether there is a lexical memorisation effect for a broad range of relation types of the sort identified by Weeds et al. (2014) and Levy et al. (2015b) for hypernyms, by experimenting with disjoint training and test vocabulary.

5.1 CLOSED-WORLD Classification

For the CLOSED-WORLD setting, we train and test a multiclass classifier on datasets comprising $\langle \text{DIFFVEC}, r \rangle$ pairs, where r is one of our nine relation types, and DIFFVEC is based on one of w2v, w2v_{wiki} and SVD. As a baseline, we cluster the data as described in §4, running the clusterer several times over the 9-relation data to select the optimal V-Measure value based on the development data, resulting in 50 clusters. We label each cluster with the majority class based on the training instances, and evaluate the resultant labelling for the test instances.

We use an SVM with a linear kernel, and report results from 10-fold cross-validation in Table 4.

The SVM achieves a higher F-score than the baseline on almost every relation, particularly on $\text{LEXSEM}_{\text{Hyper}}$, and the lower-frequency NOUN_{SP}, $\text{NOUN}_{\text{Coll}}$, and PREFIX. Most of the relations — even the most difficult ones from our clustering experiment — are classified with very high F-score. That is, with a simple linear transformation of the embedding dimensions, we are able to achieve near-perfect results. The PREFIX relation achieved markedly lower recall, resulting in a lower F-score, due to large differences in the predominant usages associated with the respective words (e.g., (*union*, *reunion*), where the vector for *union* is heavily biased by contexts associated with trade unions, but *reunion* is heavily biased by contexts relating to social get-togethers; and (*entry*, *reentry*), where *entry* is associated with competitions and entrance to schools, while *reentry* is associated with space travel). Somewhat surprisingly, given the small dimensionality of the input (vectors of size 300 for all three methods), we found that the linear SVM slightly outperformed a non-linear SVM using an RBF kernel. We observe no real difference between w2v_{wiki} and SVD_{wiki}, supporting the hypothesis of Levy et al. (2015a) that under appropriate parameter settings, count-based methods achieve high results. The impact of the training data volume for pre-training of the embeddings is also less pronounced than in the case of our clustering experiment.

Relation	Baseline	w2v	w2v_{wiki}	SVD_{wiki}
$\text{LEXSEM}_{\text{Hyper}}$	0.60	0.93	0.91	0.91
$\text{LEXSEM}_{\text{Mero}}$	0.90	0.97	0.96	0.96
$\text{LEXSEM}_{\text{Event}}$	0.87	0.98	0.97	0.97
NOUN_{SP}	0.00	0.83	0.78	0.74
VERB_{3}	0.99	0.98	0.96	0.97
$\text{VERB}_{\text{Past}}$	0.78	0.98	0.98	0.95
$\text{VERB}_{3\text{Past}}$	0.99	0.98	0.98	0.96
PREFIX	0.00	0.82	0.34	0.60
$\text{NOUN}_{\text{Coll}}$	0.19	0.95	0.91	0.92
Micro-average	0.84	0.97	0.95	0.95

Table 4: F-scores ($\mathcal{F}$) for CLOSED-WORLD classification, for a baseline method based on clustering + majority-class labelling, a multiclass linear SVM trained on w2v, w2v_{wiki} and SVD_{wiki} DIFFVEC inputs.

5.2 OPEN-WORLD Classification

We now turn to a more challenging evaluation setting: a test set including word pairs drawn at random. This setting aims to illustrate whether a DIFFVEC-based classifier is capable of differentiating related word pairs from noise, and can be applied to open data to learn new related word pairs.[10]

For these experiments, we train a binary classifier for each relation type, using $\frac{2}{3}$ of our relation data for training and $\frac{1}{3}$ for testing. The test data is augmented with an equal quantity of random pairs, generated as follows:

(1) sample a seed lexicon by drawing words proportional to their frequency in Wikipedia;[11]

[10] Hereafter we provide results for w2v only, as we found that SVD achieved similar results.

[11] Filtered to consist of words for which we have embed-

Relation	Orig			+neg		
	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
LEXSEM$_{\text{Hyper}}$	0.95	**0.92**	**0.93**	**0.99**	0.84	0.91
LEXSEM$_{\text{Mero}}$	0.13	**0.96**	0.24	**0.95**	0.84	**0.89**
LEXSEM$_{\text{Event}}$	0.44	**0.98**	0.61	**0.93**	0.90	**0.91**
NOUN$_{\text{SP}}$	0.95	**0.68**	0.8	**1.00**	**0.68**	**0.81**
VERB$_3$	0.75	**1.00**	0.86	**0.93**	0.93	**0.93**
VERB$_{\text{Past}}$	0.94	**0.86**	**0.90**	**0.97**	0.84	**0.90**
VERB$_{\text{3Past}}$	0.76	**0.95**	0.84	**0.87**	0.93	**0.90**
PREFIX	**1.00**	**0.29**	**0.44**	**1.00**	0.13	0.23
NOUN$_{\text{Coll}}$	0.43	**0.74**	0.55	**0.97**	0.41	**0.57**

Table 5: Precision ($\mathcal{P}$) and recall ($\mathcal{R}$) for OPEN-WORLD classification, using the binary classifier without ("Orig") and with ("+neg") negative samples .

(2) take the Cartesian product over pairs of words from the seed lexicon;
(3) sample word pairs uniformly from this set.

This procedure generates word pairs that are representative of the frequency profile of our corpus.

We train 9 binary RBF-kernel SVM classifiers on the training partition, and evaluate on our randomly augmented test set. Fully annotating our random word pairs is prohibitively expensive, so instead, we manually annotated only the word pairs which were positively classified by one of our models. The results of our experiments are presented in the left half of Table 5, in which we report on results over the combination of the original test data from §5.1 and the random word pairs, noting that recall ($\mathcal{R}$) for OPEN-WORLD takes the form of relative recall (Pantel et al., 2004) over the positively-classified word pairs. The results are much lower than for the closed-word setting (Table 4), most notably in terms of precision ($\mathcal{P}$). For instance, the random pairs (*have*, *works*), (*turn*, *took*), and (*works*, *started*) were incorrectly classified as VERB$_3$, VERB$_{\text{Past}}$ and VERB$_{\text{3Past}}$, respectively. That is, the model captures syntax, but lacks the ability to capture lexical paradigms, and tends to overgenerate.

5.3 OPEN-WORLD Training with Negative Sampling

To address the problem of incorrectly classifying random word pairs as valid relations, we retrain the classifier on a dataset comprising both valid and automatically-generated negative distractor samples. The basic intuition behind this approach is to construct samples which will force the model to learn decision boundaries that more tightly capture the true scope of a given relation. To this end, we automatically generated two types of negative distractors:

opposite pairs: generated by switching the order of word pairs, $\mathit{Oppos}_{w1,w2} = \mathbf{word_1} - \mathbf{word_2}$. This ensures the classifier adequately captures the asymmetry in the relations.

shuffled pairs: generated by replacing w_2 with a random word w_2' from the same relation, $\mathit{Shuff}_{w1,w2} = \mathbf{word_2'} - \mathbf{word_1}$. This is targeted at relations that take specific word classes in particular positions, e.g., (VB, VBD) word pairs, so that the model learns to encode the relation rather than simply learning the properties of the word classes.

Both types of distractors are added to the training set, such that there are equal numbers of valid relations, opposite pairs and shuffled pairs.

After training our classifier, we evaluate its predictions in the same way as in §5.2, using the same test set combining related and random word pairs.[12] The results are shown in the right half of Table 5 (as "+neg"). Observe that the precision is much higher and recall somewhat lower compared to the classifier trained with only positive samples. This follows from the adversarial training scenario: using negative distractors results in a more conservative classifier, that correctly classifies the vast majority of the random word pairs as not corresponding to a given relation, resulting in higher precision at the expense of a small drop in recall. Overall this leads to higher F-scores, as shown in Figure 3, other than for hypernyms (LEXSEM$_{\text{Hyper}}$) and prefixes (PREFIX). For example, the standard classifier for NOUN$_{\text{Coll}}$ learned to match word pairs including an animal name (e.g., (*plague*, *rats*)), while training with negative samples resulted in much more conservative predictions and consequently much lower recall. The classifier was able to capture (*herd*, *horses*) but not (*run*, *salmon*), (*party*, *jays*) or (*singular*, *boar*) as instances of NOUN$_{\text{Coll}}$, possibly because of polysemy. The most striking difference in performance was for LEXSEM$_{\text{Mero}}$, where the standard classifier generated many false positive noun pairs (e.g. (*series*, *radio*)), but the false positive rate was considerably reduced with negative sampling.

5.4 Lexical Memorisation

Weeds et al. (2014) and Levy et al. (2015b) recently showed that supervised methods using DIFFVECs achieve artificially high results as a result of "lexical memorisation" over frequent words asso-

dings.

[12] But noting that relative recall for the random word pairs is based on the pool of positive predictions from both models.

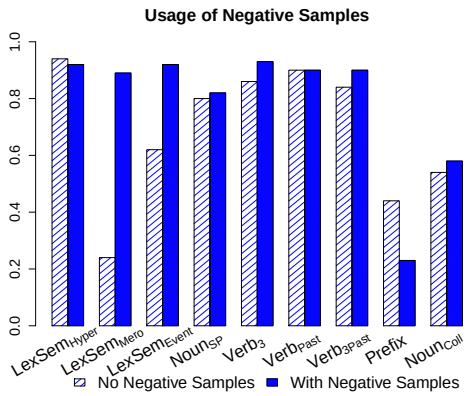


Figure 3: F-score for OPEN-WORLD classification, comparing models trained with and without negative samples.

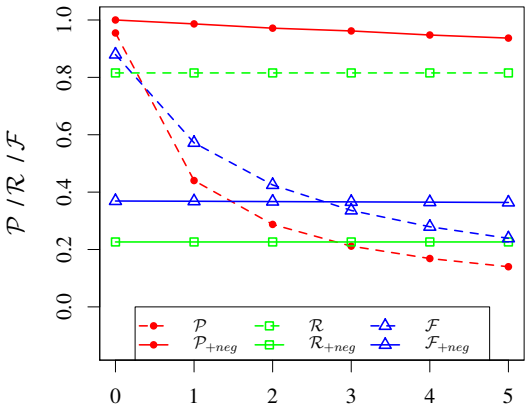


Figure 4: Evaluation of the OPEN-WORLD model when trained on split vocabulary, for varying numbers of random word pairs in the test dataset (expressed as a multiplier relative to the number of CLOSED-WORLD test instances).

ciated with the hypernym relation. For example, (*animal*, *cat*), (*animal*, *dog*), and (*animal*, *pig*) all share the superclass *animal*, and the model thus learns to classify as positive any word pair with *animal* as the first word.

To address this effect, we follow Levy et al. (2015b) in splitting our vocabulary into training and test partitions, to ensure there is no overlap between training and test vocabulary. We then train classifiers with and without negative sampling (§5.3), incrementally adding the random word pairs from §5.2 to the test data (from no random word pairs to five times the original size of the test data) to investigate the interaction of negative sampling with greater diversity in the test set when there is a split vocabulary. The results are shown in Figure 4.

Observe that the precision for the standard classifier decreases rapidly as more random word pairs are added to the test data. In comparison, the precision when negative sampling is used shows only a small drop-off, indicating that negative sampling is effective at maintaining precision in an OPEN-WORLD setting even when the training and test vocabulary are disjoint. This benefit comes at the expense of recall, which is much lower when negative sampling is used (note that recall stays relatively constant as random word pairs are added, as the vast majority of them do not correspond to any relation). At the maximum level of random word pairs in the test data, the F-score for the negative sampling classifier is higher than for the standard classifier.

6 Conclusions

This paper is the first to test the generalisability of the vector difference approach across a broad range of lexical relations (in raw number and also variety). Using clustering we showed that many types of morphosyntactic and morphosemantic differences are captured by DIFFVECs, but that lexical semantic relations are captured less well, a finding which is consistent with previous work (Köper et al., 2015). In contrast, classification over the DIFFVECs works extremely well in a closed-world setting, showing that dimensions of DIFFVECs encode lexical relations. Classification performs less well over open data, although with the introduction of automatically-generated negative samples, the results improve substantially. Negative sampling also improves classification when the training and test vocabulary are split to minimise lexical memorisation. Overall, we conclude that the DIFFVEC approach has impressive utility over a broad range of lexical relations, especially under supervised classification.

Acknowledgments

LR was supported by EPSRC grant EP/I037512/1 and ERC Starting Grant DisCoTex (306920). TC and TB were supported by the Australian Research Council.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Random walks on context spaces: Towards an explanation of the mysteries of semantic word embeddings. arXiv:1502.03520 [cs.LG].

Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pages 2670–2676, Hyderabad, India.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, GEMS '11, pages 1–10, Edinburgh, Scotland.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the EACL (EACL 2012)*, pages 23–32, Avignon, France.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems 25 (NIPS-13)*, pages 2787–2795.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 160–167, Helsinki, Finland.

Manaal Faruqui, Jesse Dodge, Sujay Jauhar, Chris Dyer, Ed Hovy, and Noah Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2015)*, pages 1351–1356, Denver, USA.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, USA.

Daniel Fried and Kevin Duh. 2015. Incorporating both distributional and relational semantics in word representations. In *Proceedings of the Third International Conference on Learning Representations (ICLR 2015)*, San Diego, USA.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 1199–1209, Baltimore, USA.

Maayan Geffet and Ido Dagan. 2005. The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pages 107–114, Ann Arbor, USA.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. 2007. SemEval-2007 Task 4: Classification of semantic relations between nominals. In *Proceedings of the 4th International Workshop on Semantic Evaluation (SemEval 2007)*, pages 13–18, Prague, Czech Republic.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. SemEval-2010 Task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SemEval 2010)*, pages 33–38, Uppsala, Sweden.

David Jurgens, Saif Mohammad, Peter Turney, and Keith Holyoak. 2012. SemEval-2012 Task 2: Measuring degrees of relational similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation (SemEval 2012)*, pages 356–364, Montréal, Canada.

Joo-Kyung Kim and Marie-Catherine de Marneffe. 2013. Deriving adjectival scales from continuous space word representations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)*, pages 1625–1630, Seattle, USA.

Maximilian Köper, Christian Scheible, and Sabine Schulte im Walde. 2015. Multilingual reliability and "semantic" structure of continuous word spaces. In *Proceedings of the Eleventh International Workshop on Computational Semantics (IWCS-11)*, pages 40–45, London, UK.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16:359–389.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*, pages 75–79, Montréal, Canada.

Omer Levy and Yoav Goldberg. 2014a. Linguistic regularities in sparse and explicit word representations. In *Proceedings of the 18th Conference on Natural Language Learning (CoNLL-2014)*, pages 171–180, Baltimore, USA.

Omer Levy and Yoav Goldberg. 2014b. Neural word embeddings as implicit matrix factorization. In *Advances in Neural Information Processing Systems 26 (NIPS-14)*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015a. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Omer Levy, Steffen Remus, Chris Biemann, Ido Dagan, and Israel Ramat-Gan. 2015b. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics — Human Language Technologies (NAACL HLT 2015)*, pages 970–976, Denver, USA.

Márton Makrai, Dávid Nemeskey, and András Kornai. 2013. Applicative structure in vector space models. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 59–63, Sofia, Bulgaria.

Suresh Manandhar, Ioannis Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. SemEval-2010 Task 14: Word sense induction & disambiguation. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 63–68, Uppsala, Sweden.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Proceedings of the Workshop of the First International Conference on Learning Representations (ICLR 2013)*, Scottsdale, USA.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 25 (NIPS-13)*.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*, pages 746–751, Atlanta, USA.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in Neural Information Processing Systems 21 (NIPS-09)*, pages 1081–1088.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. In *Advances in Neural Information Processing Systems 25 (NIPS-13)*.

Silvia Necşulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics (*SEM 2015)*, pages 182–192, Denver, USA.

Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. 2004. Towards terascale semantic acquisition. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 771–777, Geneva, Switzerland.

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

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 511–519, Gothenburg, Sweden.

Stephen Roller and Katrin Erk. 2016. Relations such as hypernymy: Identifying and exploiting Hearst patterns in distributional vectors for lexical entailment. *arXiv preprint arXiv:1605.05433*.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1025–1036, Dublin, Ireland.

Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning 2007 (EMNLP-CoNLL 2007)*, pages 410–420, Prague, Czech Republic.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, pages 38–42, Gothenburg, Sweden.

Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems 25 (NIPS-13)*.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2006a. *Introduction to Data Mining*. Addison Wesley.

Yee Fan Tan, Min-Yen Kan, and Hang Cui. 2006b. Extending corpus-based identification of light verb constructions using a supervised learning framework. In *Proceedings of the EACL 2006 Workshop on Multiword-expressions in a Multilingual Context*, pages 49–56, Trento, Italy.

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the ACL (ACL 2010)*, pages 384–394, Uppsala, Sweden.

Peter D. Turney. 2006. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416.

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

Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 2249–2259, Dublin, Ireland.

Gerhard Weikum and Martin Theobald. 2010. From information to knowledge: harvesting entities and relationships from web sources. In *Proceedings of the Twenty Ninth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 65–76, Indianapolis, USA.

Chang Xu, Yanlong Bai, Jiang Bian, Bin Gao, Gang Wang, Xiaoguang Liu, and Tie-Yan Liu. 2014. RC-NET: A general framework for incorporating knowledge into word representations. In *Proceedings of the 23rd ACM Conference on Information and Knowledge Management (CIKM 2014)*, pages 1219–1228, Shanghai, China.

Ichiro Yamada, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, Masaki Murata, Stijn De Saeger, Francis Bond, and Asuka Sumida. 2009. Hypernym discovery based on distributional similarity and hierarchical structures. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*, pages 929–937, Singapore.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 545–550, Baltimore, USA.

A. Zhila, W.T. Yih, C. Meek, G. Zweig, and T. Mikolov. 2013. Combining heterogeneous models for measuring relational similarity. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT 2013)*.

Minimum Risk Training for Neural Machine Translation

Shiqi Shen[†], Yong Cheng[#], Zhongjun He[+], Wei He[+], Hua Wu[+], Maosong Sun[†], Yang Liu[†*]
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[#]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
[+]Baidu Inc., Beijing, China
{vicapple22, chengyong3001}@gmail.com, {hezhongjun, hewei06, wu_hua}@baidu.com, {sms, liuyang2011}@tsinghua.edu.cn

Abstract

We propose minimum risk training for end-to-end neural machine translation. Unlike conventional maximum likelihood estimation, minimum risk training is capable of optimizing model parameters directly with respect to arbitrary evaluation metrics, which are not necessarily differentiable. Experiments show that our approach achieves significant improvements over maximum likelihood estimation on a state-of-the-art neural machine translation system across various languages pairs. Transparent to architectures, our approach can be applied to more neural networks and potentially benefit more NLP tasks.

1 Introduction

Recently, end-to-end neural machine translation (NMT) (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015) has attracted increasing attention from the community. Providing a new paradigm for machine translation, NMT aims at training a single, large neural network that directly transforms a source-language sentence to a target-language sentence without explicitly modeling latent structures (e.g., word alignment, phrase segmentation, phrase reordering, and SCFG derivation) that are vital in conventional statistical machine translation (SMT) (Brown et al., 1993; Koehn et al., 2003; Chiang, 2005).

Current NMT models are based on the *encoder-decoder* framework (Cho et al., 2014; Sutskever et al., 2014), with an encoder to read and encode a source-language sentence into a vector, from which a decoder generates a target-language sentence. While early efforts encode the input into a fixed-length vector, Bahdanau et al. (2015) advocate the attention mechanism to dynamically generate a context vector for a target word being generated.

Although NMT models have achieved results on par with or better than conventional SMT, they still suffer from a major drawback: the models are optimized to maximize the likelihood of training data instead of evaluation metrics that actually quantify translation quality. Ranzato et al. (2015) indicate two drawbacks of *maximum likelihood estimation* (MLE) for NMT. First, the models are only exposed to the training distribution instead of model predictions. Second, the loss function is defined at the word level instead of the sentence level.

In this work, we introduce *minimum risk training* (MRT) for neural machine translation. The new training objective is to minimize the expected loss (i.e., risk) on the training data. MRT has the following advantages over MLE:

1. *Direct optimization with respect to evaluation metrics*: MRT introduces evaluation metrics as loss functions and aims to minimize expected loss on the training data.

2. *Applicable to arbitrary loss functions*: our approach allows arbitrary sentence-level loss functions, which are not necessarily differentiable.

3. *Transparent to architectures*: MRT does not assume the specific architectures of NMT and can be applied to any end-to-end NMT systems.

While MRT has been widely used in conventional SMT (Och, 2003; Smith and Eisner, 2006; He and Deng, 2012) and deep learning based MT (Gao et al., 2014), to the best of our knowledge, this work is the first effort to introduce MRT

*Corresponding author: Yang Liu.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1683–1692,
Berlin, Germany, August 7-12, 2016.

into end-to-end NMT. Experiments on a variety of language pairs (Chinese-English, English-French, and English-German) show that MRT leads to significant improvements over MLE on a state-of-the-art NMT system (Bahdanau et al., 2015).

2 Background

Given a source sentence $\mathbf{x} = \mathbf{x}_1, \ldots, \mathbf{x}_m, \ldots, \mathbf{x}_M$ and a target sentence $\mathbf{y} = \mathbf{y}_1, \ldots, \mathbf{y}_n, \ldots, \mathbf{y}_N$, end-to-end NMT directly models the translation probability:

$$P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta}) = \prod_{n=1}^{N} P(\mathbf{y}_n|\mathbf{x}, \mathbf{y}_{<n};\boldsymbol{\theta}), \quad (1)$$

where $\boldsymbol{\theta}$ is a set of model parameters and $\mathbf{y}_{<n} = \mathbf{y}_1, \ldots, \mathbf{y}_{n-1}$ is a partial translation.

Predicting the n-th target word can be modeled by using a recurrent neural network:

$$P(\mathbf{y}_n|\mathbf{x}, \mathbf{y}_{<n};\boldsymbol{\theta}) \propto \exp\Big\{q(\mathbf{y}_{n-1}, \mathbf{z}_n, \mathbf{c}_n, \boldsymbol{\theta})\Big\}, \quad (2)$$

where $\mathbf{z}_n$ is the n-th hidden state on the target side, $\boldsymbol{c}_n$ is the context for generating the n-th target word, and $q(\cdot)$ is a non-linear function. Current NMT approaches differ in calculating $\mathbf{z}_n$ and $\mathbf{c}_n$ and defining $q(\cdot)$. Please refer to (Sutskever et al., 2014; Bahdanau et al., 2015) for more details.

Given a set of training examples $D = \{\langle \mathbf{x}^{(s)}, \mathbf{y}^{(s)} \rangle\}_{s=1}^{S}$, the standard training objective is to maximize the log-likelihood of the training data:

$$\hat{\boldsymbol{\theta}}_{\mathrm{MLE}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \Big\{\mathcal{L}(\boldsymbol{\theta})\Big\}, \quad (3)$$

where

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{s=1}^{S} \log P(\mathbf{y}^{(s)}|\mathbf{x}^{(s)};\boldsymbol{\theta}) \quad (4)$$

$$= \sum_{s=1}^{S} \sum_{n=1}^{N^{(s)}} \log P(\mathbf{y}_n^{(s)}|\mathbf{x}^{(s)}, \mathbf{y}_{<n}^{(s)};\boldsymbol{\theta}). \quad (5)$$

We use $N^{(s)}$ to denote the length of the s-th target sentence $\mathbf{y}^{(s)}$.

The partial derivative with respect to a model parameter $\boldsymbol{\theta}_i$ is calculated as

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} = \sum_{s=1}^{S} \sum_{n=1}^{N^{(s)}} \frac{\partial P(\mathbf{y}_n^{(s)}|\mathbf{x}^{(s)}, \mathbf{y}_{<n}^{(s)};\boldsymbol{\theta})/\partial \boldsymbol{\theta}_i}{P(\mathbf{y}_n^{(s)}|\mathbf{x}^{(s)}, \mathbf{y}_{<n}^{(s)};\boldsymbol{\theta})}. \quad (6)$$

Ranzato et al. (2015) point out that MLE for end-to-end NMT suffers from two drawbacks. First, while the models are trained only on the training data distribution, they are used to generate target words on previous model predictions, which can be erroneous, at test time. This is referred to as *exposure bias* (Ranzato et al., 2015). Second, MLE usually uses the cross-entropy loss focusing on word-level errors to maximize the probability of the next correct word, which might hardly correlate well with corpus-level and sentence-level evaluation metrics such as BLEU (Papineni et al., 2002) and TER (Snover et al., 2006).

As a result, it is important to introduce new training algorithms for end-to-end NMT to include model predictions during training and optimize model parameters directly with respect to evaluation metrics.

3 Minimum Risk Training for Neural Machine Translation

Minimum risk training (MRT), which aims to minimize the expected loss on the training data, has been widely used in conventional SMT (Och, 2003; Smith and Eisner, 2006; He and Deng, 2012) and deep learning based MT (Gao et al., 2014). The basic idea is to introduce evaluation metrics as loss functions and assume that the optimal set of model parameters should minimize the expected loss on the training data.

Let $\langle \mathbf{x}^{(s)}, \mathbf{y}^{(s)} \rangle$ be the s-th sentence pair in the training data and $\mathbf{y}$ be a model prediction. We use a *loss function* $\Delta(\mathbf{y}, \mathbf{y}^{(s)})$ to measure the discrepancy between the model prediction $\mathbf{y}$ and the gold-standard translation $\mathbf{y}^{(s)}$. Such a loss function can be negative smoothed sentence-level evaluation metrics such as BLEU (Papineni et al., 2002), NIST (Doddington, 2002), TER (Snover et al., 2006), or METEOR (Lavie and Denkowski, 2009) that have been widely used in machine translation evaluation. Note that a loss function is not parameterized and thus not differentiable.

In MRT, the *risk* is defined as the expected loss with respect to the posterior distribution:

$$\mathcal{R}(\boldsymbol{\theta}) = \sum_{s=1}^{S} \mathbb{E}_{\mathbf{y}|\mathbf{x}^{(s)};\boldsymbol{\theta}}\Big[\Delta(\mathbf{y}, \mathbf{y}^{(s)})\Big] \quad (7)$$

$$= \sum_{s=1}^{S} \sum_{\mathbf{y} \in \mathcal{Y}(\mathbf{x}^{(s)})} P(\mathbf{y}|\mathbf{x}^{(s)};\boldsymbol{\theta}) \Delta(\mathbf{y}, \mathbf{y}^{(s)}), \quad (8)$$

where $\mathcal{Y}(\mathbf{x}^{(s)})$ is a set of all possible candidate translations for $\mathbf{x}^{(s)}$.

	$\Delta(\mathbf{y}, \mathbf{y}^{(s)})$	$P(\mathbf{y}\|\mathbf{x}^{(s)};\boldsymbol{\theta})$			
$\mathbf{y}_1$	-1.0	0.2	0.3	0.5	0.7
$\mathbf{y}_2$	-0.3	0.5	0.2	0.2	0.1
$\mathbf{y}_3$	-0.5	0.3	0.5	0.3	0.2
$\mathbb{E}_{\mathbf{y}\|\mathbf{x}^{(s)};\boldsymbol{\theta}}[\Delta(\mathbf{y}, \mathbf{y}^{(s)})]$	-0.50	-0.61	-0.71	-0.83	

Table 1: Example of minimum risk training. $\mathbf{x}^{(s)}$ is an observed source sentence, $\mathbf{y}^{(s)}$ is its corresponding gold-standard translation, and $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$ are model predictions. For simplicity, we suppose that the full search space contains only three candidates. The loss function $\Delta(\mathbf{y}, \mathbf{y}^{(s)})$ measures the difference between model prediction and gold-standard. The goal of MRT is to find a distribution (the last column) that correlates well with the gold-standard by minimizing the expected loss.

The training objective of MRT is to minimize the risk on the training data:

$$\hat{\boldsymbol{\theta}}_{\text{MRT}} = \underset{\boldsymbol{\theta}}{\text{argmin}} \Big\{ \mathcal{R}(\boldsymbol{\theta}) \Big\}. \tag{9}$$

Intuitively, while MLE aims to maximize the likelihood of training data, our training objective is to discriminate between candidates. For example, in Table 1, suppose the candidate set $\mathcal{Y}(\mathbf{x}^{(s)})$ contains only three candidates: $\mathbf{y}_1$, $\mathbf{y}_2$, and $\mathbf{y}_3$. According to the losses calculated by comparing with the gold-standard translation $\mathbf{y}^{(s)}$, it is clear that $\mathbf{y}_1$ is the best candidate, $\mathbf{y}_3$ is the second best, and $\mathbf{y}_2$ is the worst: $\mathbf{y}_1 > \mathbf{y}_3 > \mathbf{y}_2$. The right half of Table 1 shows four models. As model 1 (column 3) ranks the candidates in a reverse order as compared with the gold-standard (i.e., $\mathbf{y}_2 > \mathbf{y}_3 > \mathbf{y}_1$), it obtains the highest risk of -0.50. Achieving a better correlation with the gold-standard than model 1 by predicting $\mathbf{y}_3 > \mathbf{y}_1 > \mathbf{y}_2$, model 2 (column 4) reduces the risk to -0.61. As model 3 (column 5) ranks the candidates in the same order with the gold-standard, the risk goes down to -0.71. The risk can be further reduced by concentrating the probability mass on $\mathbf{y}_1$ (column 6). As a result, by minimizing the risk on the training data, we expect to obtain a model that correlates well with the gold-standard.

In MRT, the partial derivative with respect to a model parameter $\boldsymbol{\theta}_i$ is given by

$$\frac{\partial \mathcal{R}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} = \sum_{s=1}^{S} \mathbb{E}_{\mathbf{y}|\mathbf{x}^{(s)};\boldsymbol{\theta}} \Bigg[\Delta(\mathbf{y}, \mathbf{y}^{(s)}) \times \sum_{n=1}^{N^{(s)}} \frac{\partial P(\mathbf{y}_n|\mathbf{x}^{(s)}, \mathbf{y}_{<n}; \boldsymbol{\theta}) / \partial \boldsymbol{\theta}_i}{P(\mathbf{y}_n|\mathbf{x}^{(s)}, \mathbf{y}_{<n}; \boldsymbol{\theta})} \Bigg]. \tag{10}$$

Since Eq. (10) suggests there is no need to differentiate $\Delta(\mathbf{y}, \mathbf{y}^{(s)})$, MRT allows arbitrary non-differentiable loss functions. In addition, our approach is transparent to architectures and can be applied to arbitrary end-to-end NMT models.

Despite these advantages, MRT faces a major challenge: the expectations in Eq. (10) are usually intractable to calculate due to the exponential search space of $\mathcal{Y}(\mathbf{x}^{(s)})$, the non-decomposability of the loss function $\Delta(\mathbf{y}, \mathbf{y}^{(s)})$, and the context sensitiveness of NMT.

To alleviate this problem, we propose to only use a subset of the full search space to approximate the posterior distribution and introduce a new training objective:

$$\tilde{\mathcal{R}}(\boldsymbol{\theta}) = \sum_{s=1}^{S} \mathbb{E}_{\mathbf{y}|\mathbf{x}^{(s)};\boldsymbol{\theta},\alpha} \Big[\Delta(\mathbf{y}, \mathbf{y}^{(s)}) \Big] \tag{11}$$

$$= \sum_{s=1}^{S} \sum_{\mathbf{y} \in \mathcal{S}(\mathbf{x}^{(s)})} Q(\mathbf{y}|\mathbf{x}^{(s)}; \boldsymbol{\theta}, \alpha) \Delta(\mathbf{y}, \mathbf{y}^{(s)}), \tag{12}$$

where $\mathcal{S}(\mathbf{x}^{(s)}) \subset \mathcal{Y}(\mathbf{x}^{(s)})$ is a sampled subset of the full search space, and $Q(\mathbf{y}|\mathbf{x}^{(s)}; \boldsymbol{\theta}, \alpha)$ is a distribution defined on the subspace $\mathcal{S}(\mathbf{x}^{(s)})$:

$$Q(\mathbf{y}|\mathbf{x}^{(s)}; \boldsymbol{\theta}, \alpha) = \frac{P(\mathbf{y}|\mathbf{x}^{(s)}; \boldsymbol{\theta})^{\alpha}}{\sum_{\mathbf{y}' \in \mathcal{S}(\mathbf{x}^{(s)})} P(\mathbf{y}'|\mathbf{x}^{(s)}; \boldsymbol{\theta})^{\alpha}}. \tag{13}$$

Note that α is a hyper-parameter that controls the sharpness of the Q distribution (Och, 2003).

Algorithm 1 shows how to build $\mathcal{S}(\mathbf{x}^{(s)})$ by sampling the full search space. The sampled subset initializes with the gold-standard translation (line 1). Then, the algorithm keeps sampling a target word given the source sentence and the partial translation until reaching the end of sentence (lines 3-16). Note that sampling might produce duplicate candidates, which are removed when building

Input: the s-th source sentence in the training data $\mathbf{x}^{(s)}$, the s-th target sentence in the training data $\mathbf{y}^{(s)}$, the set of model parameters $\boldsymbol{\theta}$, the limit on the length of a candidate translation l, and the limit on the size of sampled space k.

Output: sampled space $\mathcal{S}(\mathbf{x}^{(s)})$.

$\mathcal{S}(\mathbf{x}^{(s)}) \leftarrow \{\mathbf{y}^{(s)}\}$; // the gold-standard translation is included
$i \leftarrow 1$;
while $i \leq k$ **do**
  $\mathbf{y} \leftarrow \emptyset$; // an empty candidate translation
  $n \leftarrow 1$;
  while $n \leq l$ **do**
    $y \sim P(\mathbf{y}_n|\mathbf{x}^{(s)}, \mathbf{y}_{<n}; \boldsymbol{\theta})$; // sample the n-th target word
    $\mathbf{y} \leftarrow \mathbf{y} \cup \{y\}$;
    if $y = \text{EOS}$ **then**
      break; // terminate if reach the end of sentence
    end
    $n \leftarrow n + 1$;
  end
  $\mathcal{S}(\mathbf{x}^{(s)}) \leftarrow \mathcal{S}(\mathbf{x}^{(s)}) \cup \{\mathbf{y}\}$;
  $i \leftarrow i + 1$;
end

Algorithm 1: Sampling the full search space.

the subspace. We find that it is inefficient to force the algorithm to generate exactly k distinct candidates because high-probability candidates can be sampled repeatedly, especially when the probability mass highly concentrates on a few candidates. In practice, we take advantage of GPU's parallel architectures to speed up the sampling. [1]

Given the sampled space, the partial derivative with respect to a model parameter $\boldsymbol{\theta}_i$ of $\tilde{\mathcal{R}}(\boldsymbol{\theta})$ is given by

$$\frac{\partial \tilde{R}(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_i} = \alpha \sum_{s=1}^{S} \mathbb{E}_{\mathbf{y}|\mathbf{x}^{(s)};\boldsymbol{\theta},\alpha} \left[\frac{\partial P(\mathbf{y}|\mathbf{x}^{(s)};\boldsymbol{\theta})/\partial \boldsymbol{\theta}_i}{P(\mathbf{y}|\mathbf{x}^{(s)};\boldsymbol{\theta})} \times \Big(\Delta(\mathbf{y}, \mathbf{y}^{(s)}) - \mathbb{E}_{\mathbf{y}'|\mathbf{x}^{(s)};\boldsymbol{\theta},\alpha}[\Delta(\mathbf{y}', \mathbf{y}^{(s)})] \Big) \right]. \quad (14)$$

Since $|\mathcal{S}(\mathbf{x}^{(s)})| \ll |\mathcal{Y}(\mathbf{x}^{(s)})|$, the expectations in Eq. (14) can be efficiently calculated by explicitly enumerating all candidates in $\mathcal{S}(\mathbf{x}^{(s)})$. In our experiments, we find that approximating the full space with 100 samples (i.e., $k = 100$) works very well and further increasing sample size does not lead to significant improvements (see Section 4.3).

[1] To build the subset, an alternative to sampling is computing top-k translations. We prefer sampling to computing top-k translations for efficiency: sampling is more efficient and easy-to-implement than calculating k-best lists, especially given the extremely parallel architectures of GPUs.

4 Experiments

4.1 Setup

We evaluated our approach on three translation tasks: Chinese-English, English-French, and English-German. The evaluation metric is BLEU (Papineni et al., 2002) as calculated by the `multi-bleu.perl` script.

For Chinese-English, the training data consists of 2.56M pairs of sentences with 67.5M Chinese words and 74.8M English words, respectively. We used the NIST 2006 dataset as the validation set (hyper-parameter optimization and model selection) and the NIST 2002, 2003, 2004, 2005, and 2008 datasets as test sets.

For English-French, to compare with the results reported by previous work on end-to-end NMT (Sutskever et al., 2014; Bahdanau et al., 2015; Jean et al., 2015; Luong et al., 2015b), we used the same subset of the WMT 2014 training corpus that contains 12M sentence pairs with 304M English words and 348M French words. The concatenation of news-test 2012 and news-test 2013 serves as the validation set and news-test 2014 as the test set.

For English-German, to compare with the results reported by previous work (Jean et al., 2015; Luong et al., 2015a), we used the same subset of the WMT 2014 training corpus that contains 4M sentence pairs with 91M English words and 87M German words. The concatenation of news-test 2012 and news-test 2013 is used as the validation set and news-test 2014 as the test set.

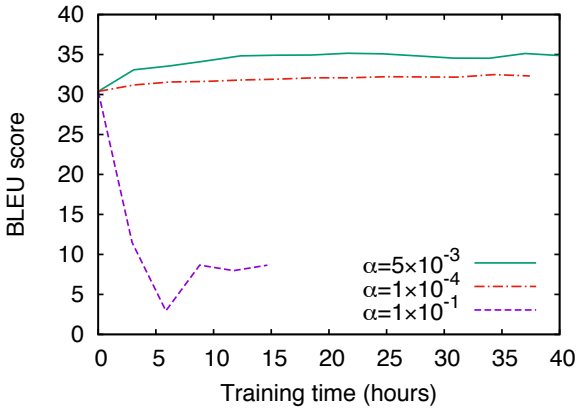


Figure 1: Effect of α on the Chinese-English validation set.

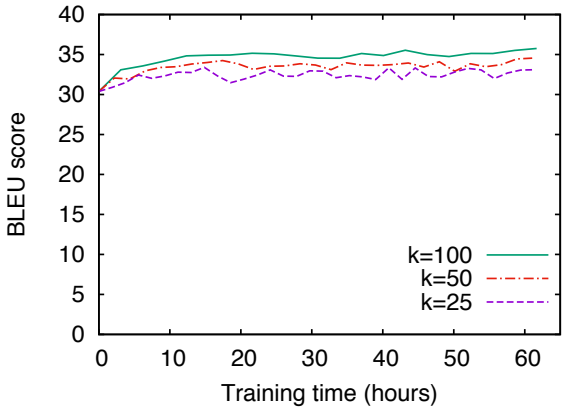


Figure 2: Effect of sample size on the Chinese-English validation set.

criterion	loss	BLEU	TER	NIST
MLE	N/A	30.48	60.85	8.26
MRT	−sBLEU	**36.71**	**53.48**	**8.90**
	sTER	30.14	53.83	6.02
	−sNIST	32.32	56.85	**8.90**

Table 2: Effect of loss function on the Chinese-English validation set.

We compare our approach with two state-of-the-art SMT and NMT systems:

1. MOSES (Koehn and Hoang, 2007): a phrase-based SMT system using minimum error rate training (Och, 2003).

2. RNNSEARCH (Bahdanau et al., 2015): an attention-based NMT system using maximum likelihood estimation.

MOSES uses the parallel corpus to train a phrase-based translation model and the target part to train a 4-gram language model using the SRILM toolkit (Stolcke, 2002). [2] The log-linear model Moses uses is trained by the minimum error rate training (MERT) algorithm (Och, 2003) that directly optimizes model parameters with respect to evaluation metrics.

RNNSEARCH uses the parallel corpus to train an attention-based neural translation model using the maximum likelihood criterion.

On top of RNNSEARCH, our approach replaces MLE with MRT. We initialize our model with the RNNsearch50 model (Bahdanau et al., 2015). We set the vocabulary size to 30K for Chinese-English and English-French and 50K for English-German. The beam size for decoding is 10. The default loss function is negative smoothed sentence-level BLEU.

4.2 Effect of α

The hyper-parameter α controls the smoothness of the Q distribution (see Eq. (13)). As shown in Figure 1, we find that α has a critical effect on BLEU scores on the Chinese-English validation set. While $\alpha = 1 \times 10^{-1}$ deceases BLEU scores dramatically, $\alpha = 5 \times 10^{-3}$ improves translation quality significantly and consistently. Reducing α further to 1×10^{-4}, however, results in lower BLEU scores. Therefore, we set $\alpha = 5 \times 10^{-3}$ in the following experiments.

4.3 Effect of Sample Size

For efficiency, we sample k candidate translations from the full search space $\mathcal{Y}(\mathbf{x}^{(s)})$ to build an approximate posterior distribution Q (Section 3). Figure 2 shows the effect of sample size k on the Chinese-English validation set. It is clear that BLEU scores consistently rise with the increase of k. However, we find that a sample size larger than 100 (e.g., $k = 200$) usually does not lead to significant improvements and increases the GPU memory requirement. Therefore, we set $k = 100$ in the following experiments.

4.4 Effect of Loss Function

As our approach is capable of incorporating evaluation metrics as loss functions, we investigate the effect of different loss functions on BLEU, TER

[2] It is possible to exploit larger monolingual corpora for both MOSES and RNNSEARCH (Gulcehre et al., 2015; Sennrich et al., 2015). We leave this for future work.

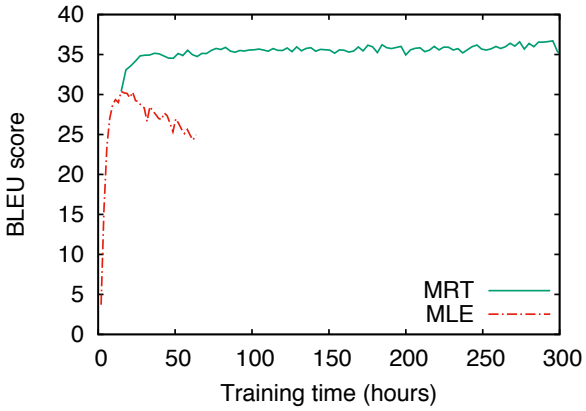


Figure 3: Comparison of training time on the Chinese-English validation set.

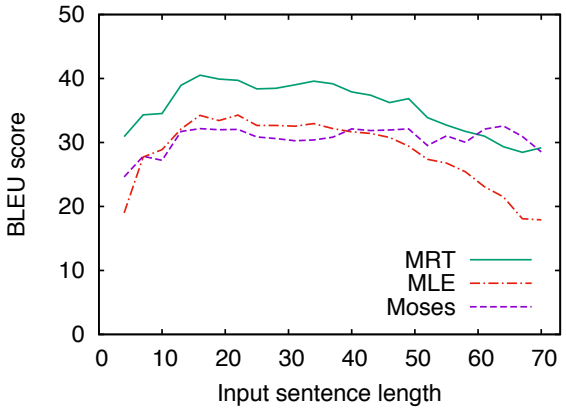


Figure 4: BLEU scores on the Chinese-English test set over various input sentence lengths.

and NIST scores on the Chinese-English validation set. As shown in Table 2, negative smoothed sentence-level BLEU (i.e, $-$sBLEU) leads to statistically significant improvements over MLE ($p < 0.01$). Note that the loss functions are all defined at the sentence level while evaluation metrics are calculated at the corpus level. This discrepancy might explain why optimizing with respect to sTER does not result in the lowest TER on the validation set. As $-$sBLEU consistently improves all evaluation metrics, we use it as the default loss function in our experiments.

4.5 Comparison of Training Time

We used a cluster with 20 Telsa K40 GPUs to train the NMT model. For MLE, it takes the cluster about one hour to train 20,000 mini-batches, each of which contains 80 sentences. The training time for MRT is longer than MLE: 13,000 mini-batches can be processed in one hour on the same cluster.

Figure 3 shows the learning curves of MLE and MRT on the validation set. For MLE, the BLEU score reaches its peak after about 20 hours and then keeps going down dramatically. Initializing with the best MLE model, MRT increases BLEU scores dramatically within about 30 hours. [3] Afterwards, the BLEU score keeps improving gradually but there are slight oscillations.

4.6 Results on Chinese-English Translation

4.6.1 Comparison of BLEU Scores

Table 3 shows BLEU scores on Chinese-English datasets. For RNNSEARCH, we follow Luong et al. (2015b) to handle rare words. We find that introducing minimum risk training into neural MT leads to surprisingly substantial improvements over MOSES and RNNSEARCH with MLE as the training criterion (up to +8.61 and +7.20 BLEU points, respectively) across all test sets. All the improvements are statistically significant.

[3] Although it is possible to initialize with a randomized model, it takes much longer time to converge.

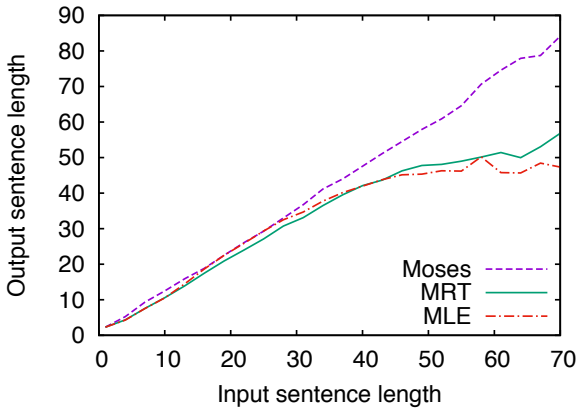


Figure 5: Comparison of output sentences lengths on the Chinese-English test set.

4.6.2 Comparison of TER Scores

Table 4 gives TER scores on Chinese-English datasets. The loss function used in MRT is $-$sBLEU. MRT still obtains dramatic improvements over MOSES and RNNSEARCH with MLE as the training criterion (up to -10.27 and -8.32 TER points, respectively) across all test sets. All the improvements are statistically significant.

4.6.3 BLEU Scores over Sentence Lengths

Figure 4 shows the BLEU scores of translations generated by MOSES, RNNSEARCH with MLE,

System	Training	MT06	MT02	MT03	MT04	MT05	MT08
MOSES	MERT	32.74	32.49	32.40	33.38	30.20	25.28
RNNSEARCH	MLE	30.70	35.13	33.73	34.58	31.76	23.57
	MRT	37.34	40.36	40.93	41.37	38.81	29.23

Table 3: Case-insensitive BLEU scores on Chinese-English translation.

System	Training	MT06	MT02	MT03	MT04	MT05	MT08
MOSES	MERT	59.22	62.97	62.44	61.20	63.44	62.36
RNNSEARCH	MLE	60.74	58.94	60.10	58.91	61.74	64.52
	MRT	52.86	52.87	52.17	51.49	53.42	57.21

Table 4: Case-insensitive TER scores on Chinese-English translation.

	MLE vs. MRT		
	$<$	$=$	$>$
evaluator 1	54%	24%	22%
evaluator 2	53%	22%	25%

Table 5: Subjective evaluation of MLE and MRT on Chinese-English translation.

and RNNSEARCH with MRT on the Chinese-English test set with respect to input sentence lengths. While MRT consistently improves over MLE for all lengths, it achieves worse translation performance for sentences longer than 60 words.

One reason is that RNNSEARCH tends to produce short translations for long sentences. As shown in Figure 5, both MLE and MRE generate much shorter translations than MOSES. This results from the length limit imposed by RNNSEARCH for efficiency reasons: a sentence in the training set is no longer than 50 words. This limit deteriorates translation performance because the sentences in the test set are usually longer than 50 words.

4.6.4 Subjective Evaluation

We also conducted a subjective evaluation to validate the benefit of replacing MLE with MRT. Two human evaluators were asked to compare MLE and MRT translations of 100 source sentences randomly sampled from the test sets without knowing from which system a candidate translation was generated.

Table 5 shows the results of subjective evaluation. The two human evaluators made close judgements: around 54% of MLE translations are worse than MRE, 23% are equal, and 23% are better.

4.6.5 Example Translations

Table 6 shows some example translations. We find that MOSES translates a Chinese string "*yi wei fuze yu pingrang dangju da jiaodao de qian guowuyuan guanyuan*" that requires long-distance reordering in a wrong way, which is a notorious challenge for statistical machine translation. In contrast, RNNSEARCH-MLE seems to overcome this problem in this example thanks to the capability of gated RNNs to capture long-distance dependencies. However, as MLE uses a loss function defined only at the word level, its translation lacks sentence-level consistency: "chinese" occurs twice while "two senate" is missing. By optimizing model parameters directly with respect to sentence-level BLEU, RNNSEARCH-MRT seems to be able to generate translations more consistently at the sentence level.

4.7 Results on English-French Translation

Table 7 shows the results on English-French translation. We list existing end-to-end NMT systems that are comparable to our system. All these systems use the same subset of the WMT 2014 training corpus and adopt MLE as the training criterion. They differ in network architectures and vocabulary sizes. Our RNNSEARCH-MLE system achieves a BLEU score comparable to that of Jean et al. (2015). RNNSEARCH-MRT achieves the highest BLEU score in this setting even with a vocabulary size smaller than Luong et al. (2015b) and Sutskever et al. (2014). Note that our approach does not assume specific architectures and can in principle be applied to any NMT systems.

4.8 Results on English-German Translation

Table 8 shows the results on English-German translation. Our approach still significantly out-

Source	*meiguo daibiao tuan baokuo laizi shidanfu daxue de yi wei zhongguo zhuanjia , liang ming canyuan waijiao zhengce zhuli yiji yi wei fuze yu pingrang dangju da jiaodao de qian guowuyuan guanyuan .*
Reference	the us delegation consists of a chinese expert from the stanford university , two senate foreign affairs policy assistants and a former state department official who was in charge of dealing with pyongyang authority .
MOSES	the united states to members of the delegation include representatives from the stanford university , a chinese expert , two assistant senate foreign policy and a responsible for dealing with pyongyang before the officials of the state council .
RNNSEARCH-MLE	the us delegation comprises a chinese expert from stanford university , a chinese foreign office assistant policy assistant and a former official who is responsible for dealing with the pyongyang authorities .
RNNSEARCH-MRT	the us delegation included a chinese expert from the stanford university , two senate foreign policy assistants , and a former state department official who had dealings with the pyongyang authorities .

Table 6: Example Chinese-English translations. "Source" is a romanized Chinese sentence, "Reference" is a gold-standard translation. "MOSES" and "RNNSEARCH-MLE" are baseline SMT and NMT systems. "RNNSEARCH-MRT" is our system.

System	Architecture	Training	Vocab	BLEU
Existing end-to-end NMT systems				
Bahdanau et al. (2015)	gated RNN with search	MLE	30K	28.45
Jean et al. (2015)	gated RNN with search		30K	29.97
Jean et al. (2015)	gated RNN with search + PosUnk		30K	33.08
Luong et al. (2015b)	LSTM with 4 layers		40K	29.50
Luong et al. (2015b)	LSTM with 4 layers + PosUnk		40K	31.80
Luong et al. (2015b)	LSTM with 6 layers		40K	30.40
Luong et al. (2015b)	LSTM with 6 layers + PosUnk		40K	32.70
Sutskever et al. (2014)	LSTM with 4 layers		80K	30.59
Our end-to-end NMT systems				
this work	gated RNN with search	MLE	30K	29.88
	gated RNN with search	MRT	30K	31.30
	gated RNN with search + PosUnk	MRT	30K	34.23

Table 7: Comparison with previous work on English-French translation. The BLEU scores are case-sensitive. "PosUnk" denotes Luong et al. (2015b)'s technique of handling rare words.

System	Architecture	Training	BLEU
Existing end-to-end NMT systems			
Jean et al. (2015)	gated RNN with search	MLE	16.46
Jean et al. (2015)	gated RNN with search + PosUnk		18.97
Jean et al. (2015)	gated RNN with search + LV + PosUnk		19.40
Luong et al. (2015a)	LSTM with 4 layers + dropout + local att. + PosUnk		20.90
Our end-to-end NMT systems			
this work	gated RNN with search	MLE	16.45
	gated RNN with search	MRT	18.02
	gated RNN with search + PosUnk	MRT	20.45

Table 8: Comparison with previous work on English-German translation. The BLEU scores are case-sensitive.

performs MLE and achieves comparable results with state-of-the-art systems even though Luong et al. (2015a) used a much deeper neural network. We believe that our work can be applied to their architecture easily.

Despite these significant improvements, the margins on English-German and English-French datasets are much smaller than Chinese-English. We conjecture that there are two possible reasons. First, the Chinese-English datasets contain four reference translations for each sentence while both English-French and English-German datasets only have single references. Second, Chinese and English are more distantly related than English, French and German and thus benefit more from MRT that incorporates evaluation metrics into optimization to capture structural divergence.

5 Related Work

Our work originated from the minimum risk training algorithms in conventional statistical machine translation (Och, 2003; Smith and Eisner, 2006; He and Deng, 2012). Och (2003) describes a smoothed error count to allow calculating gradients, which directly inspires us to use a parameter α to adjust the smoothness of the objective function. As neural networks are non-linear, our approach has to minimize the expected loss on the sentence level rather than the loss of 1-best translations on the corpus level. Smith and Eisner (2006) introduce minimum risk annealing for training log-linear models that is capable of gradually annealing to focus on the 1-best hypothesis. He et al. (2012) apply minimum risk training to learning phrase translation probabilities. Gao et al. (2014) leverage MRT for learning continuous phrase representations for statistical machine translation. The difference is that they use MRT to optimize a sub-model of SMT while we are interested in directly optimizing end-to-end neural translation models.

The Mixed Incremental Cross-Entropy Reinforce (MIXER) algorithm (Ranzato et al., 2015) is in spirit closest to our work. Building on the REINFORCE algorithm proposed by Williams (1992), MIXER allows incremental learning and the use of hybrid loss function that combines both REINFORCE and cross-entropy. The major difference is that Ranzato et al. (2015) leverage reinforcement learning while our work resorts to minimum risk training. In addition, MIXER only samples one candidate to calculate reinforcement reward while MRT generates multiple samples to calculate the expected risk. Figure 2 indicates that multiple samples potentially increases MRT's capability of discriminating between diverse candidates and thus benefit translation quality. Our experiments confirm their finding that taking evaluation metrics into account when optimizing model parameters does help to improve sentence-level text generation.

More recently, our approach has been successfully applied to summarization (Ayana et al., 2016). They optimize neural networks for headline generation with respect to ROUGE (Lin, 2004) and also achieve significant improvements, confirming the effectiveness and applicability of our approach.

6 Conclusion

In this paper, we have presented a framework for minimum risk training in end-to-end neural machine translation. The basic idea is to minimize the expected loss in terms of evaluation metrics on the training data. We sample the full search space to approximate the posterior distribution to improve efficiency. Experiments show that MRT leads to significant improvements over maximum likelihood estimation for neural machine translation, especially for distantly-related languages such as Chinese and English.

In the future, we plan to test our approach on more language pairs and more end-to-end neural MT systems. It is also interesting to extend minimum risk training to minimum risk annealing following Smith and Eisner (2006). As our approach is transparent to loss functions and architectures, we believe that it will also benefit more end-to-end neural architectures for other NLP tasks.

Acknowledgments

This work was done while Shiqi Shen and Yong Cheng were visiting Baidu. Maosong Sun and Hua Wu are supported by the 973 Program (2014CB340501 & 2014CB34505). Yang Liu is supported by the National Natural Science Foundation of China (No.61522204 and No.61432013) and the 863 Program (2015AA011808). This research is also supported by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme.

References

Ayana, Shiqi Shen, Zhiyuan Liu, and Maosong Sun. 2016. Neural headline generation with minimum risk training. arXiv:1604.01904.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguisitics*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of HLT*.

Jianfeng Gao, Xiaodong He, Wen tao Yih, and Li Deng. 2014. Learning continuous phrase representations for translation modeling. In *Proceedings of ACL*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. arXiv:1503.03535.

Xiaodong He and Li Deng. 2012. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of ACL*.

Sebastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of EMNLP*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.

Alon Lavie and Michael Denkowski. 2009. The mereor metric for automatic evaluation of machine translation. *Machine Translation*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Proceedings of ACL*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015a. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*.

Franz J. Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of ACL*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. arXiv:1511.06732v1.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. arXiv:1511.06709.

David A. Smith and Jason Eisner. 2006. Minimum risk annealing for training log-linear models. In *Proceedings of ACL*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*.

Andreas Stolcke. 2002. Srilm - am extensible language modeling toolkit. In *Proceedings of ICSLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Ronald J. Willams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*.

A Character-Level Decoder without Explicit Segmentation for Neural Machine Translation

Junyoung Chung
Université de Montréal
junyoung.chung@umontreal.ca

Kyunghyun Cho
New York University

Yoshua Bengio
Université de Montréal
CIFAR Senior Fellow

Abstract

The existing machine translation systems, whether phrase-based or neural, have relied almost exclusively on word-level modelling with explicit segmentation. In this paper, we ask a fundamental question: can neural machine translation generate a character sequence without any explicit segmentation? To answer this question, we evaluate an attention-based encoder–decoder with a subword-level encoder and a character-level decoder on four language pairs–En-Cs, En-De, En-Ru and En-Fi–using the parallel corpora from WMT'15. Our experiments show that the models with a character-level decoder outperform the ones with a subword-level decoder on all of the four language pairs. Furthermore, the ensembles of neural models with a character-level decoder outperform the state-of-the-art non-neural machine translation systems on En-Cs, En-De and En-Fi and perform comparably on En-Ru.

1 Introduction

The existing machine translation systems have relied almost exclusively on word-level modelling with explicit segmentation. This is mainly due to the issue of data sparsity which becomes much more severe, especially for n-grams, when a sentence is represented as a sequence of characters rather than words, as the length of the sequence grows significantly. In addition to data sparsity, we often have a priori belief that a word, or its segmented-out lexeme, is a basic unit of meaning, making it natural to approach translation as mapping from a sequence of source-language words to a sequence of target-language words.

This has continued with the more recently proposed paradigm of neural machine translation, although neural networks do not suffer from character-level modelling and rather suffer from the issues specific to word-level modelling, such as the increased computational complexity from a very large target vocabulary (Jean et al., 2015; Luong et al., 2015b). Therefore, in this paper, we address a question of whether *neural machine translation can be done directly on a sequence of characters without any explicit word segmentation*.

To answer this question, we focus on representing the target side as a character sequence. We evaluate neural machine translation models with a character-level decoder on four language pairs from WMT'15 to make our evaluation as convincing as possible. We represent the source side as a sequence of subwords extracted using byte-pair encoding from Sennrich et al. (2015), and vary the target side to be either a sequence of subwords or characters. On the target side, we further design a novel recurrent neural network (RNN), called *bi-scale recurrent network*, that better handles multiple timescales in a sequence, and test it in addition to a naive, stacked recurrent neural network.

On all of the four language pairs–En-Cs, En-De, En-Ru and En-Fi–, the models with a character-level decoder outperformed the ones with a subword-level decoder. We observed a similar trend with the ensemble of each of these configurations, outperforming both the previous best neural and non-neural translation systems on En-Cs, En-De and En-Fi, while achieving a comparable result on En-Ru. We find these results to be a strong evidence that neural machine translation can indeed learn to translate at the character-level and that in fact, it benefits from doing so.

2 Neural Machine Translation

Neural machine translation refers to a recently proposed approach to machine translation (Cho et

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1693–1703,
Berlin, Germany, August 7-12, 2016.

al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015). This approach aims at building an end-to-end neural network that takes as input a source sentence $X = (x_1, \ldots, x_{T_x})$ and outputs its translation $Y = (y_1, \ldots, y_{T_y})$, where x_t and $y_{t'}$ are respectively source and target symbols. This neural network is constructed as a composite of an encoder network and a decoder network.

The encoder network encodes the input sentence X into its continuous representation. In this paper, we closely follow the neural translation model proposed in Bahdanau et al. (2015) and use a bidirectional recurrent neural network, which consists of two recurrent neural networks. The forward network reads the input sentence in a forward direction: $\overrightarrow{\mathbf{z}}_t = \overrightarrow{\phi}(e_x(x_t), \overrightarrow{\mathbf{z}}_{t-1})$, where $e_x(x_t)$ is a continuous embedding of the t-th input symbol, and ϕ is a recurrent activation function. Similarly, the reverse network reads the sentence in a reverse direction (right to left): $\overleftarrow{\mathbf{z}}_t = \overleftarrow{\phi}(e_x(x_t), \overleftarrow{\mathbf{z}}_{t+1})$. At each location in the input sentence, we concatenate the hidden states from the forward and reverse RNNs to form a context set $C = \{\mathbf{z}_1, \ldots, \mathbf{z}_{T_x}\}$, where $\mathbf{z}_t = \left[\overrightarrow{\mathbf{z}}_t; \overleftarrow{\mathbf{z}}_t\right]$.

Then the decoder computes the conditional distribution over all possible translations based on this context set. This is done by first rewriting the conditional probability of a translation: $\log p(Y|X) = \sum_{t'=1}^{T_y} \log p(y_{t'}|y_{<t'}, X)$. For each conditional term in the summation, the decoder RNN updates its hidden state by

$$\mathbf{h}_{t'} = \phi(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, \mathbf{c}_{t'}), \quad (1)$$

where e_y is the continuous embedding of a target symbol. $\mathbf{c}_{t'}$ is a context vector computed by a soft-alignment mechanism:

$$\mathbf{c}_{t'} = f_{\text{align}}(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, C)). \quad (2)$$

The soft-alignment mechanism f_{align} weights each vector in the context set C according to its relevance given what has been translated. The weight of each vector $\mathbf{z}_t$ is computed by

$$\alpha_{t,t'} = \frac{1}{Z} e^{f_{\text{score}}(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, \mathbf{z}_t)}, \quad (3)$$

where f_{score} is a parametric function returning an unnormalized score for $\mathbf{z}_t$ given $\mathbf{h}_{t'-1}$ and $y_{t'-1}$. We use a feedforward network with a single hidden layer in this paper.[1] Z is a normalization constant: $Z = \sum_{k=1}^{T_x} e^{f_{\text{score}}(e_y(y_{t'-1}), \mathbf{h}_{t'-1}, \mathbf{z}_k)}$. This procedure can be understood as computing the alignment probability between the t'-th target symbol and t-th source symbol.

The hidden state $\mathbf{h}_{t'}$, together with the previous target symbol $y_{t'-1}$ and the context vector $\mathbf{c}_{t'}$, is fed into a feedforward neural network to result in the conditional distribution:

$$p(y_{t'} \mid y_{<t'}, X) \propto e^{f_{\text{out}}^{y_{t'}}(e_y(y_{t'-1}), \mathbf{h}_{t'}, \mathbf{c}_{t'})}. \quad (4)$$

The whole model, consisting of the encoder, decoder and soft-alignment mechanism, is then tuned end-to-end to minimize the negative log-likelihood using stochastic gradient descent.

3 Towards Character-Level Translation

3.1 Motivation

Let us revisit how the source and target sentences (X and Y) are represented in neural machine translation. For the source side of any given training corpus, we scan through the whole corpus to build a vocabulary V_x of unique tokens to which we assign integer indices. A source sentence X is then built as a sequence of the indices of such tokens belonging to the sentence, i.e., $X = (x_1, \ldots, x_{T_x})$, where $x_t \in \{1, 2, \ldots, |V_x|\}$. The target sentence is similarly transformed into a target sequence of integer indices.

Each token, or its index, is then transformed into a so-called one-hot vector of dimensionality $|V_x|$. All but one elements of this vector are set to 0. The only element whose index corresponds to the token's index is set to 1. This one-hot vector is the one which any neural machine translation model sees. The embedding function, e_x or e_y, is simply the result of applying a linear transformation (the embedding matrix) to this one-hot vector.

The important property of this approach based on one-hot vectors is that the neural network is oblivious to the underlying semantics of the tokens. To the neural network, each and every token in the vocabulary is equal distance away from every other token. The semantics of those tokens are simply *learned* (into the embeddings) to maximize the translation quality, or the log-likelihood of the model.

This property allows us great freedom in the choice of tokens' unit. Neural networks have been shown to work well with word tokens (Bengio et al., 2001; Schwenk, 2007; Mikolov et al., 2010) but also with finer units, such as subwords (Sennrich et al., 2015; Botha and Blunsom, 2014; Lu-

[1] For other possible implementations, see (Luong et al., 2015a).

ong et al., 2013) as well as symbols resulting from compression/encoding (Chitnis and DeNero, 2015). Although there have been a number of previous research reporting the use of neural networks with characters (see, e.g., Mikolov et al. (2012) and Santos and Zadrozny (2014)), the dominant approach has been to preprocess the text into a sequence of symbols, each associated with a sequence of characters, after which the neural network is presented with those symbols rather than with characters.

More recently in the context of neural machine translation, two research groups have proposed to directly use characters. Kim et al. (2015) proposed to represent each word not as a single integer index as before, but as a sequence of characters, and use a convolutional network followed by a highway network (Srivastava et al., 2015) to extract a continuous representation of the word. This approach, which effectively replaces the embedding function e_x, was adopted by Costa-Jussà and Fonollosa (2016) for neural machine translation. Similarly, Ling et al. (2015b) use a bidirectional recurrent neural network to replace the embedding functions e_x and e_y to respectively encode a character sequence to and from the corresponding continuous word representation. A similar, but slightly different approach was proposed by Lee et al. (2015), where they explicitly mark each character with its relative location in a word (e.g., "B"eginning and "I"ntermediate).

Despite the fact that these recent approaches work at the level of characters, it is less satisfying that they all rely on knowing how to segment characters into words. Although it is generally easy for languages like English, this is not always the case. This word segmentation procedure can be as simple as tokenization followed by some punctuation normalization, but also can be as complicated as morpheme segmentation requiring a separate model to be trained in advance (Creutz and Lagus, 2005; Huang and Zhao, 2007). Furthermore, these segmentation[2] steps are often tuned or designed separately from the ultimate objective of translation quality, potentially contributing to a suboptimal quality.

Based on this observation and analysis, in this paper, we ask ourselves and the readers a question which should have been asked much earlier: *Is it possible to do character-level translation without any explicit segmentation?*

3.2 Why Word-Level Translation?

(1) Word as a Basic Unit of Meaning A word can be understood in two different senses. In the abstract sense, a word is a basic unit of meaning (lexeme), and in the other sense, can be understood as a "concrete word as used in a sentence." (Booij, 2012). A word in the former sense turns into that in the latter sense via a process of morphology, including inflection, compounding and derivation. These three processes do alter the meaning of the lexeme, but often it stays close to the original meaning. Because of this view of words as basic units of meaning (either in the form of lexemes or derived form) from linguistics, much of previous work in natural language processing has focused on using words as basic units of which a sentence is encoded as a sequence. Also, the potential difficulty in finding a mapping between a word's character sequence and meaning[3] has likely contributed to this trend toward word-level modelling.

(2) Data Sparsity There is a further technical reason why much of previous research on machine translation has considered words as a basic unit. This is mainly due to the fact that major components in the existing translation systems, such as language models and phrase tables, are a count-based estimator of probabilities. In other words, a probability of a subsequence of symbols, or pairs of symbols, is estimated by counting the number of its occurrences in a training corpus. This approach severely suffers from the issue of data sparsity, which is due to a large state space which grows exponentially w.r.t. the length of subsequences while growing only linearly w.r.t. the corpus size. This poses a great challenge to character-level modelling, as any subsequence will be on average 4–5 times longer when characters, instead of words, are used. Indeed, Vilar et al. (2007) reported worse performance when the character sequence was directly used by a phrase-based machine translation system. More recently, Neubig et al. (2013) proposed a method to improve character-level translation with phrase-based translation systems, however, with only a limited success.

[2]From here on, the term *segmentation* broadly refers to any method that splits a given character sequence into a sequence of subword symbols.

[3]For instance, "quit", "quite" and "quiet" are one edit-distance away from each other but have distinct meanings.

(3) Vanishing Gradient Specifically to neural machine translation, a major reason behind the wide adoption of word-level modelling is due to the difficulty in modelling long-term dependencies with recurrent neural networks (Bengio et al., 1994; Hochreiter, 1998). As the lengths of the sentences on both sides grow when they are represented in characters, it is easy to believe that there will be more long-term dependencies that must be captured by the recurrent neural network for successful translation.

3.3 Why Character-Level Translation?

Why *not* Word-Level Translation? The most pressing issue with word-level processing is that we do not have a perfect word segmentation algorithm for any one language. A perfect segmentation algorithm needs to be able to segment any given sentence into a sequence of lexemes and morphemes. This problem is however a difficult problem on its own and often requires decades of research (see, e.g., Creutz and Lagus (2005) for Finnish and other morphologically rich languages and Huang and Zhao (2007) for Chinese). Therefore, many opt to using either a rule-based tokenization approach or a suboptimal, but still available, learning based segmentation algorithm.

The outcome of this naive, sub-optimal segmentation is that the vocabulary is often filled with many similar words that share a lexeme but have different morphology. For instance, if we apply a simple tokenization script to an English corpus, "run", "runs", "ran" and "running" are all separate entries in the vocabulary, while they clearly share the same lexeme "run". This prevents any machine translation system, in particular neural machine translation, from modelling these morphological variants efficiently.

More specifically in the case of neural machine translation, each of these morphological variants– "run", "runs", "ran" and "running"– will be assigned a d-dimensional word vector, leading to four independent vectors, while it is clear that if we can segment those variants into a lexeme and other morphemes, we can model them more efficiently. For instance, we can have a d-dimensional vector for the lexeme "run" and much smaller vectors for "s" and"ing". Each of those variants will be then a composite of the lexeme vector (shared across these variants) and morpheme vectors (shared across words sharing the same suffix, for example) (Botha and Blunsom, 2014). This makes use of distributed representation, which generally yields better generalization, but seems to require an optimal segmentation, which is unfortunately almost never available.

In addition to inefficiency in modelling, there are two additional negative consequences from using (unsegmented) words. First, the translation system cannot generalize well to novel words, which are often mapped to a token reserved for an unknown word. This effectively ignores any meaning or structure of the word to be incorporated when translating. Second, even when a lexeme is common and frequently observed in the training corpus, its morphological variant may not be. This implies that the model sees this specific, rare morphological variant much less and will not be able to translate it well. However, if this rare morphological variant shares a large part of its spelling with other more common words, it is desirable for a machine translation system to exploit those common words when translating those rare variants.

Why Character-Level Translation? All of these issues can be addressed to certain extent by directly modelling characters. Although the issue of data sparsity arises in character-level translation, it is elegantly addressed by using a parametric approach based on recurrent neural networks instead of a non-parametric count-based approach. Furthermore, in recent years, we have learned how to build and train a recurrent neural network that can well capture long-term dependencies by using more sophisticated activation functions, such as long short-term memory (LSTM) units (Hochreiter and Schmidhuber, 1997) and gated recurrent units (Cho et al., 2014).

Kim et al. (2015) and Ling et al. (2015a) recently showed that by having a neural network that converts a character sequence into a word vector, we avoid the issues from having many morphological variants appearing as separate entities in a vocabulary. This is made possible by sharing the character-to-word neural network across all the unique tokens. A similar approach was applied to machine translation by Ling et al. (2015b).

These recent approaches, however, still rely on the availability of a good, if not optimal, segmentation algorithm. Ling et al. (2015b) indeed states that "[m]uch of the prior information regarding morphology, cognates and rare word translation

among others, should be incorporated".

It however becomes unnecessary to consider these prior information, if we use a neural network, be it recurrent, convolution or their combination, directly on the unsegmented character sequence. The possibility of using a sequence of unsegmented characters has been studied over many years in the field of deep learning. For instance, Mikolov et al. (2012) and Sutskever et al. (2011) trained a recurrent neural network language model (RNN-LM) on character sequences. The latter showed that it is possible to generate sensible text sequences by simply sampling a character at a time from this model. More recently, Zhang et al. (2015) and Xiao and Cho (2016) successfully applied a convolutional net and a convolutional-recurrent net respectively to character-level document classification without any explicit segmentation. Gillick et al. (2015) further showed that it is possible to train a recurrent neural network on unicode bytes, instead of characters or words, to perform part-of-speech tagging and named entity recognition.

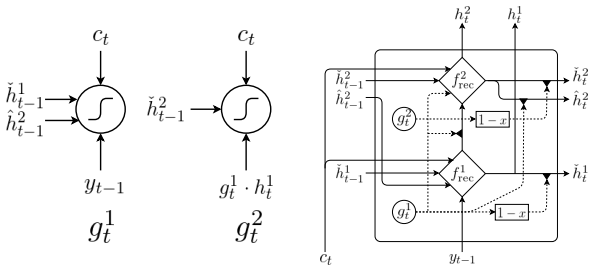
These previous works suggest the possibility of applying neural networks for the task of machine translation, which is often considered a substantially more difficult problem compared to document classification and language modelling.

3.4 Challenges and Questions

There are two overlapping sets of challenges for the source and target sides. On the source side, it is unclear how to build a neural network that learns a highly nonlinear mapping from a spelling to the meaning of a sentence.

On the target side, there are two challenges. The first challenge is the same one from the source side, as the decoder neural network needs to summarize what has been translated. In addition to this, the character-level modelling on the target side is more challenging, as the decoder network must be able to generate a long, coherent sequence of characters. This is a great challenge, as the size of the state space grows exponentially w.r.t. the number of symbols, and in the case of characters, it is often 300-1000 symbols long.

All these challenges should first be framed as questions; whether the current recurrent neural networks, which are already widely used in neural machine translation, are able to address these challenges as they are. In this paper, we aim at answering these *questions* empirically and focus on the challenges on the target side (as the target side shows both of the challenges).



(a) Gating units (b) One-step processing

Figure 1: Bi-scale recurrent neural network

4 Character-Level Translation

In this paper, we try to answer the questions posed earlier by testing two different types of recurrent neural networks on the target side (decoder).

First, we test an existing recurrent neural network with gated recurrent units (GRUs). We call this decoder a *base* decoder.

Second, we build a novel two-layer recurrent neural network, inspired by the gated-feedback network from Chung et al. (2015), called a *bi-scale* recurrent neural network. We design this network to facilitate capturing two timescales, motivated by the fact that characters and words may work at two separate timescales.

We choose to test these two alternatives for the following purposes. Experiments with the base decoder will clearly answer whether the existing neural network is enough to handle character-level decoding, which has not been properly answered in the context of machine translation. The alternative, the bi-scale decoder, is tested in order to see whether it is possible to design a better decoder, if the answer to the first question is positive.

4.1 Bi-Scale Recurrent Neural Network

In this proposed bi-scale recurrent neural network, there are two sets of hidden units, $\mathbf{h}^1$ and $\mathbf{h}^2$. They contain the same number of units, i.e., $\dim(\mathbf{h}^1) = \dim(\mathbf{h}^2)$. The first set $\mathbf{h}^1$ models a fast-changing timescale (thereby, a *faster layer*), and $\mathbf{h}^2$ a slower timescale (thereby, a *slower layer*). For each hidden unit, there is an associated gating unit, to which we refer by $\mathbf{g}^1$ and $\mathbf{g}^2$. For the description below, we use $y_{t'-1}$ and $\mathbf{c}_{t'}$ for the previous target symbol and the context vector (see Eq. (2)), respectively.

Let us start with the faster layer. The faster layer outputs two sets of activations, a normal output $\mathbf{h}^1_{t'}$ and its gated version $\check{\mathbf{h}}^1_{t'}$. The activation of the faster layer is computed by

$$\mathbf{h}^1_{t'} = \tanh\left(\mathbf{W}^{h^1}\left[e_y(y_{t'-1}); \check{\mathbf{h}}^1_{t'-1}; \hat{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right),$$

where $\check{\mathbf{h}}^1_{t'-1}$ and $\hat{\mathbf{h}}^2_{t'-1}$ are the gated activations of the faster and slower layers respectively. These gated activations are computed by

$$\check{\mathbf{h}}^1_{t'} = (1 - \mathbf{g}^1_{t'}) \odot \mathbf{h}^1_{t'}, \;\; \hat{\mathbf{h}}^2_{t'} = \mathbf{g}^1_{t'} \odot \mathbf{h}^2_{t'}.$$

In other words, the faster layer's activation is based on the adaptive combination of the faster and slower layers' activations from the previous time step. Whenever the faster layer determines that it needs to reset, i.e., $\mathbf{g}^1_{t'-1} \approx 1$, the next activation will be determined based more on the slower layer's activation.

The faster layer's gating unit is computed by

$$\mathbf{g}^1_{t'} = \sigma\left(\mathbf{W}^{g^1}\left[e_y(y_{t'-1}); \check{\mathbf{h}}^1_{t'-1}; \hat{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right),$$

where σ is a sigmoid function.

The slower layer also outputs two sets of activations, a normal output $\mathbf{h}^2_{t'}$ and its gated version $\check{\mathbf{h}}^2_{t'}$. These activations are computed as follows:

$$\mathbf{h}^2_{t'} = (1 - \mathbf{g}^1_{t'}) \odot \mathbf{h}^2_{t'-1} + \mathbf{g}^1_{t'} \odot \tilde{\mathbf{h}}^2_{t'},$$
$$\check{\mathbf{h}}^2_{t'} = (1 - \mathbf{g}^2_{t'}) \odot \mathbf{h}^2_{t'},$$

where $\tilde{\mathbf{h}}^2_{t'}$ is a candidate activation. The slower layer's gating unit $\mathbf{g}^2_{t'}$ is computed by

$$\mathbf{g}^2_{t'} = \sigma\left(\mathbf{W}^{g^2}\left[(\mathbf{g}^1_{t'} \odot \mathbf{h}^1_{t'}); \check{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right).$$

This adaptive leaky integration based on the gating unit from the faster layer has a consequence that the slower layer updates its activation only when the faster layer resets. This puts a soft constraint that the faster layer runs at a faster rate by preventing the slower layer from updating while the faster layer is processing a current chunk.

The candidate activation is then computed by

$$\tilde{\mathbf{h}}^2_{t'} = \tanh\left(\mathbf{W}^{h^2}\left[(\mathbf{g}^1_{t'} \odot \mathbf{h}^1_{t'}); \check{\mathbf{h}}^2_{t'-1}; \mathbf{c}_{t'}\right]\right). \quad (5)$$

$\check{\mathbf{h}}^2_{t'-1}$ indicates the *reset* activation from the previous time step, similarly to what happened in the faster layer, and $\mathbf{c}_{t'}$ is the input from the context.

According to $\mathbf{g}^1_{t'} \odot \mathbf{h}^1_{t'}$ in Eq. (5), the faster layer influences the slower layer, only when the faster layer has finished processing the current chunk and is about to *reset* itself ($\mathbf{g}^1_{t'} \approx 1$). In other words, the slower layer does not receive any input from the faster layer, until the faster layer has quickly processed the current chunk, thereby running at a slower rate than the faster layer does.

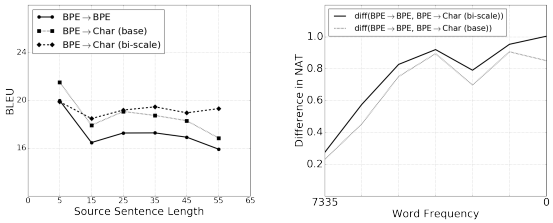


Figure 2: (left) The BLEU scores on En-Cs w.r.t. the length of source sentences. (right) The difference of word negative log-probabilities between the subword-level decoder and either of the character-level base or bi-scale decoder.

At each time step, the final output of the proposed bi-scale recurrent neural network is the concatenation of the output vectors of the faster and slower layers, i.e., $\left[\mathbf{h}^1; \mathbf{h}^2\right]$. This concatenated vector is used to compute the probability distribution over all the symbols in the vocabulary, as in Eq. (4). See Fig. 1 for graphical illustration.

5 Experiment Settings

For evaluation, we represent a source sentence as a sequence of subword symbols extracted by byte-pair encoding (BPE, Sennrich et al. (2015)) and a target sentence either as a sequence of BPE-based symbols or as a sequence of characters.

Corpora and Preprocessing We use all available parallel corpora for four language pairs from WMT'15: En-Cs, En-De, En-Ru and En-Fi. They consist of 12.1M, 4.5M, 2.3M and 2M sentence pairs, respectively. We tokenize each corpus using a tokenization script included in Moses.[4] We only use the sentence pairs, when the source side is up to 50 subword symbols long and the target side is either up to 100 subword symbols or 500 characters. We do not use any monolingual corpus.

For all the pairs other than En-Fi, we use newstest-2013 as a development set, and newstest-2014 (Test$_1$) and newstest-2015 (Test$_2$) as test sets. For En-Fi, we use newsdev-2015 and newstest-2015 as development and test sets, respectively.

[4]Although tokenization is not necessary for character-level modelling, we tokenize the all target side corpora to make comparison against word-level modelling easier.

		Src	Trgt	Depth	Attention $\mathbf{h}^1$	Attention $\mathbf{h}^2$	Model	Development Single	Development Ens	Test_1 Single	Test_1 Ens	Test_2 Single	Test_2 Ens
En-De	(a)	BPE	BPE	1	✓		Base	20.78	–	19.98	–	21.72	–
	(b)			2	✓	✓		$21.26^{21.45}_{20.62}$	23.49	$20.47^{20.88}_{19.30}$	23.10	$22.02^{22.21}_{21.35}$	24.83
	(c)		Char	2		✓	Base	$21.57^{21.88}_{20.88}$	23.14	$\mathbf{21.33}^{21.56}_{19.82}$	23.11	$\mathbf{23.45}^{23.91}_{21.72}$	25.24
	(d)			2	✓	✓		20.31	–	19.70	–	21.30	–
	(e)			2		✓	Bi-S	$21.29^{21.43}_{21.13}$	23.05	$21.25^{21.47}_{20.62}$	23.04	$23.06^{23.47}_{22.85}$	25.44
	(f)			2	✓	✓		20.78	–	20.19	–	22.26	–
	(g)			2	✓			20.08	–	19.39	–	20.94	–
	State-of-the-art Non-Neural Approach*							–		20.60(1)		24.00(2)	
En-Cs	(h)	BPE	BPE	2	✓	✓	Base	$16.12^{16.96}_{15.96}$	19.21	$17.16^{17.68}_{16.38}$	20.79	$14.63^{15.09}_{14.26}$	17.61
	(i)		Char	2		✓	Base	$17.68^{17.78}_{17.39}$	19.52	$19.25^{19.55}_{18.89}$	21.95	$\mathbf{16.98}^{17.17}_{16.81}$	18.92
	(j)			2		✓	Bi-S	$17.62^{17.93}_{17.43}$	19.83	$\mathbf{19.27}^{19.53}_{19.15}$	22.15	$16.86^{17.10}_{16.68}$	18.93
	State-of-the-art Non-Neural Approach*							–		21.00(3)		18.20(4)	
En-Ru	(k)	BPE	BPE	2	✓	✓	Base	$18.56^{18.70}_{18.26}$	21.17	$25.30^{25.40}_{24.95}$	29.26	$19.72^{20.29}_{19.02}$	22.96
	(l)		Char	2		✓	Base	$18.56^{18.87}_{18.39}$	20.53	$\mathbf{26.00}^{26.07}_{25.04}$	29.37	$\mathbf{21.10}^{21.24}_{20.14}$	23.51
	(m)			2		✓	Bi-S	$18.30^{18.54}_{17.88}$	20.53	$25.59^{25.76}_{24.57}$	29.26	$20.73^{21.02}_{19.97}$	23.75
	State-of-the-art Non-Neural Approach*							–		28.70(5)		24.30(6)	
En-Fi	(n)	BPE	BPE	2	✓	✓	Base	$9.61^{10.02}_{9.24}$	11.92	–	–	$8.97^{9.17}_{8.88}$	11.73
	(o)		Char	2		✓	Base	$11.19^{11.55}_{11.09}$	13.72	–	–	$\mathbf{10.93}^{11.56}_{10.11}$	13.48
	(p)			2		✓	Bi-S	$10.73^{11.04}_{10.40}$	13.39	–	–	$10.24^{10.63}_{9.71}$	13.32
	State-of-the-art Non-Neural Approach*							–		–		12.70(7)	

Table 1: BLEU scores of the subword-level, character-level base and character-level bi-scale decoders for both single models and ensembles. The best scores among the single models per language pair are bold-faced, and those among the ensembles are underlined. When available, we report the median value, and the minimum and maximum values as a subscript and a superscript, respectively. (*) `http://matrix.statmt.org/` as of 11 March 2016 (constrained only). (1) Freitag et al. (2014). (2, 6) Williams et al. (2015). (3, 5) Durrani et al. (2014). (4) Haddow et al. (2015). (7) Rubino et al. (2015).

Models and Training We test three models settings: (1) BPE→BPE, (2) BPE→Char (base) and (3) BPE→Char (bi-scale). The latter two differ by the type of recurrent neural network we use. We use GRUs for the encoder in all the settings. We used GRUs for the decoders in the first two settings, (1) and (2), while the proposed bi-scale recurrent network was used in the last setting, (3). The encoder has 512 hidden units for each direction (forward and reverse), and the decoder has 1024 hidden units per layer.

We train each model using stochastic gradient descent with Adam (Kingma and Ba, 2014). Each update is computed using a minibatch of 128 sentence pairs. The norm of the gradient is clipped with a threshold 1 (Pascanu et al., 2013).

Decoding and Evaluation We use beamsearch to approximately find the most likely translation given a source sentence. The beam widths are 5 and 15 respectively for the subword-level and character-level decoders. They were chosen based on the translation quality on the development set. The translations are evaluated using BLEU.[5]

Multilayer Decoder and Soft-Alignment Mechanism When the decoder is a multilayer recurrent neural network (including a stacked network as well as the proposed bi-scale network), the decoder outputs multiple hidden vectors– $\left\{\mathbf{h}^1, \ldots, \mathbf{h}^L\right\}$ for L layers, at a time. This allows an extra degree of freedom in the soft-alignment mechanism (f_{score} in Eq. (3)). We evaluate using alternatives, including (1) using only $\mathbf{h}^L$ (slower layer) and (2) using all of them (concatenated).

Ensembles We also evaluate an ensemble of neural machine translation models and compare its performance against the state-of-the-art phrase-based translation systems on all four language pairs. We decode from an ensemble by taking the average of the output probabilities at each step.

6 Quantitative Analysis

Slower Layer for Alignment On En-De, we test which layer of the decoder should be used

[5]We used the multi-bleu.perl script from Moses.

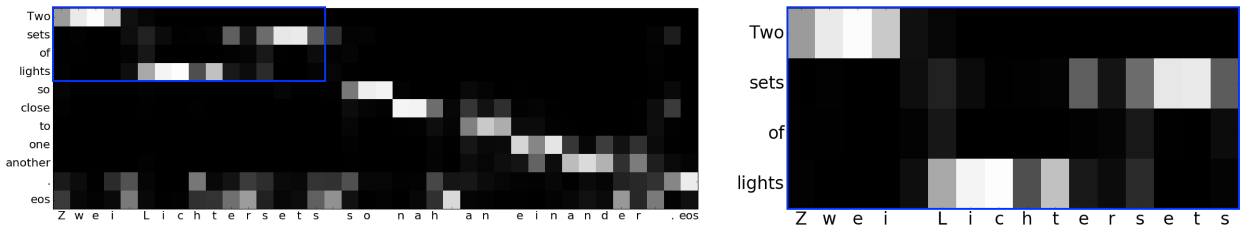


Figure 3: Alignment matrix of a test example from En-De using the BPE→Char (bi-scale) model.

for computing soft-alignments. In the case of subword-level decoder, we observed no difference between choosing any of the two layers of the decoder against using the concatenation of all the layers (Table 1 (a–b)) On the other hand, with the character-level decoder, we noticed an improvement when only the slower layer ($\mathbf{h}^2$) was used for the soft-alignment mechanism (Table 1 (c–g)). This suggests that the soft-alignment mechanism benefits by aligning a larger chunk in the target with a subword unit in the source, and we use only the slower layer for all the other language pairs.

Single Models In Table 1, we present a comprehensive report of the translation qualities of (1) subword-level decoder, (2) character-level base decoder and (3) character-level bi-scale decoder, for all the language pairs. We see that the both types of character-level decoder outperform the subword-level decoder for En-Cs and En-Fi quite significantly. On En-De, the character-level base decoder outperforms both the subword-level decoder and the character-level bi-scale decoder, validating the effectiveness of the character-level modelling. On En-Ru, among the single models, the character-level decoders outperform the subword-level decoder, but in general, we observe that all the three alternatives work comparable to each other.

These results clearly suggest that *it is indeed possible to do character-level translation without explicit segmentation.* In fact, what we observed is that character-level translation often surpasses the translation quality of word-level translation. Of course, we note once again that our experiment is restricted to using an unsegmented character sequence at the decoder only, and a further exploration toward replacing the source sentence with an unsegmented character sequence is needed.

Ensembles Each ensemble was built using eight independent models. The first observation we make is that in all the language pairs, neural machine translation performs comparably to, or often better than, the state-of-the-art non-neural translation system. Furthermore, the character-level decoders outperform the subword-level decoder in all the cases.

7 Qualitative Analysis

(1) Can the character-level decoder generate a long, coherent sentence? The translation in characters is dramatically longer than that in words, likely making it more difficult for a recurrent neural network to generate a coherent sentence in characters. This belief turned out to be false. As shown in Fig. 2 (left), there is no significant difference between the subword-level and character-level decoders, even though the lengths of the generated translations are generally 5–10 times longer in characters.

(2) Does the character-level decoder help with rare words? One advantage of character-level modelling is that it can model the composition of any character sequence, thereby better modelling rare morphological variants. We empirically confirm this by observing the growing gap in the average negative log-probability of words between the subword-level and character-level decoders as the frequency of the words decreases. This is shown in Fig. 2 (right) and explains one potential cause behind the success of character-level decoding in our experiments (we define $\mathrm{diff}(x, y) = x - y$).

(3) Can the character-level decoder soft-align between a source word and a target character? In Fig. 3 (left), we show an example soft-alignment of a source sentence, "Two sets of light so close to one another". It is clear that the character-level translation model well captured the alignment between the source subwords and target characters. We observe that the character-level decoder correctly aligns to "lights" and "sets of" when generating a German compound word

"Lichtersets" (see Fig. 3 (right) for the zoomed-in version). This type of behaviour happens similarly between "one another" and "einander". Of course, this does not mean that there exists an alignment between a source word and a target character. Rather, this suggests that the internal state of the character-level decoder, the base or bi-scale, well captures the meaningful chunk of characters, allowing the model to map it to a larger chunk (subword) in the source.

(4) How fast is the decoding speed of the character-level decoder? We evaluate the decoding speed of subword-level base, character-level base and character-level bi-scale decoders on newstest2013 corpus (En-De) with a single Titan X GPU. The subword-level base decoder generates 31.9 words per second, and the character-level base decoder and character-level bi-scale decoder generate 27.5 words per second and 25.6 words per second, respectively. Note that this is evaluated in an online setting, performing consecutive translation, where only one sentence is translated at a time. Translating in a batch setting could differ from these results.

8 Conclusion

In this paper, we addressed a fundamental question on whether a recently proposed neural machine translation system can directly handle translation at the level of characters without any word segmentation. We focused on the target side, in which a decoder was asked to generate one character at a time, while soft-aligning between a target character and a source subword. Our extensive experiments, on four language pairs–En-Cs, En-De, En-Ru and En-Fi– strongly suggest that it is indeed possible for neural machine translation to translate at the level of characters, and that it actually benefits from doing so.

Our result has one limitation that we used subword symbols in the source side. However, this has allowed us a more fine-grained analysis, but in the future, a setting where the source side is also represented as a character sequence must be investigated.

Acknowledgments

The authors would like to thank the developers of Theano (Team et al., 2016). We acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs, CIFAR and Samsung. KC thanks the support by Facebook, Google (Google Faculty Award 2016) and NVIDIA (GPU Center of Excellence 2015-2016). JC thanks Orhan Firat for his constructive feedbacks.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2001. A neural probabilistic language model. In *Advances in Neural Information Processing Systems*, pages 932–938.

Geert Booij. 2012. *The grammar of words: An introduction to linguistic morphology*. Oxford University Press.

Jan A Botha and Phil Blunsom. 2014. Compositional morphology for word representations and language modelling. In *ICML 2014*.

Rohan Chitnis and John DeNero. 2015. Variable-length word encodings for neural translation models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2088–2093.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, October.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2015. Gated feedback recurrent neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*.

Marta R Costa-Jussà and José AR Fonollosa. 2016. Character-based neural machine translation. *arXiv preprint arXiv:1603.00810*.

Mathias Creutz and Krista Lagus. 2005. *Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0*. Helsinki University of Technology.

Nadir Durrani, Barry Haddow, Philipp Koehn, and Kenneth Heafield. 2014. Edinburgh's phrase-based machine translation systems for wmt-14. In *Proceedings of the ACL 2014 Ninth Workshop on Statistical Machine Translation, Baltimore, MD, USA*, pages 97–104.

Markus Freitag, Stephan Peitz, Joern Wuebker, Hermann Ney, Matthias Huck, Rico Sennrich, Nadir Durrani, Maria Nadejde, Philip Williams, Philipp Koehn, et al. 2014. Eu-bridge mt: Combined machine translation.

Dan Gillick, Cliff Brunk, Oriol Vinyals, and Amarnag Subramanya. 2015. Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*.

Barry Haddow, Matthias Huck, Alexandra Birch, Nikolay Bogoychev, and Philipp Koehn. 2015. The edinburgh/jhu phrase-based machine translation systems for wmt 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 126–133.

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

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

Changning Huang and Hai Zhao. 2007. Chinese word segmentation: A decade review. *Journal of Chinese Information Processing*, 21(3):8–20.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. 2015. Character-aware neural language models. *arXiv preprint arXiv:1508.06615*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hyoung-Gyu Lee, JaeSong Lee, Jun-Seok Kim, and Chang-Ki Lee. 2015. Naver machine translation system for wat 2015. In *Proceedings of the 2nd Workshop on Asian Translation (WAT2015)*, pages 69–73.

Wang Ling, Tiago Luís, Luís Marujo, Ramón Fernandez Astudillo, Silvio Amir, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015a. Finding function in form: Compositional character models for open vocabulary word representation. *arXiv preprint arXiv:1508.02096*.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015b. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Thang Luong, Richard Socher, and Christopher D Manning. 2013. Better word representations with recursive neural networks for morphology. In *CoNLL*, pages 104–113.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015a. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3.

Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and J Cernocky. 2012. Subword language modeling with neural networks. *Preprint*.

Graham Neubig, Taro Watanabe, Shinsuke Mori, and Tatsuya Kawahara. 2013. Substring-based machine translation. *Machine translation*, 27(2):139–166.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv preprint arXiv:1312.6026*.

Raphael Rubino, Tommi Pirinen, Miquel Espla-Gomis, N Ljubešic, Sergio Ortiz Rojas, Vassilis Papavassiliou, Prokopis Prokopidis, and Antonio Toral. 2015. Abu-matran at wmt 2015 translation task: Morphological segmentation and web crawling. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 184–191.

Cicero D Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1818–1826.

Holger Schwenk. 2007. Continuous space language models. *Computer Speech & Language*, 21(3):492–518.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Advances in Neural Information Processing Systems*, pages 2368–2376.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*, pages 1017–1024.

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

The Theano Development Team, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, et al. 2016. Theano: A python framework for fast computation of mathematical expressions. *arXiv preprint arXiv:1605.02688*.

David Vilar, Jan-T Peter, and Hermann Ney. 2007. Can we translate letters? In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 33–39. Association for Computational Linguistics.

Philip Williams, Rico Sennrich, Maria Nadejde, Matthias Huck, and Philipp Koehn. 2015. Edinburgh's syntax-based systems at wmt 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 199–209.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers. *arXiv preprint arXiv:1602.00367*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, pages 649–657.

Target-Side Context for Discriminative Models in Statistical Machine Translation

Aleš Tamchyna[1,2], **Alexander Fraser**[1], **Ondřej Bojar**[2] **and Marcin Junczys-Dowmunt**[3]
[1]LMU Munich, Munich, Germany
[2]Charles University in Prague, Prague, Czech Republic
[3]Adam Mickiewicz University in Poznań, Poznań, Poland
{tamchyna,bojar}@ufal.mff.cuni.cz fraser@cis.uni-muenchen.de junczys@amu.edu.pl

Abstract

Discriminative translation models utilizing source context have been shown to help statistical machine translation performance. We propose a novel extension of this work using target context information. Surprisingly, we show that this model can be efficiently integrated directly in the decoding process. Our approach scales to large training data sizes and results in consistent improvements in translation quality on four language pairs. We also provide an analysis comparing the strengths of the baseline source-context model with our extended source-context and target-context model and we show that our extension allows us to better capture morphological coherence. Our work is freely available as part of Moses.

1 Introduction

Discriminative lexicons address some of the core challenges of phrase-based MT (PBMT) when translating to morphologically rich languages, such as Czech, namely sense disambiguation and morphological coherence. The first issue is semantic: given a source word or phrase, which of its possible meanings (i.e., which stem or lemma) should we choose? Previous work has shown that this can be addressed using a discriminative lexicon. The second issue has to do with morphology (and syntax): given that we selected the correct meaning, which of its inflected surface forms is appropriate? In this work, we integrate such a model directly into the SMT decoder. This enables our classifier to extract features not only from the full source sentence but also from a limited target-side context. This allows the model to not only help with semantics but also to improve morphological and syntactic coherence.

For *sense disambiguation*, source context is the main source of information, as has been shown in previous work (Vickrey et al., 2005), (Carpuat and Wu, 2007), (Gimpel and Smith, 2008) inter alia. Consider the first set of examples in Figure 1, produced by a strong baseline PBMT system. The English word "shooting" has multiple senses when translated into Czech: it may either be the act of firing a weapon or making a film. When the cue word "film" is close, the phrase-based model is able to use it in one phrase with the ambiguous "shooting", disambiguating correctly the translation. When we add a single word in between, the model fails to capture the relationship and the most frequent sense is selected instead. Wider source context information is required for correct disambiguation.

While word/phrase senses can usually be inferred from the source sentence, the correct *selection of surface forms* requires also information from the target. Note that we can obtain *some* information from the source. For example, an English subject is often translated into a Czech subject; in which case the Czech word should be in nominative case. But there are many decisions that happen during decoding which determine morphological and syntactic properties of words – verbs can have translations which differ in valency frames, they may be translated in either active or passive voice (in which case subject and object would be switched), nouns may have different possible translations which differ in gender, etc.

The correct selection of surface forms plays a crucial role in preserving meaning in morphologically rich languages because it is morphology rather than word order that expresses relations between words. (Word order tends to be

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1704–1714,
Berlin, Germany, August 7-12, 2016.

Input	PBMT Output	
shooting of the film .	natáčení filmu .	✓
	shooting$_{camera}$ *of_film* .	
shooting of the expensive film .	střelby na drahý film .	✗
	shootings$_{gun}$ *at expensive film* .	
the man saw a cat .	muž uviděl kočku .	✓
	man saw ***cat***$_{acc}$.	
the man saw a black cat .	muž spatřil černou kočku .	✓
	man saw ***black***$_{acc}$ ***cat***$_{acc}$.	
the man saw a yellowish cat .	muž spatřil nažloutlá kočka .	✗
	man saw ***yellowish***$_{nom}$ ***cat***$_{nom}$.	

Figure 1: Examples of problems of PBMT: lexical selection and morphological coherence. Each translation has a corresponding gloss in italics.

relatively free and driven more by semantic constraints rather than syntactic constraints.)

The language model is only partially able to capture this phenomenon. It has a limited scope and perhaps more seriously, it suffers from data sparsity. The units captured by both the phrase table and the LM are mere sequences of words. In order to estimate their probability, we need to observe them in the training data (many times, if the estimates should be reliable). However, the number of possible n-grams grows exponentially as we increase n, leading to unrealistic requirements on training data sizes. This implies that the current models can (and often do) miss relationships between words even within their theoretical scope.

The second set of sentences in Figure 1 demonstrates the problem of data sparsity for morphological coherence. While the phrase-based system can correctly transfer the morphological case of "cat" and even "black cat", the less usual "yellowish cat" is mistranslated into nominative case, even though the correct phrase "yellowish ||| nažloutlou" exists in the phrase table. A model with a suitable representation of two preceding words could easily infer the correct case in this example.

Our contributions are the following:

- We show that the addition of a feature-rich discriminative model significantly improves translation quality even for large data sizes and that target-side context information consistently further increases this improvement.
- We provide an analysis of the outputs which confirms that source-context features indeed help with semantic disambiguation (as is well known). Importantly, we also show that our novel use of target context improves morphological and syntactic coherence.
- In addition to extensive experimentation on translation from English to Czech, we also evaluate English to German, English to Polish and English to Romanian tasks, with improvements on translation quality in all tasks, showing that our work is broadly applicable.
- We describe several optimizations which allow target-side features to be used efficiently in the context of phrase-based decoding.
- Our implementation is freely available in the widely used open-source MT toolkit Moses, enabling other researchers to explore discriminative modelling with target context in MT.

2 Discriminative Model with Target-Side Context

Several different ways of using feature-rich models in MT have been proposed, see Section 6. We describe our approach in this section.

2.1 Model Definition

Let f be the source sentence and e its translation. We denote source-side phrases (given a particular phrasal segmentation) $(\bar{f}_1, \ldots, \bar{f}_m)$ and the individual words $(f_1, \ldots, f_n)$. We use a similar notation for target-side words/phrases.

For simplicity, let e_{prev}, e_{prev-1} denote the words preceding the current target phrase. Assuming target context size of two, we model the following probability distribution:

$$P(e|f) \propto \prod_{(\bar{e}_i, \bar{f}_i) \in (e,f)} P(\bar{e}_i | \bar{f}_i, f, e_{prev}, e_{prev-1}) \quad (1)$$

The probability of a translation is the product of phrasal translation probabilities which are conditioned on the source phrase, the full source sentence and several previous target words.

Let $\mathrm{GEN}(\bar{f}_i)$ be the set of possible translations of the source phrase $\bar{f}_i$ according to the phrase table. We also define a "feature vector" function $\mathrm{fv}(\bar{e}_i, \bar{f}_i, f, e_{prev}, e_{prev-1})$ which outputs a vector of features given the phrase pair and its context information. We also have a vector of feature weights w estimated from the training data. Then our model defines the phrasal translation probability simply as follows:

$$P(\bar{e}_i | \bar{f}_i, f, e_{prev}, e_{prev-1}) = \frac{\exp(w \cdot \mathrm{fv}(\bar{e}_i, \bar{f}_i, f, e_{prev}, e_{prev-1}))}{\sum_{\bar{e}' \in \mathrm{GEN}(\bar{f}_i)} \exp(w \cdot \mathrm{fv}(\bar{e}', \bar{f}_i, f, e_{prev}, e_{prev-1}))} \quad (2)$$

This definition implies that we have to locally normalize the classifier outputs so that they sum to one.

In PBMT, translations are usually scored by a log-linear model. Our classifier produces a single score (the conditional phrasal probability) which we add to the standard log-linear model as an additional feature. The MT system therefore does not have direct access to the *classifier* features, only to the final score.

2.2 Global Model

We use the Vowpal Wabbit (VW) classifier[1] in this work. Tamchyna et al. (2014) already integrated VW into Moses. We started from their implementation in order to carry out our work. Classifier features are divided into two "namespaces":

- **S.** Features that do not depend on the current phrasal translation (i.e., source- and target-context features).
- **T.** Features of the current phrasal translation.

We make heavy use of feature processing available in VW, namely quadratic feature expansions and label-dependent features. When generating features for a particular set of translations, we first create the *shared* features (in the namespace S). These only depend on (source and target) context and are therefore constant for all possible translations of a given phrase. (Note that target-side context naturally depends on the current partial translation. However, when we process the possible translations for a single source phrase, the target context is constant.)

Then for each translation, we extract its features and store them in the namespace T. Note that we do not provide a label (or class) to VW – it is up to these *translation* features to describe the target phrase. (And this is what is referred to as "label-dependent features" in VW.)

Finally, we add the Cartesian product between the two namespaces to the feature set: every *shared* feature is combined with every *translation* feature.

This setting allows us to train only a single, global model with powerful feature sharing. For example, thanks to the label-dependent format, we can decompose both the source phrase and the target phrase into words and have features such as `s_cat_t_kočka` which capture phrase-internal word translations. Predictions for rare phrase pairs are then more robust thanks to the rich statistics collected for these word-level feature pairs.

2.3 Extraction of Training Examples

Discriminative models in MT are typically trained by creating one training instance per extracted phrase from the entire training data. The target side of the extracted phrase is a positive label, and all other phrases observed aligned to the extracted phrase (anywhere in the training data) are the negative labels.

We train our model in a similar fashion: for each sentence in the parallel training data, we look at all possible phrasal segmentations. Then for each source span, we create a training example. We obtain the set of possible translations $\mathrm{GEN}(\bar{f})$ from the phrase table. Because we do not have actual classes, each translation is defined by its label-dependent features and we associate a loss with it: 0 loss for the correct translation and 1 for all others.

Because we train both our model and the standard phrase table on the same dataset, we use leaving-one-out in the classifier training to avoid

[1] `http://hunch.net/~vw/`

Feature Type	Configurations		
	Czech	German	Polish, Romanian
Source Indicator	f, l, l+t, t	f, l, l+t, t	l, t
Source Internal	f, f+a, f+p, l, l+t, t, a+p	f, f+a, f+p, l, l+t, t, a+p	l, l+a, l+p, t, a+p
Source Context	f (-3,3), l (-3,3), t (-5,5)	f (-3,3), l (-3,3), t (-5,5)	l (-3,3), t (-5,5)
Target Context	f (2), l (2), t (2), l+t (2)	f (2), l (2), t (2), l+t (2)	l (2), t (2)
Bilingual Context	—	l+t/l+t (2)	l+t/l+t (2)
Target Indicator	f, l, t	f, l, t	l, t
Target Internal	f, l, l+t, t	f, l, l+t, t	l, t

Table 1: List of used feature templates. Letter abbreviations refer to word factors: f (form), l (lemma), t (morphological tag), a (analytical function), p (lemma of dependency parent). Numbers in parentheses indicate context size.

over-fitting. We look at phrase counts and co-occurrence counts in the training data, we subtract one from the number of occurrences for the current source phrase, target phrase and the phrase pair. If the count goes to zero, we skip the training example. Without this technique, the classifier might learn to simply trust very long phrase pairs which were extracted from the same training sentence.

For target-side context features, we simply use the true (gold) target context. This leads to training which is similar to language model estimation; this model is somewhat similar to the neural joint model for MT (Devlin et al., 2014), but in our case implemented using a linear (maximum-entropy-like) model.

2.4 Training

We use Vowpal Wabbit in the `--csoaa_ldf mc` setting which reduces our multi-class problem to one-against-all binary classification. We use the logistic loss as our objective. We experimented with various settings of L2 regularization but were not able to get an improvement over not using regularization at all. We train each model with 10 iterations over the data.

We evaluate all of our models on a held-out set. We use the same dataset as for MT system tuning because it closely matches the domain of our test set. We evaluate model accuracy after each pass over the training data to detect over-fitting and we select the model with the highest held-out accuracy.

2.5 Feature Set

Our feature set requires some linguistic processing of the data. We use the factored MT setting (Koehn and Hoang, 2007) and we represent each type of information as an individual factor. On the source side, we use the word surface form, its lemma, morphological tag, analytical function (such as *Subj* for subjects) and the lemma of the parent node in the dependency parse tree. On the target side, we only use word lemmas and morphological tags.

Table 1 lists our feature sets for each language pair. We implemented *indicator* features for both the source and target side; these are simply concatenations of the words in the current phrase into a single feature. *Internal* features describe words within the current phrase. *Context* features are extracted either from a window of a fixed size around the current phrase (on the source side) or from a limited left-hand side context (on the target side). *Bilingual context* features are concatenations of target-side context words and their source-side counterparts (according to word alignment); these features are similar to bilingual tokens in bilingual LMs (Niehues et al., 2011). Each of our feature types can be configured to look at any individual factors or their combinations.

The features in Table 1 are divided into three sets. The first set contains label-independent (=shared) features which only depend on the source sentence. The second set contains shared features which depend on target-side context; these can only be used when VW is applied during decoding. We use target context size two in all our experiments.[2] Finally, the third set contains label-dependent features which describe the currently predicted phrasal translation.

[2]In preliminary experiments we found that using a single word was less effective and larger context did not bring improvements, possibly because of over-fitting.

Going back to the examples from Figure 1, our model can disambiguate the translation of "shooting" based on the source-context features (either the full form or lemma). For the morphological disambiguation of the translation of "yellowish cat", the model has access to the morphological tags of the preceding target words which can disambiguate the correct morphological case.

We used slightly different subsets of the full feature set for different languages. In particular, we left out surface form features and/or bilingual features in some settings because they decreased performance, presumably due to over-fitting.

3 Efficient Implementation

Originally, we assumed that using target-side context features in decoding would be too expensive, considering that we would have to query our model roughly as often as the language model. In preliminary experiments, we therefore focused on n-best list re-ranking. We obtained small gains but all of our results were substantially worse than with the integrated model, so we omit them from the paper.

We find that decoding with a feature-rich target-context model is in fact feasible. In this section, we describe optimizations at different stages of our pipeline which make training and inference with our model practical.

3.1 Feature Extraction

We implemented the code for feature extraction only once; identical code is used at training time and in decoding. At training time, the generated features are written into a file whereas at test time, they are fed directly into the classifier via its library interface.

This design decision not only ensures consistency in feature representation but also makes the process of feature extraction efficient. In training, we are easily able to use multi-threading (already implemented in Moses) and because the processing of training data is a trivially parallel task, we can also use distributed computation and run separate instances of (multi-threaded) Moses on several machines. This enables us to easily produce training files from millions of parallel sentences within a short time.

3.2 Model Training

VW is a very fast classifier by itself, however for very large data, its training can be further sped up by using parallelization. We take advantage of its implementation of the *AllReduce* scheme which we utilize in a grid engine environment. We shuffle and shard the data and then assign each shard to a worker job. With AllReduce, there is a master job which synchronizes the learned weight vector with all workers. We have compared this approach with the standard single-threaded, single-process training and found that we obtain identical model accuracy. We usually use around 10-20 training jobs.

This way, we can process our large training files quickly and train the full model (using multiple passes over the data) within hours; effectively, neither feature extraction nor model training become a significant bottleneck in the full MT system training pipeline.

3.3 Decoding

In phrase-based decoding, translation is generated from left to right. At each step, a partial translation (initially empty) is extended by translating a previously uncovered part of the source sentence. There are typically many ways to translate each source span, which we refer to as translation options. The decoding process gradually extends the generated partial translations until the whole source sentence is covered; the final translation is then the full translation hypothesis with the highest model score. Various pruning strategies are applied to make decoding tractable.

Evaluating a feature-rich classifier during decoding is a computationally expensive operation. Because the features in our model depend on target-side context, the feature function which computes the classifier score cannot evaluate the translation options in isolation (independently of the partial translation). Instead, similarly to a language model, it needs to look at previously generated words. This also entails maintaining a state which captures the required context information.

A naive integration of the classifier would simply generate all source-context features, all target-context features and all features describing the translation option each time a partial hypothesis is evaluated. This is a computationally very expensive approach.

We instead propose several technical solutions

which make decoding reasonably fast. Decoding a single sentence with the naive approach takes 13.7 seconds on average. With our optimization, this average time is reduced to 2.9 seconds, i.e. almost by 80 per cent. The baseline system produces a translation in 0.8 seconds on average.

Separation of source-context and target-context evaluation. Because we have a linear model, the final score is simply the dot product between a weight vector and a (sparse) feature vector. It is therefore trivial to separate it into two components: one that only contains features which depend on the source context and the other with target context features. We can pre-compute the source-context part of the score before decoding (once we have all translation options for the given sentence). We cache these partial scores and when the translation option is evaluated, we add the partial score of the target-context features to arrive at the final classifier score.

Caching of feature hashes. VW uses feature hashing internally and it is possible to obtain the hash of any feature that we use. When we encounter a previously unseen target context (=state) during decoding, we store the hashes of extracted features in a cache. Therefore for each context, we only run the expensive feature extraction once. Similarly, we pre-compute feature hash vectors for all translation options.

Caching of final results. Our classifier locally normalizes the scores so that the probabilities of translations for a given span sum to one. This cannot be done without evaluating all translation options for the span at the same time. Therefore, when we get a translation option to be scored, we fetch all translation options for the given source span and evaluate all of them. We then normalize the scores and add them to a cache of final results. When the other translation options come up, their scores are simply fetched from the cache. This can also further save computation when we get into a previously seen state (from the point of view of our classifier) and we evaluate the same set of translation options in that state; we will simply find the result in cache in such cases.

When we combine all of these optimizations, we arrive at the query algorithm shown in Figure 2.

```
function EVALUATE(t, s)
    span = t.getSourceSpan()
    if not resultCache.has(span, s) then
        scores = ()
        if not stateCache.has(s) then
            stateCache[s] = CtxFeatures(s)
        end if
        for all t' ← span.tOpts() do
            srcScore = srcScoreCache[t']
            c.addFeatures(stateCache[s])
            c.addFeatures(translationCache[t'])
            tgtScore = c.predict()
            scores[t'] = srcScore + tgtScore
        end for
        normalize(scores)
        resultCache[span, s] = scores
    end if
    return resultCache[span, s][t]
end function
```

Figure 2: Algorithm for obtaining classifier predictions during decoding. The variable t stands for the current translation, s is the current state and c is an instance of the classifier.

4 Experimental Evaluation

We run the main set of experiments on English to Czech translation. To verify that our method is applicable to other language pairs, we also present experiments in English to German, Polish, and Romanian.

In all experiments, we use Treex (Popel and Žabokrtský, 2010) to lemmatize and tag the source data and also to obtain dependency parses of all English sentences.

4.1 English-Czech Translation

As parallel training data, we use (subsets of) the CzEng 1.0 corpus (Bojar et al., 2012). For tuning, we use the WMT13 test set (Bojar et al., 2013) and we evaluate the systems on the WMT14 test set (Bojar et al., 2014). We lemmatize and tag the Czech data using Morphodita (Straková et al., 2014).

Our baseline system is a standard phrase-based Moses setup. The phrase table in both cases is factored and outputs also lemmas and morphological tags. We train a 5-gram LM on the target side of parallel data.

We evaluate three settings in our experiments:

- baseline – vanilla phrase-based system,
- +source – our classifier with source-context features only,

- +target – our classifier with both source-context and target-context features.

For each of these settings, we vary the size of the training data for our classifier, the phrase table and the LM. We experiment with three different sizes: small (200 thousand sentence pairs), medium (5 million sentence pairs), and full (the whole CzEng corpus, over 14.8 million sentence pairs).

For each setting, we run system weight optimization (tuning) using minimum error rate training (Och, 2003) five times and report the average BLEU score. We use MultEval (Clark et al., 2011) to compare the systems and to determine whether the differences in results are statistically significant. We always compare the baseline with +source and +source with +target.

Table 2 shows the obtained results. Statistically significant differences (α=0.01) are marked in bold. The source-context model does not help in the small data setting but brings a substantial improvement of 0.7-0.8 BLEU points for the medium and full data settings, which is an encouraging result.

Target-side context information allows our model to push the translation quality further: even for the small data setting, it brings a substantial improvement of 0.5 BLEU points and the gain remains significant as the data size increases. Even in the full data setting, target-side features improve the score by roughly 0.2 BLEU points.

Our results demonstrate that feature-rich models scale to large data size both in terms of technical feasibility and of translation quality improvements. Target side information seems consistently beneficial, adding further 0.2-0.5 BLEU points on top of the source-context model.

data size	small	medium	full
baseline	10.7	15.2	16.7
+source	10.7	**16.0**	**17.3**
+target	**11.2**	**16.4**	**17.5**

Table 2: BLEU scores obtained on the WMT14 test set. We report the performance of the baseline, the source-context model and the full model.

Intrinsic Evaluation. For completeness, we report intrinsic evaluation results. We evaluate the classifier on a held-out set (WMT13 test set) by extracting all phrase pairs from the test input aligned with the test reference (similarly as we would in training) and scoring each phrase pair (along with other possible translations of the source phrase) with our classifier. An instance is classified correctly if the true translation obtains the highest score by our model. A baseline which always chooses the most frequent phrasal translation obtains accuracy of 51.5. For the source-context model, the held-out accuracy was 66.3, while the target context model achieved accuracy of 74.8. Note that this high difference is somewhat misleading because in this setting, the target-context model has access to the true target context (i.e., it is cheating).

4.2 Additional Language Pairs

We experiment with translation from English into German, Polish, and Romanian.

Our English-German system is trained on the data available for the WMT14 translation task: Europarl (Koehn, 2005) and the Common Crawl corpus,[3] roughly 4.3 million sentence pairs altogether. We tune the system on the WMT13 test set and we test on the WMT14 set. We use TreeTagger (Schmid, 1994) to lemmatize and tag the German data.

English-Polish has not been included in WMT shared tasks so far, but was present as a language pair for several IWSLT editions which concentrate on TED talk translation. Full test sets are only available for 2010, 2011, and 2012. The references for 2013 and 2014 were not made public. We use the development set and test set from 2010 as development data for parameter tuning. The remaining two test sets (2011, 2012) are our test data. We train on the concatenation of Europarl and WIT^3 (Cettolo et al., 2012), ca. 750 thousand sentence pairs. The Polish half has been tagged using WCRFT (Radziszewski, 2013) which produces full morphological tags compatible with the NKJP tagset (Przepiórkowski, 2009).

English-Romanian was added in WMT16. We train our system using the available parallel data – Europarl and SETIMES2 (Tiedemann, 2009), roughly 600 thousand sentence pairs. We tune the English-Romanian system on the official development set and we test on the WMT16 test set. We use the online tagger by Tufis et al. (2008) to preprocess the data.

Table 3 shows the obtained results. Similarly to English-Czech experiments, BLEU scores are av-

[3] http://commoncrawl.org/

input:	the most intensive mining took place there from 1953 to 1962 .
baseline:	nejvíce intenzivní těžba **došlo** tam z roku 1953 **, aby** 1962 .
	the_most intensive mining$_{nom}$ ***there_occurred*** *there from 1953* ***, in_order_to*** *1962 .*
+source:	nejvíce intenzivní **těžby místo** tam z roku 1953 do roku 1962 .
	the_most intensive ***mining***$_{gen}$ ***place*** *there from year 1953 until year 1962 .*
+target:	nejvíce intenzivní těžba probíhala od roku 1953 do roku 1962 .
	the_most intensive mining$_{nom}$ *occurred from year 1953 until year 1962 .*

Figure 3: An example sentence from the test set. Each translation has a corresponding gloss in italics. Errors are marked in bold.

language	de	pl (2011)	pl (2012)	ro
baseline	15.7	12.8	10.4	19.6
+target	**16.2**	**13.4**	**11.1**	**20.2**

Table 3: BLEU scores of the baseline and of the full model for English to German, Polish, and Romanian.

eraged over 5 independent optimization runs. Our system outperforms the baseline by 0.5-0.7 BLEU points in all cases, showing that the method is applicable to other languages with rich morphology.

5 Analysis

We manually analyze the outputs of English-Czech systems. Figure 3 shows an example sentence from the WMT14 test set translated by all the system variants. The baseline system makes an error in verb valency; the Czech verb "došlo" could be used but this verb already has an (implicit) subject and the translation of "mining" ("těžba") would have to be in a different case and at a different position in the sentence. The second error is more interesting, however: the baseline system fails to correctly identify the word sense of the particle "to" and translates it in the sense of purpose, as in "in order to". The source-context model takes the context (span of years) into consideration and correctly disambiguates the translation of "to", choosing the temporal meaning. It still fails to translate the main verb correctly, though. Only the full model with target-context information is able to also correctly translate the verb and inflect its arguments according to their roles in the valency frame. The translation produced by this final system in this case is almost flawless.

In order to verify that the automatically measured results correspond to visible improvements in translation quality, we carried out two annotation experiments. We took a random sample of 104 sentences from the test set and blindly ranked two competing translations (the selection of sentences was identical for both experiments). In the first experiment, we compared the baseline system with +source. In the other experiment, we compared the baseline with +target. The instructions for annotation were simply to compare overall translation quality; we did not ask the annotator to look for any specific phenomena. In terms of automatic measures, our selection has similar characteristics as the full test set: BLEU scores obtained on our sample are 15.08, 16.22 and 16.53 for the baseline, +source and +target respectively.

In the first case, the annotator marked 52 translations as equal in quality, 26 translations produced by +source were marked as better and in the remaining 26 cases, the baseline won the ranking. Even though there is a difference in BLEU, human annotation does not confirm this measurement, ranking both systems equally.

In the second experiment, 52 translations were again marked as equal. In 34 cases, +target produced a better translation while in 18 cases, the baseline output won. The difference between the baseline and +target suggests that the target-context model may provide information which is useful for translation quality as perceived by humans.

Our overall impression from looking at the system outputs was that both the source-context and target-context model tend to fix many morpho-syntactic errors. Interestingly, we do not observe as many improvements in the word/phrase sense disambiguation, though the source context does help semantics in some sentences. The target-context model tends to preserve the overall agreement and coherence better than the system with a source-context model only. We list several such examples in Figure 4. Each of them is fully cor-

input:	destruction of the equipment means that Syria can no longer produce new chemical weapons .
+source:	**zničením** zařízení znamená , že Sýrie již nemůže vytvářet nové chemické zbraně .
	destruction_of$_{instr}$ *equipment means , that Syria already cannot produce new chemical weapons .*
+target:	zničení zařízení znamená , že Sýrie již nemůže vytvářet nové chemické zbraně .
	destruction_of$_{nom}$ *equipment means , that Syria already cannot produce new chemical weapons .*
input:	nothing like that existed , and despite that we knew far more about each other .
+source:	nic takového neexistovalo , a přesto jsme věděli daleko víc o **jeden na druhého** .
	nothing like_that existed , and despite_that we knew far more about ***one***$_{nom}$ ***on other*** *.*
+target:	nic takového neexistovalo , a přesto jsme věděli daleko víc o sobě navzájem .
	nothing like_that existed , and despite_that we knew far more about each other .
input:	the authors have been inspired by their neighbours .
+source:	autoři byli inspirováni **svých sousedů** .
	the authors have been inspired ***their***$_{gen}$ ***neighbours***$_{gen}$ *.*
+target:	autoři byli inspirováni svými sousedy .
	the authors have been inspired their$_{instr}$ *neighbours*$_{instr}$ *.*

Figure 4: Example sentences from the test set showing improvements in morphological coherence. Each translation has a corresponding gloss in italics. Errors are marked in bold.

rected by the target-context model, producing an accurate translation of the input.

6 Related Work

Discriminative models in MT have been proposed before. Carpuat and Wu (2007) trained a maximum entropy classifier for each source phrase type which used source context information to disambiguate its translations. The models did not capture target-side information and they were independent; no parameters were shared between classifiers for different phrases. They used a strong feature set originally developed for word sense disambiguation. Gimpel and Smith (2008) also used wider source-context information but did not train a classifier; instead, the features were included directly in the log-linear model of the decoder. Mauser et al. (2009) introduced the "discriminative word lexicon" and trained a binary classifier for each target word, using as features only the bag of words (from the whole source sentence). Training sentences where the target word occurred were used as positive examples, other sentences served as negative examples. Jeong et al. (2010) proposed a discriminative lexicon with a rich feature set tailored to translation into morphologically rich languages; unlike our work, their model only used source-context features.

Subotin (2011) included target-side context information in a maximum-entropy model for the prediction of morphology. The work was done within the paradigm of hierarchical PBMT and assumes that cube pruning is used in decoding. Their algorithm was tailored to the specific problem of passing non-local information about morphological agreement required by individual rules (such as explicit rules enforcing subject-verb agreement). Our algorithm only assumes that hypotheses are constructed left to right and provides a general way for including target context information in the classifier, regardless of the type of features. Our implementation is freely available and can be further extended by other researchers in the future.

7 Conclusions

We presented a discriminative model for MT which uses both source and target context information. We have shown that such a model can be used directly during decoding in a relatively efficient way. We have shown that this model consistently significantly improves the quality of English-Czech translation over a strong baseline with large training data. We have validated the effectiveness of our model on several additional language pairs. We have provided an analysis showing concrete examples of improved lexical selection and morphological coherence. Our work is available in the main branch of Moses for use by other researchers.

Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreements no. 644402 (HimL) and 645452 (QT21), from the European Research Council (ERC) under grant agreement no. 640550, and from the SVV project number 260 333. This work has been using language resources stored and distributed by the LINDAT/CLARIN project of the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071).

References

Ondřej Bojar, Zdeněk Žabokrtský, Ondřej Dušek, Petra Galuščáková, Martin Majliš, David Mareček, Jiří Maršík, Michal Novák, Martin Popel, and Aleš Tamchyna. 2012. The Joy of Parallelism with CzEng 1.0. In *Proc. of LREC*, pages 3921–3928. ELRA.

Ondřej Bojar, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on Statistical Machine Translation. In *Proceedings of the Eighth Workshop on Statistical Machine Translation*, pages 1–44, Sofia, Bulgaria, August. Association for Computational Linguistics.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, MD, USA. Association for Computational Linguistics.

Marine Carpuat and Dekai Wu. 2007. Improving statistical machine translation using word sense disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Prague, Czech Republic.

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy, May.

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 176–181, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M. Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1370–1380.

K. Gimpel and N. A. Smith. 2008. Rich Source-Side Context for Statistical Machine Translation. Columbus, Ohio.

Minwoo Jeong, Kristina Toutanova, Hisami Suzuki, and Chris Quirk. 2010. A discriminative lexicon model for complex morphology. In *The Ninth Conference of the Association for Machine Translation in the Americas*. Association for Computational Linguistics, November.

Philipp Koehn and Hieu Hoang. 2007. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 868–876, Prague, Czech Republic, June. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Conference Proceedings: the tenth Machine Translation Summit*, pages 79–86, Phuket, Thailand. AAMT, AAMT.

Arne Mauser, Sasa Hasan, and Hermann Ney. 2009. Extending Statistical Machine Translation with Discriminative and Trigger-Based Lexicon Models. pages 210–218, Suntec, Singapore.

Jan Niehues, Teresa Herrmann, Stephan Vogel, and Alex Waibel. 2011. Wider Context by Using Bilingual Language Models in Machine Translation. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 198–206, Edinburgh, Scotland, July. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proc. of ACL*, pages 160–167, Sapporo, Japan. ACL.

Martin Popel and Zdeněk Žabokrtský. 2010. TectoMT: Modular NLP Framework. In Hrafn Loftsson, Eirikur Rögnvaldsson, and Sigrun Helgadottir, editors, *IceTAL 2010*, volume 6233 of *Lecture Notes in Computer Science*, pages 293–304. Iceland Centre for Language Technology (ICLT), Springer.

Adam Przepiórkowski. 2009. A comparison of two morphosyntactic tagsets of Polish. In Violetta Koseska-Toszewa, Ludmila Dimitrova, and Roman Roszko, editors, *Representing Semantics in Digital Lexicography: Proceedings of MONDILEX Fourth Open Workshop*, pages 138–144, Warsaw.

Adam Radziszewski. 2013. A tiered CRF tagger for Polish. In Robert Bembenik, Lukasz Skonieczny, Henryk Rybinski, Marzena Kryszkiewicz, and Marek Niezgodka, editors, *Intelligent Tools for Building a Scientific Information Platform*, volume 467 of *Studies in Computational Intelligence*, pages 215–230. Springer.

Helmut Schmid. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK.

Jana Straková, Milan Straka, and Jan Hajič. 2014. Open-Source Tools for Morphology, Lemmatization, POS Tagging and Named Entity Recognition.

In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 13–18, Baltimore, Maryland, June. Association for Computational Linguistics.

Michael Subotin. 2011. An exponential translation model for target language morphology. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 230–238.

Aleš Tamchyna, Fabienne Braune, Alexander Fraser, Marine Carpuat, Hal Daumé III, and Chris Quirk. 2014. Integrating a discriminative classifier into phrase-based and hierarchical decoding. *The Prague Bulletin of Mathematical Linguistics*, 101:29–41.

Jörg Tiedemann. 2009. News from OPUS - A collection of multilingual parallel corpora with tools and interfaces. In N. Nicolov, K. Bontcheva, G. Angelova, and R. Mitkov, editors, *Recent Advances in Natural Language Processing*, volume V, pages 237–248. John Benjamins, Amsterdam/Philadelphia, Borovets, Bulgaria.

Dan Tufis, Radu Ion, Alexandru Ceausu, and Dan Stefanescu. 2008. Racai's linguistic web services. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*.

D. Vickrey, L. Biewald, M. Teyssier, and D. Koller. 2005. Word-Sense Disambiguation for Machine Translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Vancouver, Canada, October.

Neural Machine Translation of Rare Words with Subword Units

Rico Sennrich and **Barry Haddow** and **Alexandra Birch**
School of Informatics, University of Edinburgh
{rico.sennrich,a.birch}@ed.ac.uk, bhaddow@inf.ed.ac.uk

Abstract

Neural machine translation (NMT) models typically operate with a fixed vocabulary, but translation is an open-vocabulary problem. Previous work addresses the translation of out-of-vocabulary words by backing off to a dictionary. In this paper, we introduce a simpler and more effective approach, making the NMT model capable of open-vocabulary translation by encoding rare and unknown words as sequences of subword units. This is based on the intuition that various word classes are translatable via smaller units than words, for instance names (via character copying or transliteration), compounds (via compositional translation), and cognates and loanwords (via phonological and morphological transformations). We discuss the suitability of different word segmentation techniques, including simple character n-gram models and a segmentation based on the *byte pair encoding* compression algorithm, and empirically show that subword models improve over a back-off dictionary baseline for the WMT 15 translation tasks English→German and English→Russian by up to 1.1 and 1.3 BLEU, respectively.

1 Introduction

Neural machine translation has recently shown impressive results (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). However, the translation of rare words is an open problem. The vocabulary of neural models is typically limited to 30 000–50 000 words, but translation is an open-vocabulary problem, and especially for languages with productive word formation processes such as agglutination and compounding, translation models require mechanisms that go below the word level. As an example, consider compounds such as the German *Abwasser|behandlungs|anlange* 'sewage water treatment plant', for which a segmented, variable-length representation is intuitively more appealing than encoding the word as a fixed-length vector.

For word-level NMT models, the translation of out-of-vocabulary words has been addressed through a back-off to a dictionary look-up (Jean et al., 2015; Luong et al., 2015b). We note that such techniques make assumptions that often do not hold true in practice. For instance, there is not always a 1-to-1 correspondence between source and target words because of variance in the degree of morphological synthesis between languages, like in our introductory compounding example. Also, word-level models are unable to translate or generate unseen words. Copying unknown words into the target text, as done by (Jean et al., 2015; Luong et al., 2015b), is a reasonable strategy for names, but morphological changes and transliteration is often required, especially if alphabets differ.

We investigate NMT models that operate on the level of subword units. Our main goal is to model open-vocabulary translation in the NMT network itself, without requiring a back-off model for rare words. In addition to making the translation process simpler, we also find that the subword models achieve better accuracy for the translation of rare words than large-vocabulary models and back-off dictionaries, and are able to productively generate new words that were not seen at training time. Our analysis shows that the neural networks are able to learn compounding and transliteration from subword representations.

This paper has two main contributions:

- We show that open-vocabulary neural ma-

The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. - Samsung R&D Institute Poland.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1715–1725,
Berlin, Germany, August 7-12, 2016.

chine translation is possible by encoding (rare) words via subword units. We find our architecture simpler and more effective than using large vocabularies and back-off dictionaries (Jean et al., 2015; Luong et al., 2015b).

- We adapt *byte pair encoding* (BPE) (Gage, 1994), a compression algorithm, to the task of word segmentation. BPE allows for the representation of an open vocabulary through a fixed-size vocabulary of variable-length character sequences, making it a very suitable word segmentation strategy for neural network models.

2 Neural Machine Translation

We follow the neural machine translation architecture by Bahdanau et al. (2015), which we will briefly summarize here. However, we note that our approach is not specific to this architecture.

The neural machine translation system is implemented as an encoder-decoder network with recurrent neural networks.

The encoder is a bidirectional neural network with gated recurrent units (Cho et al., 2014) that reads an input sequence $x = (x_1, ..., x_m)$ and calculates a forward sequence of hidden states $(\overrightarrow{h}_1, ..., \overrightarrow{h}_m)$, and a backward sequence $(\overleftarrow{h}_1, ..., \overleftarrow{h}_m)$. The hidden states $\overrightarrow{h}_j$ and $\overleftarrow{h}_j$ are concatenated to obtain the annotation vector h_j.

The decoder is a recurrent neural network that predicts a target sequence $y = (y_1, ..., y_n)$. Each word y_i is predicted based on a recurrent hidden state s_i, the previously predicted word y_{i-1}, and a context vector c_i. c_i is computed as a weighted sum of the annotations h_j. The weight of each annotation h_j is computed through an *alignment model* α_{ij}, which models the probability that y_i is aligned to x_j. The alignment model is a single-layer feedforward neural network that is learned jointly with the rest of the network through backpropagation.

A detailed description can be found in (Bahdanau et al., 2015). Training is performed on a parallel corpus with stochastic gradient descent. For translation, a beam search with small beam size is employed.

3 Subword Translation

The main motivation behind this paper is that the translation of some words is transparent in that they are translatable by a competent translator even if they are novel to him or her, based on a translation of known subword units such as morphemes or phonemes. Word categories whose translation is potentially transparent include:

- named entities. Between languages that share an alphabet, names can often be copied from source to target text. Transcription or transliteration may be required, especially if the alphabets or syllabaries differ. Example:
  Barack Obama (English; German)
  Барак Обама (Russian)
  バラク・オバマ (ba-ra-ku o-ba-ma) (Japanese)
- cognates and loanwords. Cognates and loanwords with a common origin can differ in regular ways between languages, so that character-level translation rules are sufficient (Tiedemann, 2012). Example:
  claustrophobia (English)
  Klaustrophobie (German)
  Клаустрофобия (Klaustrofobiâ) (Russian)
- morphologically complex words. Words containing multiple morphemes, for instance formed via compounding, affixation, or inflection, may be translatable by translating the morphemes separately. Example:
  solar system (English)
  Sonnensystem (Sonne + System) (German)
  Naprendszer (Nap + Rendszer) (Hungarian)

In an analysis of 100 rare tokens (not among the 50 000 most frequent types) in our German training data[1], the majority of tokens are potentially translatable from English through smaller units. We find 56 compounds, 21 names, 6 loanwords with a common origin (*emancipate→emanzipieren*), 5 cases of transparent affixation (*sweetish* 'sweet' + '-ish' → *süßlich* 'süß' + '-lich'), 1 number and 1 computer language identifier.

Our hypothesis is that a segmentation of rare words into appropriate subword units is sufficient to allow for the neural translation network to learn transparent translations, and to generalize this knowledge to translate and produce unseen words.[2] We provide empirical support for this hy-

[1] Primarily parliamentary proceedings and web crawl data.

[2] Not every segmentation we produce is transparent. While we expect no performance benefit from opaque segmentations, i.e. segmentations where the units cannot be translated independently, our NMT models show robustness towards oversplitting.

pothesis in Sections 4 and 5. First, we discuss different subword representations.

3.1 Related Work

For Statistical Machine Translation (SMT), the translation of unknown words has been the subject of intensive research.

A large proportion of unknown words are names, which can just be copied into the target text if both languages share an alphabet. If alphabets differ, transliteration is required (Durrani et al., 2014). Character-based translation has also been investigated with phrase-based models, which proved especially successful for closely related languages (Vilar et al., 2007; Tiedemann, 2009; Neubig et al., 2012).

The segmentation of morphologically complex words such as compounds is widely used for SMT, and various algorithms for morpheme segmentation have been investigated (Nießen and Ney, 2000; Koehn and Knight, 2003; Virpioja et al., 2007; Stallard et al., 2012). Segmentation algorithms commonly used for phrase-based SMT tend to be conservative in their splitting decisions, whereas we aim for an aggressive segmentation that allows for open-vocabulary translation with a compact network vocabulary, and without having to resort to back-off dictionaries.

The best choice of subword units may be task-specific. For speech recognition, phone-level language models have been used (Bazzi and Glass, 2000). Mikolov et al. (2012) investigate subword language models, and propose to use syllables. For multilingual segmentation tasks, multilingual algorithms have been proposed (Snyder and Barzilay, 2008). We find these intriguing, but inapplicable at test time.

Various techniques have been proposed to produce fixed-length continuous word vectors based on characters or morphemes (Luong et al., 2013; Botha and Blunsom, 2014; Ling et al., 2015a; Kim et al., 2015). An effort to apply such techniques to NMT, parallel to ours, has found no significant improvement over word-based approaches (Ling et al., 2015b). One technical difference from our work is that the attention mechanism still operates on the level of words in the model by Ling et al. (2015b), and that the representation of each word is fixed-length. We expect that the attention mechanism benefits from our variable-length representation: the network can learn to place attention on different subword units at each step. Recall our introductory example *Abwasserbehandlungsanlange*, for which a subword segmentation avoids the information bottleneck of a fixed-length representation.

Neural machine translation differs from phrase-based methods in that there are strong incentives to minimize the vocabulary size of neural models to increase time and space efficiency, and to allow for translation without back-off models. At the same time, we also want a compact representation of the text itself, since an increase in text length reduces efficiency and increases the distances over which neural models need to pass information.

A simple method to manipulate the trade-off between vocabulary size and text size is to use shortlists of unsegmented words, using subword units only for rare words. As an alternative, we propose a segmentation algorithm based on byte pair encoding (BPE), which lets us learn a vocabulary that provides a good compression rate of the text.

3.2 Byte Pair Encoding (BPE)

Byte Pair Encoding (BPE) (Gage, 1994) is a simple data compression technique that iteratively replaces the most frequent pair of bytes in a sequence with a single, unused byte. We adapt this algorithm for word segmentation. Instead of merging frequent pairs of bytes, we merge characters or character sequences.

Firstly, we initialize the symbol vocabulary with the character vocabulary, and represent each word as a sequence of characters, plus a special end-of-word symbol '·', which allows us to restore the original tokenization after translation. We iteratively count all symbol pairs and replace each occurrence of the most frequent pair ('A', 'B') with a new symbol 'AB'. Each merge operation produces a new symbol which represents a character n-gram. Frequent character n-grams (or whole words) are eventually merged into a single symbol, thus BPE requires no shortlist. The final symbol vocabulary size is equal to the size of the initial vocabulary, plus the number of merge operations – the latter is the only hyperparameter of the algorithm.

For efficiency, we do not consider pairs that cross word boundaries. The algorithm can thus be run on the dictionary extracted from a text, with each word being weighted by its frequency. A minimal Python implementation is shown in Al-

Algorithm 1 Learn BPE operations

```
import re, collections

def get_stats(vocab):
  pairs = collections.defaultdict(int)
  for word, freq in vocab.items():
    symbols = word.split()
    for i in range(len(symbols)-1):
      pairs[symbols[i],symbols[i+1]] += freq
  return pairs

def merge_vocab(pair, v_in):
  v_out = {}
  bigram = re.escape(' '.join(pair))
  p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
  for word in v_in:
    w_out = p.sub(''.join(pair), word)
    v_out[w_out] = v_in[word]
  return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
         'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10
for i in range(num_merges):
  pairs = get_stats(vocab)
  best = max(pairs, key=pairs.get)
  vocab = merge_vocab(best, vocab)
  print(best)
```

r ·	→	r·
l o	→	lo
lo w	→	low
e r·	→	er·

Figure 1: BPE merge operations learned from dictionary {'low', 'lowest', 'newer', 'wider'}.

gorithm 1. In practice, we increase efficiency by indexing all pairs, and updating data structures incrementally.

The main difference to other compression algorithms, such as Huffman encoding, which have been proposed to produce a variable-length encoding of words for NMT (Chitnis and DeNero, 2015), is that our symbol sequences are still interpretable as subword units, and that the network can generalize to translate and produce new words (unseen at training time) on the basis of these subword units.

Figure 1 shows a toy example of learned BPE operations. At test time, we first split words into sequences of characters, then apply the learned operations to merge the characters into larger, known symbols. This is applicable to any word, and allows for open-vocabulary networks with fixed symbol vocabularies.[3] In our example, the OOV 'lower' would be segmented into 'low er·'.

[3]The only symbols that will be unknown at test time are unknown characters, or symbols of which all occurrences in the training text have been merged into larger symbols, like 'safeguar', which has all occurrences in our training text merged into 'safeguard'. We observed no such symbols at test time, but the issue could be easily solved by recursively reversing specific merges until all symbols are known.

We evaluate two methods of applying BPE: learning two independent encodings, one for the source, one for the target vocabulary, or learning the encoding on the union of the two vocabularies (which we call *joint BPE*).[4] The former has the advantage of being more compact in terms of text and vocabulary size, and having stronger guarantees that each subword unit has been seen in the training text of the respective language, whereas the latter improves consistency between the source and the target segmentation. If we apply BPE independently, the same name may be segmented differently in the two languages, which makes it harder for the neural models to learn a mapping between the subword units. To increase the consistency between English and Russian segmentation despite the differing alphabets, we transliterate the Russian vocabulary into Latin characters with ISO-9 to learn the joint BPE encoding, then transliterate the BPE merge operations back into Cyrillic to apply them to the Russian training text.[5]

4 Evaluation

We aim to answer the following empirical questions:

- Can we improve the translation of rare and unseen words in neural machine translation by representing them via subword units?
- Which segmentation into subword units performs best in terms of vocabulary size, text size, and translation quality?

We perform experiments on data from the shared translation task of WMT 2015. For English→German, our training set consists of 4.2 million sentence pairs, or approximately 100 million tokens. For English→Russian, the training set consists of 2.6 million sentence pairs, or approximately 50 million tokens. We tokenize and truecase the data with the scripts provided in Moses (Koehn et al., 2007). We use newstest2013 as development set, and report results on newstest2014 and newstest2015.

We report results with BLEU (*mteval-v13a.pl*), and CHRF3 (Popović, 2015), a character n-gram F_3 score which was found to correlate well with

[4]In practice, we simply concatenate the source and target side of the training set to learn joint BPE.

[5]Since the Russian training text also contains words that use the Latin alphabet, we also apply the Latin BPE operations.

human judgments, especially for translations out of English (Stanojević et al., 2015). Since our main claim is concerned with the translation of rare and unseen words, we report separate statistics for these. We measure these through unigram F_1, which we calculate as the harmonic mean of clipped unigram precision and recall.[6]

We perform all experiments with Groundhog[7] (Bahdanau et al., 2015). We generally follow settings by previous work (Bahdanau et al., 2015; Jean et al., 2015). All networks have a hidden layer size of 1000, and an embedding layer size of 620. Following Jean et al. (2015), we only keep a shortlist of $\tau = 30000$ words in memory.

During training, we use Adadelta (Zeiler, 2012), a minibatch size of 80, and reshuffle the training set between epochs. We train a network for approximately 7 days, then take the last 4 saved models (models being saved every 12 hours), and continue training each with a fixed embedding layer (as suggested by (Jean et al., 2015)) for 12 hours. We perform two independent training runs for each models, once with cut-off for gradient clipping (Pascanu et al., 2013) of 5.0, once with a cut-off of 1.0 – the latter produced better single models for most settings. We report results of the system that performed best on our development set (newstest2013), and of an ensemble of all 8 models.

We use a beam size of 12 for beam search, with probabilities normalized by sentence length. We use a bilingual dictionary based on fast-align (Dyer et al., 2013). For our baseline, this serves as back-off dictionary for rare words. We also use the dictionary to speed up translation for all experiments, only performing the softmax over a filtered list of candidate translations (like Jean et al. (2015), we use $K = 30000$; $K' = 10$).

4.1 Subword statistics

Apart from translation quality, which we will verify empirically, our main objective is to represent an open vocabulary through a compact fixed-size subword vocabulary, and allow for efficient training and decoding.[8]

Statistics for different segmentations of the German side of the parallel data are shown in Table 1. A simple baseline is the segmentation of words into character n-grams.[9] Character n-grams allow for different trade-offs between sequence length (# tokens) and vocabulary size (# types), depending on the choice of n. The increase in sequence length is substantial; one way to reduce sequence length is to leave a shortlist of the k most frequent word types unsegmented. Only the unigram representation is truly open-vocabulary. However, the unigram representation performed poorly in preliminary experiments, and we report translation results with a bigram representation, which is empirically better, but unable to produce some tokens in the test set with the training set vocabulary.

We report statistics for several word segmentation techniques that have proven useful in previous SMT research, including frequency-based compound splitting (Koehn and Knight, 2003), rule-based hyphenation (Liang, 1983), and Morfessor (Creutz and Lagus, 2002). We find that they only moderately reduce vocabulary size, and do not solve the unknown word problem, and we thus find them unsuitable for our goal of open-vocabulary translation without back-off dictionary.

BPE meets our goal of being open-vocabulary, and the learned merge operations can be applied to the test set to obtain a segmentation with no unknown symbols.[10] Its main difference from the character-level model is that the more compact representation of BPE allows for shorter sequences, and that the attention model operates on variable-length units.[11] Table 1 shows BPE with 59 500 merge operations, and joint BPE with 89 500 operations.

In practice, we did not include infrequent subword units in the NMT network vocabulary, since there is noise in the subword symbol sets, e.g. because of characters from foreign alphabets. Hence, our network vocabularies in Table 2 are typically slightly smaller than the number of types in Table 1.

[6] Clipped unigram precision is essentially 1-gram BLEU without brevity penalty.

[7] `github.com/sebastien-j/LV_groundhog`

[8] The time complexity of encoder-decoder architectures is at least linear to sequence length, and oversplitting harms efficiency.

[9] Our character n-grams do not cross word boundaries. We mark whether a subword is word-final or not with a special character, which allows us to restore the original tokenization.

[10] Joint BPE can produce segments that are unknown because they only occur in the English training text, but these are rare (0.05% of test tokens).

[11] We highlighted the limitations of word-level attention in section 3.1. At the other end of the spectrum, the character level is suboptimal for alignment (Tiedemann, 2009).

name	segmentation	shortlist	vocabulary source	vocabulary target	BLEU single	BLEU ens-8	CHRF3 single	CHRF3 ens-8	unigram F_1 (%) all	unigram F_1 (%) rare	unigram F_1 (%) OOV
syntax-based (Sennrich and Haddow, 2015)					24.4	-	55.3	-	59.1	46.0	37.7
WUnk	-	-	300 000	500 000	20.6	22.8	47.2	48.9	56.7	20.4	0.0
WDict	-	-	300 000	500 000	22.0	24.2	50.5	52.4	58.1	36.8	**36.8**
C2-50k	char-bigram	50 000	60 000	60 000	**22.8**	**25.3**	51.9	53.5	58.4	40.5	30.9
BPE-60k	BPE	-	60 000	60 000	21.5	24.5	**52.0**	53.9	58.4	40.9	29.3
BPE-J90k	BPE (joint)	-	90 000	90 000	**22.8**	24.7	51.7	**54.1**	**58.5**	**41.8**	33.6

Table 2: English→German translation performance (BLEU, CHRF3 and unigram F_1) on newstest2015. Ens-8: ensemble of 8 models. Best NMT system in bold. Unigram F_1 (with ensembles) is computed for all words ($n = 44085$), rare words (not among top 50 000 in training set; $n = 2900$), and OOVs (not in training set; $n = 1168$).

segmentation	# tokens	# types	# UNK
none	100 m	1 750 000	1079
characters	550 m	3000	0
character bigrams	306 m	20 000	34
character trigrams	214 m	120 000	59
compound splitting$^{\triangle}$	102 m	1 100 000	643
morfessor*	109 m	544 000	237
hyphenation$^{\diamond}$	186 m	404 000	230
BPE	112 m	63 000	0
BPE (joint)	111 m	82 000	32
character bigrams (shortlist: 50 000)	129 m	69 000	34

Table 1: Corpus statistics for German training corpus with different word segmentation techniques. #UNK: number of unknown tokens in newstest2013. $\triangle$: (Koehn and Knight, 2003); *: (Creutz and Lagus, 2002); $\diamond$: (Liang, 1983).

4.2 Translation experiments

English→German translation results are shown in Table 2; English→Russian results in Table 3.

Our baseline **WDict** is a word-level model with a back-off dictionary. It differs from **WUnk** in that the latter uses no back-off dictionary, and just represents out-of-vocabulary words as UNK[12]. The back-off dictionary improves unigram F_1 for rare and unseen words, although the improvement is smaller for English→Russian, since the back-off dictionary is incapable of transliterating names.

All subword systems operate without a back-off dictionary. We first focus on unigram F_1, where all systems improve over the baseline, especially for rare words (36.8%→41.8% for EN→DE; 26.5%→29.7% for EN→RU). For OOVs, the baseline strategy of copying unknown words works well for English→German. However, when alphabets differ, like in English→Russian, the subword models do much better.

[12]We use *UNK* for words that are outside the model vocabulary, and *OOV* for those that do not occur in the training text.

Unigram F_1 scores indicate that learning the BPE symbols on the vocabulary union (**BPE-J90k**) is more effective than learning them separately (**BPE-60k**), and more effective than using character bigrams with a shortlist of 50 000 unsegmented words (**C2-50k**), but all reported subword segmentations are viable choices and outperform the back-off dictionary baseline.

Our subword representations cause big improvements in the translation of rare and unseen words, but these only constitute 9-11% of the test sets. Since rare words tend to carry central information in a sentence, we suspect that BLEU and CHRF3 underestimate their effect on translation quality. Still, we also see improvements over the baseline in total unigram F_1, as well as BLEU and CHRF3, and the subword ensembles outperform the WDict baseline by 0.3–1.3 BLEU and 0.6–2 CHRF3. There is some inconsistency between BLEU and CHRF3, which we attribute to the fact that BLEU has a precision bias, and CHRF3 a recall bias.

For English→German, we observe the best BLEU score of 25.3 with C2-50k, but the best CHRF3 score of 54.1 with BPE-J90k. For comparison to the (to our knowledge) best non-neural MT system on this data set, we report syntax-based SMT results (Sennrich and Haddow, 2015). We observe that our best systems outperform the syntax-based system in terms of BLEU, but not in terms of CHRF3. Regarding other neural systems, Luong et al. (2015a) report a BLEU score of 25.9 on newstest2015, but we note that they use an ensemble of 8 independently trained models, and also report strong improvements from applying dropout, which we did not use. We are confident that our improvements to the translation of rare words are orthogonal to improvements achievable through other improvements in the network archi-

tecture, training algorithm, or better ensembles.

For English→Russian, the state of the art is the phrase-based system by Haddow et al. (2015). It outperforms our WDict baseline by 1.5 BLEU. The subword models are a step towards closing this gap, and BPE-J90k yields an improvement of 1.3 BLEU, and 2.0 CHRF3, over WDict.

As a further comment on our translation results, we want to emphasize that performance variability is still an open problem with NMT. On our development set, we observe differences of up to 1 BLEU between different models. For single systems, we report the results of the model that performs best on dev (out of 8), which has a stabilizing effect, but how to control for randomness deserves further attention in future research.

5 Analysis

5.1 Unigram accuracy

Our main claims are that the translation of rare and unknown words is poor in word-level NMT models, and that subword models improve the translation of these word types. To further illustrate the effect of different subword segmentations on the translation of rare and unseen words, we plot target-side words sorted by their frequency in the training set.[13] To analyze the effect of vocabulary size, we also include the system **C2-3/500k**, which is a system with the same vocabulary size as the WDict baseline, and character bigrams to represent unseen words.

Figure 2 shows results for the English–German ensemble systems on newstest2015. Unigram F_1 of all systems tends to decrease for lower-frequency words. The baseline system has a spike in F_1 for OOVs, i.e. words that do not occur in the training text. This is because a high proportion of OOVs are names, for which a copy from the source to the target text is a good strategy for English→German.

The systems with a target vocabulary of 500 000 words mostly differ in how well they translate words with rank > 500 000. A back-off dictionary is an obvious improvement over producing UNK, but the subword system C2-3/500k achieves better performance. Note that all OOVs that the back-off dictionary produces are words that are copied from the source, usually names, while the subword systems can productively form new words such as compounds.

For the 50 000 most frequent words, the representation is the same for all neural networks, and all neural networks achieve comparable unigram F_1 for this category. For the interval between frequency rank 50 000 and 500 000, the comparison between C2-3/500k and C2-50k unveils an interesting difference. The two systems only differ in the size of the shortlist, with C2-3/500k representing words in this interval as single units, and C2-50k via subword units. We find that the performance of C2-3/500k degrades heavily up to frequency rank 500 000, at which point the model switches to a subword representation and performance recovers. The performance of C2-50k remains more stable. We attribute this to the fact that subword units are less sparse than words. In our training set, the frequency rank 50 000 corresponds to a frequency of 60 in the training data; the frequency rank 500 000 to a frequency of 2. Because subword representations are less sparse, reducing the size of the network vocabulary, and representing more words via subword units, can lead to better performance.

The F_1 numbers hide some qualitative differences between systems. For English→German, WDict produces few OOVs (26.5% recall), but with high precision (60.6%) , whereas the subword systems achieve higher recall, but lower precision. We note that the character bigram model C2-50k produces the most OOV words, and achieves relatively low precision of 29.1% for this category. However, it outperforms the back-off dictionary in recall (33.0%). BPE-60k, which suffers from transliteration (or copy) errors due to segmentation inconsistencies, obtains a slightly better precision (32.4%), but a worse recall (26.6%). In contrast to BPE-60k, the joint BPE encoding of BPE-J90k improves both precision (38.6%) and recall (29.8%).

For English→Russian, unknown names can only rarely be copied, and usually require transliteration. Consequently, the WDict baseline performs more poorly for OOVs (9.2% precision; 5.2% recall), and the subword models improve both precision and recall (21.9% precision and 15.6% recall for BPE-J90k). The full unigram F_1 plot is shown in Figure 3.

[13] We perform binning of words with the same training set frequency, and apply bezier smoothing to the graph.

name	segmentation	shortlist	vocabulary source	vocabulary target	BLEU single	BLEU ens-8	CHRF3 single	CHRF3 ens-8	unigram F_1 (%) all	unigram F_1 (%) rare	unigram F_1 (%) OOV
phrase-based (Haddow et al., 2015)					24.3	-	53.8	-	56.0	31.3	16.5
WUnk	-	-	300 000	500 000	18.8	22.4	46.5	49.9	54.2	25.2	0.0
WDict	-	-	300 000	500 000	19.1	22.8	47.5	51.0	54.8	26.5	6.6
C2-50k	char-bigram	50 000	60 000	60 000	**20.9**	**24.1**	49.0	51.6	55.2	27.8	17.4
BPE-60k	BPE	-	60 000	60 000	20.5	23.6	**49.8**	52.7	55.3	29.7	15.6
BPE-J90k	BPE (joint)	-	90 000	100 000	20.4	**24.1**	49.7	**53.0**	**55.8**	**29.7**	**18.3**

Table 3: English→Russian translation performance (BLEU, CHRF3 and unigram F_1) on newstest2015. Ens-8: ensemble of 8 models. Best NMT system in bold. Unigram F_1 (with ensembles) is computed for all words ($n = 55654$), rare words (not among top 50 000 in training set; $n = 5442$), and OOVs (not in training set; $n = 851$).

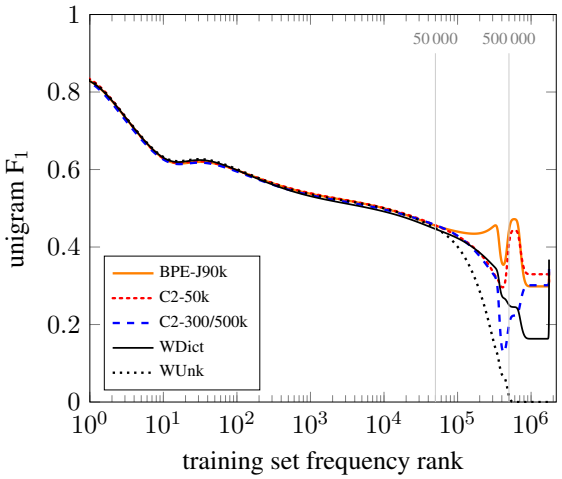


Figure 2: English→German unigram F_1 on newstest2015 plotted by training set frequency rank for different NMT systems.

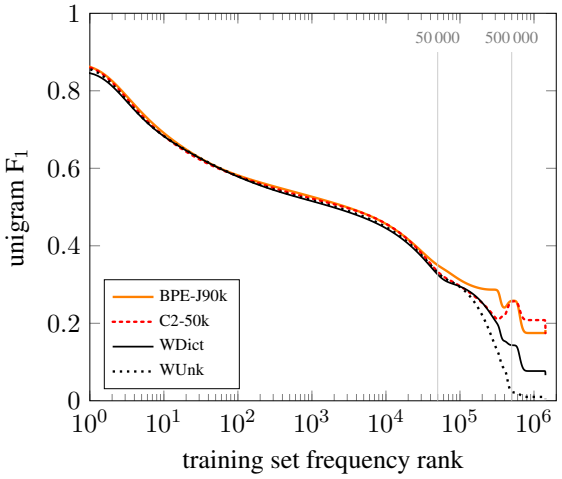


Figure 3: English→Russian unigram F_1 on newstest2015 plotted by training set frequency rank for different NMT systems.

5.2 Manual Analysis

Table 4 shows two translation examples for the translation direction English→German, Table 5 for English→Russian. The baseline system fails for all of the examples, either by deleting content (*health*), or by copying source words that should be translated or transliterated. The subword translations of *health research institutes* show that the subword systems are capable of learning translations when oversplitting (*research→Fo|rs|ch|un|g*), or when the segmentation does not match morpheme boundaries: the segmentation *Forschungs|instituten* would be linguistically more plausible, and simpler to align to the English *research institutes*, than the segmentation *Forsch|ungsinstitu|ten* in the BPE-60k system, but still, a correct translation is produced. If the systems have failed to learn a translation due to data sparseness, like for *asinine*, which should be translated as *dumm*, we see translations that are wrong, but could be plausible for (partial) loanwords (*asinine Situation→Asinin-Situation*).

The English→Russian examples show that the subword systems are capable of transliteration. However, transliteration errors do occur, either due to ambiguous transliterations, or because of non-consistent segmentations between source and target text which make it hard for the system to learn a transliteration mapping. Note that the BPE-60k system encodes *Mirzayeva* inconsistently for the two language pairs (*Mirz|ayeva*→Мир|за|ева *Mir|za|eva*). This example is still translated correctly, but we observe spurious insertions and deletions of characters in the BPE-60k system. An example is the transliteration of *rakfisk*, where a п is inserted and a к is deleted. We trace this error back to translation pairs in the training data with inconsistent segmentations, such as (*p|rak|ri|ti*→пра|крит|и

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
WDict	Forschungsinstitute
C2-50k	Fo|rs|ch|un|gs|in|st|it|ut|io|ne|n
BPE-60k	Gesundheits|forsch|ungsinstitu|ten
BPE-J90k	Gesundheits|forsch|ungsin|stitute
source	asinine situation
reference	dumme Situation
WDict	asinine situation → UNK → asinine
C2-50k	as|in|in|e situation → As|in|en|si|tu|at|io|n
BPE-60k	as|in|ine situation → A|in|line-|Situation
BPE-J90K	as|in|ine situation → As|in|in-|Situation

Table 4: English→German translation example. "|" marks subword boundaries.

system	sentence
source	Mirzayeva
reference	Мирзаева (Mirzaeva)
WDict	Mirzayeva → UNK → Mirzayeva
C2-50k	Mi|rz|ay|ev|a → Ми|рз|ае|ва (Mi|rz|ae|va)
BPE-60k	Mirz|ayeva → Мир|за|ева (Mir|za|eva)
BPE-J90k	Mir|za|yeva → Мир|за|ева (Mir|za|eva)
source	rakfisk
reference	ракфиска (rakfiska)
WDict	rakfisk → UNK → rakfisk
C2-50k	ra|kf|is|k → ра|кф|ис|к (ra|kf|is|k)
BPE-60k	rak|f|isk → пра|ф|иск (pra|f|isk)
BPE-J90k	rak|f|isk → рак|ф|иска (rak|f|iska)

Table 5: English→Russian translation examples. "|" marks subword boundaries.

(pra|krit|i)), from which the translation (*rak*→пра) is erroneously learned. The segmentation of the joint BPE system (BPE-J90k) is more consistent (*pra*|*krit*|*i*→пра|крит|и (pra|krit|i)).

6 Conclusion

The main contribution of this paper is that we show that neural machine translation systems are capable of open-vocabulary translation by representing rare and unseen words as a sequence of subword units.[14] This is both simpler and more effective than using a back-off translation model. We introduce a variant of byte pair encoding for word segmentation, which is capable of encoding open vocabularies with a compact symbol vocabulary of variable-length subword units. We show performance gains over the baseline with both BPE segmentation, and a simple character bigram segmentation.

Our analysis shows that not only out-of-vocabulary words, but also rare in-vocabulary words are translated poorly by our baseline NMT system, and that reducing the vocabulary size of subword models can actually improve performance. In this work, our choice of vocabulary size is somewhat arbitrary, and mainly motivated by comparison to prior work. One avenue of future research is to learn the optimal vocabulary size for a translation task, which we expect to depend on the language pair and amount of training data, automatically. We also believe there is further potential in bilingually informed segmentation algorithms to create more alignable subword units, although the segmentation algorithm cannot rely on the target text at runtime.

While the relative effectiveness will depend on language-specific factors such as vocabulary size, we believe that subword segmentations are suitable for most language pairs, eliminating the need for large NMT vocabularies or back-off models.

[14]The source code of the segmentation algorithms is available at `https://github.com/rsennrich/subword-nmt`.

Acknowledgments

We thank Maja Popović for her implementation of CHRF, with which we verified our re-implementation. The research presented in this publication was conducted in cooperation with Samsung Electronics Polska sp. z o.o. - Samsung R&D Institute Poland. This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 645452 (QT21).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Issam Bazzi and James R. Glass. 2000. Modeling out-of-vocabulary words for robust speech recognition. In *Sixth International Conference on Spoken Language Processing, ICSLP 2000 / INTERSPEECH 2000*, pages 401–404, Beijing, China.

Jan A. Botha and Phil Blunsom. 2014. Compositional Morphology for Word Representations and Language Modelling. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, Beijing, China.

Rohan Chitnis and John DeNero. 2015. Variable-Length Word Encodings for Neural Translation Models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Mathias Creutz and Krista Lagus. 2002. Unsupervised Discovery of Morphemes. In *Proceedings of the ACL-02 Workshop on Morphological and Phonological Learning*, pages 21–30. Association for Computational Linguistics.

Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. 2014. Integrating an Unsupervised Transliteration Model into Statistical Machine Translation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014*, pages 148–153, Gothenburg, Sweden.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

Philip Gage. 1994. A New Algorithm for Data Compression. *C Users J.*, 12(2):23–38, February.

Barry Haddow, Matthias Huck, Alexandra Birch, Nikolay Bogoychev, and Philipp Koehn. 2015. The Edinburgh/JHU Phrase-based Machine Translation Systems for WMT 2015. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 126–133, Lisbon, Portugal. Association for Computational Linguistics.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On Using Very Large Target Vocabulary for Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent Continuous Translation Models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Seattle. Association for Computational Linguistics.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-Aware Neural Language Models. *CoRR*, abs/1508.06615.

Philipp Koehn and Kevin Knight. 2003. Empirical Methods for Compound Splitting. In *EACL '03: Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics*, pages 187–193, Budapest, Hungary. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL-2007 Demo and Poster Sessions*, pages 177–180, Prague, Czech Republic. Association for Computational Linguistics.

Franklin M. Liang. 1983. *Word hy-phen-a-tion by com-put-er*. Ph.D. thesis, Stanford University, Department of Linguistics, Stanford, CA.

Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015a. Finding Function in Form: Compositional Character Models for Open Vocabulary Word Representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530, Lisbon, Portugal. Association for Computational Linguistics.

Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W. Black. 2015b. Character-based Neural Machine Translation. *ArXiv e-prints*, November.

Thang Luong, Richard Socher, and Christopher D. Manning. 2013. Better Word Representations with Recursive Neural Networks for Morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 104–113.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015a. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. 2015b. Addressing the Rare Word Problem in Neural Machine Translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, Stefan Kombrink, and Jan Cernocký. 2012. Subword Language Modeling with Neural Networks. Unpublished.

Graham Neubig, Taro Watanabe, Shinsuke Mori, and Tatsuya Kawahara. 2012. Machine Translation without Words through Substring Alignment. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 1: Long Papers*, pages 165–174.

Sonja Nießen and Hermann Ney. 2000. Improving SMT quality with morpho-syntactic analysis. In *18th Int. Conf. on Computational Linguistics*, pages 1081–1085.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013*, pages 1310–1318, Atlanta, USA.

Maja Popović. 2015. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, Lisbon, Portugal. Association for Computational Linguistics.

Rico Sennrich and Barry Haddow. 2015. A Joint Dependency Model of Morphological and Syntactic Structure for Statistical Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2081–2087, Lisbon, Portugal. Association for Computational Linguistics.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised Multilingual Learning for Morphological Segmentation. In *Proceedings of ACL-08: HLT*, pages 737–745, Columbus, Ohio. Association for Computational Linguistics.

David Stallard, Jacob Devlin, Michael Kayser, Yoong Keok Lee, and Regina Barzilay. 2012. Unsupervised Morphology Rivals Supervised Morphology for Arabic MT. In *The 50th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, July 8-14, 2012, Jeju Island, Korea - Volume 2: Short Papers*, pages 322–327.

Miloš Stanojević, Amir Kamran, Philipp Koehn, and Ondřej Bojar. 2015. Results of the WMT15 Metrics Shared Task. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 256–273, Lisbon, Portugal. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pages 3104–3112, Montreal, Quebec, Canada.

Jörg Tiedemann. 2009. Character-based PSMT for Closely Related Languages. In *Proceedings of 13th Annual Conference of the European Association for Machine Translation (EAMT'09)*, pages 12–19.

Jörg Tiedemann. 2012. Character-Based Pivot Translation for Under-Resourced Languages and Domains. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 141–151, Avignon, France. Association for Computational Linguistics.

David Vilar, Jan-Thorsten Peter, and Hermann Ney. 2007. Can We Translate Letters? In *Second Workshop on Statistical Machine Translation*, pages 33–39, Prague, Czech Republic. Association for Computational Linguistics.

Sami Virpioja, Jaakko J. Väyrynen, Mathias Creutz, and Markus Sadeniemi. 2007. Morphology-Aware Statistical Machine Translation Based on Morphs Induced in an Unsupervised Manner. In *Proceedings of the Machine Translation Summit XI*, pages 491–498, Copenhagen, Denmark.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *CoRR*, abs/1212.5701.

Implicit Discourse Relation Detection via a Deep Architecture with Gated Relevance Network

Jifan Chen, Qi Zhang, Pengfei Liu, Xipeng Qiu, Xuanjing Huang
Shanghai Key Laboratory of Data Science
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
jfchen14,qz,pfliu14,xpqiu,xjhuang@fudan.edu.cn

Abstract

Word pairs, which are one of the most easily accessible features between two text segments, have been proven to be very useful for detecting the discourse relations held between text segments. However, because of the data sparsity problem, the performance achieved by using word pair features is limited. In this paper, in order to overcome the data sparsity problem, we propose the use of word embeddings to replace the original words. Moreover, we adopt a gated relevance network to capture the semantic interaction between word pairs, and then aggregate those semantic interactions using a pooling layer to select the most informative interactions. Experimental results on Penn Discourse Tree Bank show that the proposed method without using manually designed features can achieve better performance on recognizing the discourse level relations in all of the relations.

1 Introduction

In a well-written document, no unit of the text is completely isolated, discourse relations describe how two units (e.g. clauses, sentences, and larger multi-clause groupings) of discourse are logically connected. Many downstream NLP applications such as opinion mining, summarization, and event detection, can benefit from those relations.

The task of automatically identify discourse relation is relatively simple when explicit connectives such as *however* and *because* are given (Pitler et al., 2009). However, the identification becomes much more challenging when such connectives are missing. In fact, such implicit discourse relations outnumber explicit relations in naturally occurring text, and identify those relations have been shown to be the performance bottleneck of an end-to-end discourse parser (Lin et al., 2014).

Most of the existing researches used rich linguistic features and supervised learning methods to achieve the task (Soricut and Marcu, 2003; Pitler et al., 2009; Rutherford and Xue, 2014). Among their works, word pairs are heavily used as an important feature, since word pairs like *(warm,cold)* might directly trigger a contrast relation. However, because of the data sparsity problem (McKeown and Biran, 2013) and the lack of metrics to measure the semantic relation between those pairs, which is so-called the *semantic gap* problem (Zhao and Grosky, 2002), the classifiers based on word pairs in the previous studies did not work well. Moreover, some text segment pairs are more complicated, it is hard to determine the relation held between them using only word pairs. Consider the following sentence pair with a *casual* relation as an example:

S1: Psyllium's not a good crop.

S2: You get a rain at the wrong time and the crop is ruined.

Intuitively, *(good, wrong)* and *(good, ruined)*, seem to be the most informative word pairs, and it is likely that they will trigger a contrast relation. Therefore, we can see that another main disadvantage of using word pairs is the lack of contextual information, and using n-gram pairs will again suffer from data sparsity problem.

Recently, the distributed word representations (Bengio et al., 2006; Mikolov et al., 2013) have shown an advantage when dealing with data sparsity problem (Braud and Denis, 2015), and many deep learning based models are generating substantial interests in text semantic matching and have achieved some significant progresses (Hu

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1726–1735,
Berlin, Germany, August 7-12, 2016.

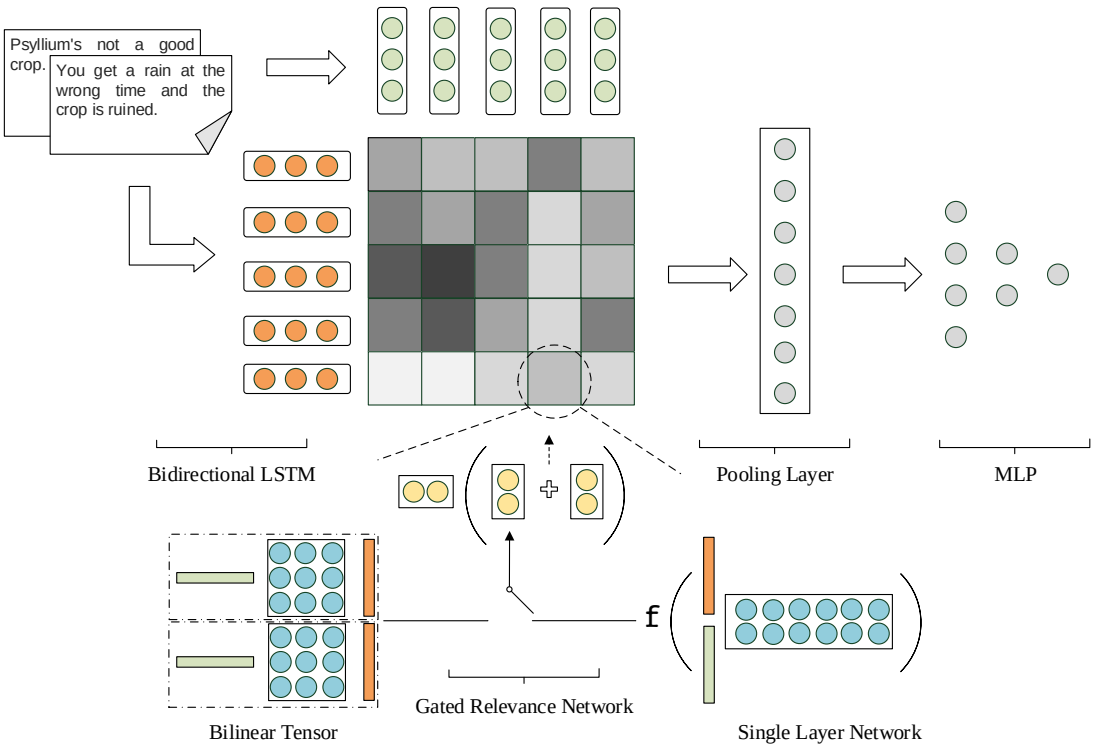


Figure 1: The processing framework of the proposed approach.

et al., 2014; Qiu and Huang, 2015; Wan et al., 2015). Inspired by their work, we in this paper propose the use of word embeddings to replace the original words in the text segments to fight against the data sparsity problem. Further more, in order to preserve the contextual information around the word embeddings, we encode the text segment to its positional representation via a recurrent neural network, specifically, we use a bidirectional LSTM (Hochreiter and Schmidhuber, 1997). Then, to overcome the *semantic gap*, we propose the use of a gated relevance network to capture the semantic interaction between those positional representations. Finally, all the interactions generated by the relevance network are fed to a max pooling layer to get the strongest interactions. We then aggregate them to predict the discourse relation through a multi-layer perceptron (MLP). Our model is trained end to end by Back-Propagation and Adagrad.

The main contribution of this paper can be summarized as follows:

- We use word embeddings to replace the original words in the text segments to overcome data sparsity problem. In order to preserve the contextual information, we further encode the text segment to its positional representation through a recurrent neural network.
- To deal with the semantic gap problem, we adopt a gated relevance network to capture the semantic interaction between the intermediate representations of the text segments.
- Experimental results on PDTB (Prasad et al., 2008) show that the proposed method can achieve better performance in recognizing discourse level relations in all of the relations than the previous methods.

2 The Proposed Method

The architecture of our proposed method is shown in figure 1. In the following of this section, we will illustrate the details of the proposed framework.

2.1 Embedding Layer

To model the sentences with neural model, we firstly need to transform the one-hot representation of word into the distributed representation. All words of two text segments X and Y will be mapped into low dimensional vector representations, which are taken as input of the network.

Through this layer, we can filter the words appear in low frequency, and we then map these words to a special OOV (out of vocabulary) word embedding. In addition, all the text segments in our experiment are padded to have the same length.

2.2 Sentence Modeling with LSTM

Long Short-Term Memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN), and specifically addresses the issue of learning long-term dependencies. Given a variable-length sentence $S = (x_0, x_1, ..., x_T)$, LSTM processes it by incrementally adding up new content into a single slot of maintained memory, with gates controlling the extent to which new content should be memorized, old content should be erased and current content should be exposed. At position t, the memory c_t and the hidden state h_t are updated with the following equations:

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} T_{\mathbf{A},\mathbf{b}} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix}, \quad (1)$$

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (3)$$

where i_t, f_t, o_t, denote the input, forget and output gate at time step t respectively, and $T_{\mathbf{A},\mathbf{b}}$ is an affine transformation which depends on parameters of the network $\mathbf{A}$ and $\mathbf{b}$. σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication.

Notice that the LSTM defined above only get context information from the past. However, context information from the future could also be crucial. To capture the context from both past and the future, we propose to use the bidirectional LSTM (Schuster and Paliwal, 1997). Bidirectional LSTM preserves the previous and future context information by two separate LSTMs, one encodes the sentence from start to the end, and the other encodes the sentence from end to the start. Therefore, at each position t of the sentence, we can obtain two representations $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$. It is natural to concatenate them to get the intermediate representation at position t, i.e. $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$. A illustration for the bidirectional LSTM are shown in Figure 2.

Given a sentence $S = (x_0, x_1, ..., x_T)$, we can now encode it with a bidirectional LSTM, and replace the word w_t with h_t, we can interpret h_t as a representation summarizing the word at position t and its contextual information.

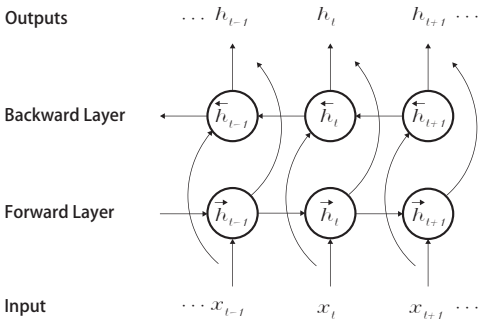


Figure 2: A illustration of bidirectional LSTM.

2.3 Gated Relevance Network

Given two text segments $X = x_1, x_2, ..., x_n$ and $Y = y_1, y_2, ..., y_m$, after the encoding procedure with a bidirectional LSTM, we can get their positional representation $X_h = x_{h_1}, x_{h_2}, ..., x_{h_n}$ and $Y_h = y_{h_1}, y_{h_2}, ..., y_{h_m}$. We then compute the relevance score between every intermediate representation pair x_{h_i} and y_{h_j} with dimension d_h. Traditional ways to measure their relevance includes cosine distance, bilinear model (Sutskever et al., 2009; Jenatton et al., 2012), single layer neural network (Collobert and Weston, 2008), etc. We then illustrate the bilinear model and the single layer neural network in details.

Bilinear Model is defined as follows:

$$s(h_{x_i}, h_{y_j}) = h_{x_i}^T M h_{y_j}, \quad (4)$$

where the only parameter $M \in \mathbf{R}^{d_h \times d_h}$. The bilinear model is a simple but efficient way to incorporate the strong linear interactions between two vectors, while the main weakness of it is the lack of ability to deal with nonlinear interaction.

Single Layer Network is defined as:

$$s(h_{x_i}, h_{y_j}) = u^T f(V \begin{bmatrix} h_{x_i} \\ h_{y_j} \end{bmatrix} + b), \quad (5)$$

where f is a standard nonlinearity applied element-wise, $V \in \mathbf{R}^{k \times 2d_h}$, $b \in \mathbf{R}^k$, and $u \in \mathbf{R}^k$. The single layer network could capture nonlinear interaction, while at the expense of a weak interaction between two vectors.

Each of the two models have its own advantages, and they can not take the place of each other.

In our work, we propose to incorporate the two models through the gate mechanism, so that the model is more powerful to capture more complex semantic interactions. The incorporated model, namely **gated relevance network** (GRN), is defined as:

$$s(h_{x_i}, h_{y_j}) = u^T(g \odot h_{x_i}^T M^{[1:r]} h_{y_j} + (1-g) \odot f(V \begin{bmatrix} h_{x_i} \\ h_{y_j} \end{bmatrix}) + b), \quad (6)$$

where f is a standard nonlinearity applied element-wise, $M^{[1:r]} \in \mathbf{R}^{r \times d_h \times d_h}$ is a bilinear tensor and the tensor product $h_{x_i}^T M^{[1:r]} h_{y_j}$ results in a vector $m \in \mathbf{R}^r$, where each entry is computed by one slice $k = 1, 2, ..., r$ of the tensor: $m_k = h_{x_i}^T M^k h_{y_j}$, $V \in \mathbf{R}^{r \times 2d_h}$, $b \in \mathbf{R}^r$, and $u \in \mathbf{R}^r$, g is a gate expressing how the output is produced by the linear and nonlinear semantic interactions between the input, defined as:

$$g = \sigma(W_g \begin{bmatrix} h_{x_i} \\ h_{y_j} \end{bmatrix} + b_g), \quad (7)$$

where $W_g \in \mathbf{R}^{r \times 2d_h}$, $b \in \mathbf{R}^r$ and σ denotes the logistic sigmoid function.

The gated relevance network is a little bit similar to the Neural Tensor Network (NTN) proposed by Socher et al. (2013):

$$s(h_{x_i}, h_{y_j}) = u^T f(h_{x_i}^T M^{[1:r]} h_{y_j} + V \begin{bmatrix} h_{x_i} \\ h_{y_j} \end{bmatrix} + b). \quad (8)$$

Compared with NTN, the main advantage of our model is we use a gate to tell how the linear and nonlinear interaction should be combined, while in NTN, the interaction generated by bilinear model and single layer network are treated equally. Also, NTN feeds the incorporated interaction to a nonlinearity, while we are not.

As we can see, for each pair of the intermediate representation, the gated relevance network will produce a semantic interaction score, thus, the entire output of two text segments is an interaction score matrix.

2.4 Max-Pooling Layer and MLP

The relation between two text segments is often determined by some strong semantic interactions, therefore, we adopt max-pooling strategy which partitions the score matrix as shown in Figure 1 into a set of non-overlapping sub-regions, and for each such sub-region, outputs the maximum value. The pooling scores are further reshaped to a vector and fed to a multi-layer perceptron (MLP). More specifically, the vector obtained by the pooling layer is fed into a full connection hidden layer to get a more abstractive representation first, and then connect to the output layer. For the task of classification, the outputs are probabilities of different classes, which is computed by a softmax function after the fully-connected layer. We name the full architecture of our model **Bi-LSTM+GRN**.

2.5 Model Training

Given a text segment pair (X, Y) and its label l, the training objective is to minimize the cross-entropy of the predicted and the true label distributions, defined as:

$$L(X, Y; \boldsymbol{l}, \hat{\boldsymbol{l}}) = -\sum_{j=1}^{C} \boldsymbol{l}_j \log(\hat{\boldsymbol{l}}_j), \quad (9)$$

where $\boldsymbol{l}$ is one-hot representation of the ground-truth label l; $\hat{\boldsymbol{l}}$ is the predicted probabilities of labels; C is the class number.

To minimize the objective, we use stochastic gradient descent with the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatches. The parameter update for the i-th parameter $\theta_{t,i}$ at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i}, \quad (10)$$

where α is the initial learning rate and $g_\tau \in \mathbf{R}^{|\theta_{\tau,i}|}$ is the gradient at time step τ for parameter $\theta_{\tau,i}$.

3 Experiment

3.1 Dataset

The dataset we used in this work is Penn Discourse Treebank 2.0 (Prasad et al., 2008), which is one of the largest available annotated corpora of discourse relations. It contains 40,600 relations, which are manually annotated from the same 2,312 Wall Street Journal (WSJ) articles as the Penn Treebank. We follow the recommended section partition of PDTB 2.0, which is to use sections 2-20 for training, sections 21-22 for testing and the other sections for validation (Prasad et al., 2008). For comparison with the previous work (Pitler et al., 2009; Zhou et al., 2010; Park

Table 1: The unbalanced sample distribution of PDTB.

Relation	Train	Dev	Test
Comparison	1894	401	146
Contingency	3281	628	276
Expansion	6792	1253	556
Temporal	665	93	68

Table 2: Hyperparameters for our model in the experiment.

Word Embedding size	$n_w = 50$
Initial learning rate	$\rho = 0.01$
Minibatch size	$m = 32$
Pooling Size	$(p, q) = (3, 3)$
Number of tensor slice	$r = 2$

and Cardie, 2012; Rutherford and Xue, 2014; Ji and Eisenstein, 2015), we train four binary classifiers to identify each of the top level relations, the *EntRel* relations are merged with *Expansion* relations. For each classifier, we use an equal number of positive and negative samples as training data, because each of the relations except *Expansion* is infrequent (Pitler et al., 2009) as what shows in Table 1. The negative samples were chosen randomly from training sections 2-20.

3.2 Experiment Protocols

In this part, we will mainly introduce the experiment settings, including baselines and parameter setting.

3.2.1 Baselines

The baselines for comparison with our proposed method are listed as follows:

- LSTM: We use two single LSTM to encode the two text segments, then concatenate them and feed to a MLP to do the relation detection.
- Bi-LSTM: We use two single bidirectional LSTM to encode the two text segments, then concatenate them and feed to a MLP to do the relation detection.
- Word+NTN: We use the neural tensor defined in (8) to capture the semantic interaction scores between every word embedding pair, the rest of the method is the same as our proposed method.
- LSTM+NTN: We use two single LSTM to generate the positional text segments representation. The rest of the method is the same as Word-NTN.
- BLSTM+NTN: We use two single bidirectional LSTM to generate the positional text segments representation. The rest of the method is the same as Word-NTN.
- Word+GRN: We use the gated relevance network proposed in this paper to capture the semantic interaction scores between every word embedding pair of the two text segments. The rest of the method is the same as our model.
- LSTM+GRN: We use the gated relevance network proposed in this paper to capture the semantic interaction scores between every intermediate representation pair of the two text segments generated by LSTM. The rest of the method is the same as our model.

3.2.2 Parameter Setting

For the initialization of the word embeddings used in our model, we use the 50-dimensional pre-trained embeddings provided by Turian et al. (2010), and the embeddings are fixed during training. We only preserve the top 10,000 words according to its frequency of occurrence in the training data, all the text segments are padded to have the same length of 50, the intermediate representations of LSTM are also set to 50. The other parameters are initialized by randomly sampling from uniform distribution in [-0.1,0.1].

For other hyperparameters of our proposed model, we take those hyperparameters that achieved best performance on the development set, and keep the same parameters for other competitors. The final hyper-parameters are show in Table 2.

3.3 Result

The results on PDTB are show in Table3, from the results, we have several experiment findings.

First of all, it is easy to notice that LSTM and Bi-LSTM achieve lower performance than all of the methods of using a tensor to capture the semantic interactions between word pairs and the intermediate representation pairs. Because the main

Table 3: The performances of different approaches on the PDTB.

	Comparison	Contingency	Expansion	Temporal
(Pitler et al., 2009)	21.96%	47.13%	76.42%	16.76%
(Zhou et al., 2010)	31.79%	47.16%	70.11%	20.30%
(Park and Cardie, 2012)	31.32%	49.82%	79.22%	26.57%
(Rutherford and Xue, 2014)	39.70%	54.42%	80.44%	28.69%
(Ji and Eisenstein, 2015)	35.93%	52.78%	80.02%	27.63%
LSTM	31.78%	45.39%	75.10%	19.65%
Bi-LSTM	31.97%	45.66%	75.13%	20.02%
Word+NTN	32.18%	46.45%	77.64%	21.60%
LSTM+NTN	36.82%	50.09%	79.88%	26.54%
Bi-LSTM+NTN	39.36%	53.74%	80.02%	28.41%
Word+GRN	32.67%	46.52%	77.68%	21.21%
LSTM+GRN	38.13%	52.25%	79.96%	27.15%
Bi-LSTM+GRN	**40.17**%	**54.76**%	**80.62**%	**31.32**%

disadvantage of using LSTM and Bi-LSTM to encode the text segment into a single representation is that some important local information such as key words can not be fully preserved when compressing a long sentence into a single representation.

Second, the performance improves a lot when using LSTM and Bi-LSTM to encode the text segments to positional representations instead of using word representations directly. We conclude it is mainly because the following two reasons: for one thing, some words are important only when they are associated with their context, for the other, the intermediate representations are the high-level representation of the sentence at each position, there is no doubt for they can obtain much more semantic information than the words along. In addition, Bi-LSTM also takes the future information of the text segments into consideration, resulting in a consistently better performance than LSTM.

Third, take a comparison to the methods using NTN and the methods using GRN, we can find that the GRN performs consistently better. Such results show that the gate we proposed to combine the information of two aspects is actually useful.

At last, our proposed model, namely, Bi-LSTM+GRN achieves best performance on all of the relations. It not only shows the interaction between word pairs is useful, but also shows the way we proposed to capture such information is useful too. Further more, compared with the previous methods (Pitler et al., 2009; Park and Cardie, 2012; Rutherford and Xue, 2014), which used either a lot of complex textual features and contextual information about the two text segments or a larger unannotated corpus to help the prediction, our model only uses the the information of the two text segments themselves, but yet achieves better performance. It demonstrates that our model is powerful in modeling the discourse relations.

3.4 Parameter Sensitivity

In this section, we evaluate our model through different settings of the proposed gated relevance network, the other hyperparameters are the same as mentioned above and we use a bidirectional LSTM to encode the text segments. The settings are shown as follows:

- **GRN-1** We set the parameters $r = 1$, $M^{[1]} = I$, $V = 0$, $b = 0$ and $g = 1$. The model can be regarded as cosine similarity.

	Cmp	Ctg	Exp	Tmp
GRN-1	34.01%	50.23%	78.66%	21.33%
GRN-2	35.74%	50.91%	79.40%	23.28%
GRN-3	34.15%	51.78%	79.33%	26.16%
GRN-4	37.18%	52.36%	79.86%	25.60%
GRN-5	**40.17**%	**54.76**%	**80.62**%	**31.32**%
GRN-6	38.26%	52.08%	80.02%	28.51%

Table 4: Comparison of our model with different parameter settings to the gated relevance network. *Cmp* denotes the comparison relation, *Ctg* denotes the contingency relation, *Exp* denotes the expansion relation and *Tmp* denotes the temporal relation.

- **GRN-2** We set the parameters $r = 1$, $V = 0$, $b = 0$, and $g = 1$. The model can be regarded as the bilinear model.
- **GRN-3** We set the parameters $r = 1$, $M^{[1]} = 0$, and $g = 0$. The model can be regarded as a single layer network.
- **GRN-4** We set the parameters $r = 1$. This model is the full GRN model.
- **GRN-5** We set the parameters $r = 2$. This model is the full GRN model.
- **GRN-6** We set the parameters $r = 3$. This model is the full GRN model.

The results for different parameter settings are shown in Table 4. It is obvious that **GRN-1** achieves a relatively lower performance, showing that the cosine similarity is not enough to capture the complex semantic interaction. Take a comparison on **GRN-2** and **GRN-3**, we can see that **GRN-2** outperforms **GRN-3** on *Comparison* and *Expansion* relation, while achieves a lower performance on the other two relations, moreover, the combination method **GRN-4** outperforms both of the methods, demonstrating that the semantic interactions captured by the bilinear model and the single layer network are different. Hence, they can not take the place of each other, and it is reasonable to use a gate to combine them.

Among the methods of using the full GRN model, **GRN-5** which has 2 bilinear tensor slices achieves the best performance. We explain this phenomenon on two aspects, on one hand, we can see each slice of the bilinear tensor as being responsible for one type of the relation, a bilinear tensor with 2 slices is more suitable for training a binary classifier than the original bilinear model. On the other hand, increasing the number of slices will increase the complexity of the model, thus making it harder to train.

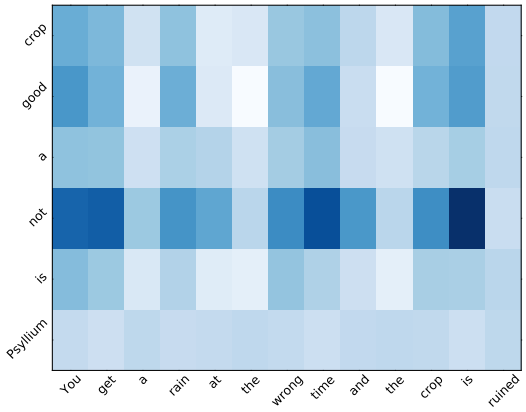
3.5 Case Study

In this section, we go back to the example mentioned above to show see what information between the text segment pairs is captured, and how the positional sentence representations affect the performance of our model.

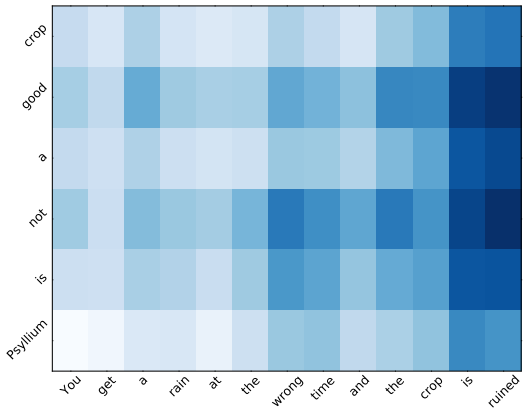
The examples is listed below:

S1: Psyllium's not a good crop.

S2: You get a rain at the wrong time and the crop is ruined.



(a) Word+GRN



(b) Bi-LSTM+GRN

Figure 3: A visualization of the interaction score matrix between two relatively complex sentences. The darker patches denote the corresponding scores generated by the gated relevance network are higher.

In this case, the relation between the sentence pair is *Contingency*, and the implicit connective annotated by human is "because". The pair is likely to be classified to a wrong contrast relation if we only focus on the informative word pairs *(good,wrong)* and *(good,ruined)*. It is mainly because their relation is highly depended on the semantic of the whole sentence, and the words should be considered with their context.

Figure 3 explains this phenomenon, in Figure 3a, we can see that the word pairs which associate with "not" get high scores, scores on the other pairs are relatively arbitrary. It demonstrates the word embedding model failed to learn which part of the sentence should be focused, although the useless word such as "Psyllium" and "a" are

ignored, thus making it harder to identify the relation.

Take Figure 3b for a comparison, from the figure we can observe the pairs that associate with "not" and "good" which are import context to determine the semantic of the sentence get much higher scores. Moreover, the scores increase along with the sentence encoding procedure, especially when the last informative word "ruined" appears. Once again, some useless word are also ignored by this model. It demonstrates the bidirectional LSTM we used in our model could encode the contextual information to the intermediate representations, thus these information could help to determine which part of the two sentence should be focused when identifying their relation.

4 Related Work

Discourse relations, which link clauses in text, are used to represent the overall text structure. Many downstream NLP tasks such as text summarization, question answering, and textual entailment can benefit from the task. Along with the increasing requirement, many works have been constructed to automatically identify these relations from different aspects (Pitler et al., 2008; Pitler et al., 2009; Zhou et al., 2010; McKeown and Biran, 2013; Rutherford and Xue, 2014; Xue et al., 2015).

For training and comparing the performance of different methods, the Penn Discourse Treebank (PDTB) 2.0, which is large annotated discourse corpuses, were released in 2008 (Prasad et al., 2008). The annotation methodology of it follows the lexically grounded, predicate-argument approach. In PDTB, the discourse relations were predefined by Webber (2004). PDTB-styled discourse relations hold in only a local contextual window, and these relations are organized hierarchically. Also, every relation in PDTB has either an *explicit* or an *implicit* marker. Since explicit relations are easy to identify (Pitler et al., 2008), existing methods achieved good performance on the relations with explicit maker. In recent years, researchers mainly focused on implicit relations. For easily comparing with other methods, in this work, we also use PDTB as the training and testing corpus.

As we mentioned above, various approaches have been proposed to do the task. Pitler et al. (2009) proposed to train four binary classifiers using word pairs as well as other rich linguistic features to automatically identify the top-level PDTB relations. Park and Cardie (2012) achieved a higher performance by optimizing the feature set. McKeown and Biran (2013) aims at solving the data sparsity problem, and they extended the work of Pitler et al. (2009) by aggregating word pairs. Rutherford and Xue (2014) used Brown clusters and coreferential patterns as new features and improved the baseline a lot. Braud and Denis (2015) compared different word representations for implicit relation classification. The word pairs feature have been studied by all of the work above, showing its importance on discourse relation. We follow their work, and incorporate word embedding to deal with this problem.

There also exist some work performing this task from other perspectives. Zhou et al. (2010) studied the problem from predicting *implicit marker*. They used a language model to add implicit markers as an additional feature to improve performance. Their approach can be seen as a semi-supervised method. Ji and Eisenstein (2015) computes distributed meaning representations for each discourse argument by composition up the syntactic parse tree. Chen et al. (2016) used vector offsets to represent this relation between sentence pairs, and aggregate this offsets through the Fisher vector. Liu et al. (2016) used a a mutil-task deep learning framework to deal with this problem, they incorporate other similar corpus to deal with the data sparsity problem.

Most of the previous works mentioned above used rich linguistic features and supervised learning methods to achieve the task. In this paper, we propose a deep architecture, which does not need these manually selected features and additional linguistic knowledge base to do it.

5 Conclusion

In this work, we propose to use word embeddings to fight against the data sparsity problem of word pairs. In order to preserve contextual information, we encode a sentence to its positional representation via a recurrent neural network, specifically, a LSTM. To solve the semantic gap between the word pairs, we propose to use a gated relevance network which incorporates both the linear and nonlinear interactions between pairs. Experiment results on PDTB show the proposed model outperforms the existing methods using traditional fea-

tures on all of the relations.

Acknowledgement

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Chloé Braud and Pascal Denis. 2015. Comparing word representations for implicit discourse relation classification. In *Empirical Methods in Natural Language Processing (EMNLP 2015)*.

Jifan Chen, Qi Zhang, Pengfei Liu, and Xuanjing Huang. 2016. Discourse relations detection via a mixed generative-discriminative framework. In *AAAI*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems*, pages 2042–2050.

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Yangfeng Ji and Jacob Eisenstein. 2015. One vector is not enough: Entity-augmented distributed semantics for discourse relations. *Transactions of the Association for Computational Linguistics*, 3:329–344.

Ziheng Lin, Hwee Tou Ng, and Min-Yen Kan. 2014. A pdtb-styled end-to-end discourse parser. *Natural Language Engineering*, 20(02):151–184.

Yang Liu, Sujian Li, Xiaodong Zhang, and Zhifang Sui. 2016. Implicit discourse relation classification via multi-task neural networks. In *AAAI*.

Kathleen McKeown and Or Biran. 2013. Aggregated word pair features for implicit discourse relation disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 69–73. The Association for Computational Linguistics.

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

Joonsuk Park and Claire Cardie. 2012. Improving implicit discourse relation recognition through feature set optimization. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 108–112. Association for Computational Linguistics.

Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K Joshi. 2008. Easily identifiable discourse relations. *Technical Reports (CIS)*, page 884.

Emily Pitler, Annie Louis, and Ani Nenkova. 2009. Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 683–691. Association for Computational Linguistics.

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

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1305–1311.

Attapol Rutherford and Nianwen Xue. 2014. Discovering implicit discourse relations through brown cluster pair representation and coreference patterns. In *EACL*, volume 645, page 2014.

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

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 149–156. Association for Computational Linguistics.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan R Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, pages 1821–1828.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394. Association for Computational Linguistics.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2015. A deep architecture for semantic matching with multiple positional sentence representations. *arXiv preprint arXiv:1511.08277*.

Bonnie Webber. 2004. D-ltag: Extending lexicalized tag to discourse. *Cognitive Science*, 28(5):751–779.

Nianwen Xue, Hwee Tou Ng, Sameer Pradhan, Rashmi PrasadO Christopher Bryant, and Attapol T Rutherford. 2015. The conll-2015 shared task on shallow discourse parsing. In *Proceedings of CoNLL*, page 2.

Rong Zhao and William I Grosky. 2002. Narrowing the semantic gap-improved text-based web document retrieval using visual features. *Multimedia, IEEE Transactions on*, 4(2):189–200.

Zhi-Min Zhou, Yu Xu, Zheng-Yu Niu, Man Lan, Jian Su, and Chew Lim Tan. 2010. Predicting discourse connectives for implicit discourse relation recognition. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 1507–1514. Association for Computational Linguistics.

Model Architectures for Quotation Detection

Christian Scheible, Roman Klinger and **Sebastian Padó**
Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
{scheibcn,klinger,pado}@ims.uni-stuttgart.de

Abstract

Quotation detection is the task of locating spans of quoted speech in text. The state of the art treats this problem as a sequence labeling task and employs linear-chain conditional random fields. We question the efficacy of this choice: The Markov assumption in the model prohibits it from making joint decisions about the begin, end, and internal context of a quotation. We perform an extensive analysis with two new model architectures. We find that (a), simple boundary classification combined with a greedy prediction strategy is competitive with the state of the art; (b), a semi-Markov model significantly outperforms all others, by relaxing the Markov assumption.

1 Introduction

Quotations are occurrences of reported speech, thought, and writing in text. They play an important role in computational linguistics and digital humanities, providing evidence for, e.g., speaker relationships (Elson et al., 2010), inter-speaker sentiment (Nalisnick and Baird, 2013) or politeness (Faruqui and Pado, 2012). Due to a lack of general-purpose automatic systems, such information is often obtained through manual annotation (e.g., Agarwal et al. (2012)), which is labor-intensive and costly. Thus, models for *automatic quotation detection* form a growing research area (e.g., Pouliquen et al. (2007); Pareti et al. (2013)).

Quotation detection looks deceptively simple, but is challenging, as the following example shows:

> [The pipeline], the company said, [would be built by a proposed joint venture ..., and Trunkline ... will "build and operate" the system ...].[1]

[1]Penn Attributions Relation Corpus (PARC), wsj_0260

Note that quotations can (i) be signalled by lexical cues (e.g., communication verbs) without quotation marks, (ii) contain misleading quotation marks; (iii) be discontinuous, and (iv) be arbitrarily long.

Early approaches to quotation detection use hand-crafted rules based on syntactic markers (Pouliquen et al., 2007; Krestel et al., 2008). While yielding high precision, they suffered from low recall. The state of the art (Pareti et al., 2013; Pareti, 2015) treats the task as a sequence classification problem and uses a linear-chain conditional random field (CRF). This approach works well for the prediction of the approximate location of quotations, but yields a lower performance detecting their exact span.

In this paper, we show that linear-chain sequence models are a sub-optimal choice for this task. The main reason is their *length*, as remarked above: Most sequence labeling tasks in NLP (such as most cases of named entity recognition) deal with spans of a few tokens. In contrast, the median quotation length on the Penn Attributions Relation Corpus (PARC, Pareti et al. (2013)) is 16 tokens and the longest span has over 100 tokens. As a result of the strong Markov assumptions that linear-chain CRFs make to ensure tractability, they cannot capture "global" properties of (almost all) quotations and are unable to make joint decisions about the begin point, end point, and content of quotations.

As our first main contribution in this paper, we propose two novel model architectures designed to investigate this claim. The first is *simpler* than the CRF. It uses token-level classifiers to predict quotation boundaries and combines the boundaries greedily to predict spans. The second model is *more expressive*. It is a semi-Markov sequence model which relaxes the Markov assumption, enabling it to consider global features of quotation spans. In our second main contribution, an analysis of the models' performances, we find that the sim-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1736–1745,
Berlin, Germany, August 7-12, 2016.

pler model is competitive with the state-of-the-art CRF. The semi-Markov model outperforms both of them significantly by 3 % F_1. This demonstrates that the relaxed Markov assumptions help improve performance. Our final contribution is to make implementations of all models publicly available.[2]

2 The Task: Quotation Detection

Problem Definition Following the terminology established by Pareti et al. (2013), we deal with the detection of *content spans*, the parts of the text that are being quoted. To locate such spans, it is helpful to first detect *cues* which often mark the beginning or end of a quotation. The following example shows an annotated sentence from the PARC corpus; each content span (CONT) is associated with exactly one cue span (CUE):

> Mr. Kaye [denies]$_{\text{CUE}}$ [the suit's charges]$_{\text{CONT}}$ and [says]$_{\text{CUE}}$ [his only mistake was taking on Sony in the marketplace]$_{\text{CONT}}$.[3]

Pareti et al. (2013) distinguish three types of quotations. *Direct* quotations are fully enclosed in quotation marks and are a verbatim reproduction of the original utterance. *Indirect* quotations paraphrase the original utterance and have no quotation marks. *Mixed* quotations contain both verbatim and paraphrase content and may thus contain quotation marks. Note that the type of a content span is assigned automatically based on its surface form using the definitions just given.

Quotation Detection as Sequence Modeling In this paper, we compare our new model architectures to the state-of-the-art approach by Pareti (2015), an extension of Pareti et al. (2013). Their system is a pipeline: Its first component is the *cue model*, a token-level k-NN classifier applied to the syntactic heads of all verb groups. After cues are detected, content spans are localized using the *content model*, a linear-chain conditional random field (CRF) which makes use of the location of cues in the document through features.

As their system is not publicly available, we reimplement it. Our cue classifier is an averaged perceptron (Collins, 2002) which we describe in more detail in the following section. It uses the features in Table 1.[4] Our content model is a CRF with BIOE labels. It uses all features from Table 1 plus features that build on the output of the cue classifier, shown in Table 2.

C1. Surface form, lemma, and PoS tag for all tokens within a window of ± 5.
C2. Bigrams of surface form, lemma, and PoS tag
C3. Shape of t_i
C4. Is any token in a window of ± 5 a named entity?
C5. Does a quotation mark open or close at t_i (determined by counting)? Is t_i within quotation marks?
C6. Is t_i in the list of reporting verbs, noun cue verbs, titles, WordNet persons or organizations, and its VerbNet class
C7. Do a sentence, paragraph, or the document begin or end at t_i, t_{i-1}, or t_{i+1}?
C8. Distance to sentence begin and end; sentence length
C9. Does the sentence contain t_i a pronoun/named entity/quotation mark?
C10. Does a syntactic constituent starts or ends at t_i?
C11. Level of t_i in the constituent tree
C12. Label and level of the highest constituent in the tree starting at t_i; label of t_i's the parent node
C13. Dependency relation with parent or any child of t_i (with and without parent surface form)
C14. Any conjunction of C5, C9, C10

Table 1: Cue detection features for a token t_i at position i, mostly derived from Pareti (2015)

S1. Is a direct or indirect dependency parent of t_i classified as a cue, in the cue list, or the phrase "according to"?
S2. Was any token in a window of ± 5 classified as a cue?
S3. Distance to the previous and next cue
S4. Does the sentence containing t_i have a cue?
S5. Conjunction of S4 and all features from C14

Table 2: Additional features for content span detection, mostly derived from Pareti (2015)

3 New Model Architectures

While Pareti (2015) apply sequence modeling for quotation detection, they do not provide an analysis what the model learns. In this paper, we follow the intuition that a linear-chain CRF mostly makes local decisions about spans, while ignoring their global structure, such as joint information about the context of the begin and end points. If this is true, then (a) a model might work as well as the CRF without learning from label sequences, and (b) a model which makes joint decisions with global information might improve over the CRF.

This motivates our two new model architectures for the task. We illustrate the way the different architectures make use of information in Figure 1. Our simpler model (GREEDY) makes strictly local classification decisions, completely ignoring

[2] http://www.ims.uni-stuttgart.de/data/qsample

[3] PARC, `wsj_2418`

[4] For replicability, we give more detailed definitions of the features in the supplementary notes.

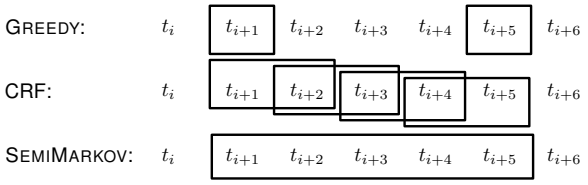


Figure 1: Information usage by model architecture. Frames indicate joint decisions on token labels.

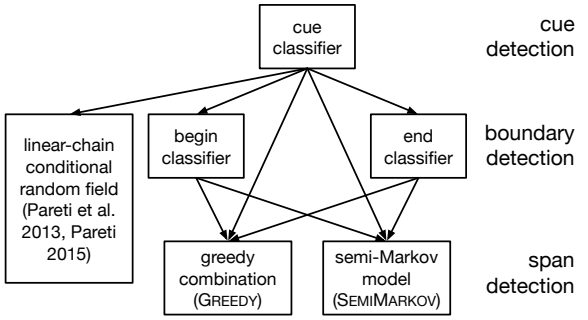


Figure 2: Information flow in all three models

those around it. The CRF is able to coordinate decisions within a window, which is propagated through Viterbi decoding. The more powerful model (SEMIMARKOV) takes the full span into account and makes a joint decision about the begin and end points.

Our intuition about the shortcomings of the CRF is based on an empirical analysis. However, to simplify the presentation, we postpone the presentation of this analysis to Section 6 where we can discuss and compare the results of all three models.

3.1 Model Decomposition and Formalization

We first introduce a common formalization for our model descriptions. Our problem of interest is *content span detection*, the task of predicting a set $\mathcal{S}$ of content spans (t_b, t_e) delimited by their begin and end tokens. The CRF solves this task by classifying tokens as begin/end/inside/outside tokens and thus solves a proxy problem. The problem is difficult because corresponding begin and end points need to be matched up over long distances, a challenge for probabilistic finite state automata such as CRFs.

In our model, *cue detection*, the task of detecting cue tokens t_c (cf. Section 2), remains the first step. However, we then decompose the content span problem solved by the CRF by introducing the intermediary task of *boundary detection*. As illustrated in Figure 2, this means identifying the sets of all *begin* and *end* tokens, t_b and t_e, ignoring their interdependencies. We then recombine these

Algorithm 1 GREEDY content span algorithm

Input: List of documents D; feature functions $\boldsymbol{f}_x$ for cue, begin, and end ($x \in c, b, e$); distance parameter $d_{\max}$; length parameter $\ell_{\max}$
Output: Content span labeling $\mathcal{S}$

$\boldsymbol{\theta}_c, \boldsymbol{\theta}_b, \boldsymbol{\theta}_e \leftarrow$ TRAINCLASSIFIERS$(D, \boldsymbol{f}_c, \boldsymbol{f}_b, \boldsymbol{f}_e)$
for d in D **do**
  $\mathcal{S} \leftarrow \varnothing$
  for token t in d **do**
    if $\boldsymbol{\theta}_c \cdot \boldsymbol{f}_c(t) > 0$ **then**
      $t_b \leftarrow$ next token right of t where $\boldsymbol{\theta}_b \cdot \boldsymbol{f}_b(t) > 0$ ▷ next begin
      $t_e \leftarrow$ next token right of t_b where $\boldsymbol{\theta}_e \cdot \boldsymbol{f}_e(t) > 0$ ▷ next end
      if $|t_b - t_c| \leq d_{\max}$ **and** $|t_e - t_b| \leq \ell_{\max}$ **and** OVERLAPPING$(t_b, t_e) = \varnothing$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{(t_b, t_e)\}$ ▷ add span

predictions with two different strategies, as detailed in Section 3.2 and Section 3.3. This decomposition has two advantages: (a), we expect that boundary detection is easier than content span detection, as we remove the combinatorial complexity of matching begin and end tokens; (b), begin, end, and cue detection are now three identical classification tasks that can be solved by the same machinery.

We model each of the three tasks (cue/begin/end detection) with a linear classifier of the form

$$\text{score}_x(t) = \boldsymbol{\theta}_x \cdot \boldsymbol{f}_x(t) \quad (1)$$

for a token t, a class $x \in \{c, b, e\}$ (for *cue*, *begin*, and *end*), a feature extraction function $\boldsymbol{f}_x(t)$, and a weight vector $\boldsymbol{\theta}_x$. We re-use the feature templates from Section 2 to remain comparable to the CRF.

We estimate all parameters $\boldsymbol{\theta}_x$ with the perceptron algorithm, and use parameter averaging (Collins, 2002). Since class imbalances, which occur in the boundary detection tasks, can have strong effects (Barandela et al., 2003), we train the perceptron with uneven margins (Li et al., 2002). This variant introduces two learning margins: τ_{-1} for the negative class and τ_{+1} for the positive class. Increasing τ_{+1} at a constant τ_{-1} increases recall (as failure to predict this class is punished more), potentially at the loss of precision, and vice versa.

3.2 Greedy Span Detection

Our first new model, GREEDY (Figure 2, bottom center), builds on the assumption that the modeling of sequence properties in a linear-chain CRF is weak enough that sequence learning can be replaced by a greedy procedure. Algorithm 1 shows how we generate a span labeling based on the output of the boundary classifiers. Starting at each cue,

we add all spans within a given distance $d_{\max}$ from the cue whose length is below a given maximum $\ell_{\max}$. If the candidate span is OVERLAPPING with any existing spans, we discard it. Analogously, we search for spans to the left of the cue. The algorithm is motivated by the structure of attribution relations: each content span has one associated cue.

3.3 Semi-Markov Span Detection

Our second model extends the CRF into a semi-Markov architecture which is able to handle *global features* of quotation span candidates (SEMIMARKOV, Figure 2 bottom right). Following previous work (Sarawagi and Cohen, 2004), we relax the Markov assumption inside spans. This allows for extracting arbitrary features on each span, such as conjunctions of features on the begin and end tokens or occurrence counts within the span.

Unfortunately, the more powerful model architecture comes at the cost of a more difficult prediction problem. Sarawagi and Cohen (2004) propose a variant of the Viterbi algorithm. This however does not scale to our application, since the maximum length of a span factors into the prediction runtime, and quotations can be arbitrarily long. As an alternative, we propose a sampling-based approach: we draw candidate spans (*proposals*) from an informed, non-uniform distribution of spans. We score these spans to decide whether they should be added to the document (*accepted*) or not (*rejected*). This way, we efficiently traverse the space of potential span assignments while still being able to make informed decisions (cf. Wick et al. (2011)).

To obtain a distribution over spans, we adapt the approach by Zhang et al. (2015). We introduce two independent probability distributions: P_b is the distribution of probabilities of a token being a *begin token*; P_e is the distribution of probabilities of a token being an *end token*. We sample a single content span proposal (DRAWPROPOSAL) by first sampling the *order* in which the boundaries are to be determined (begin token or end token first) by sampling a binary variable $d \sim \text{Bernoulli}(0.5)$. If the begin token is to be sampled first, we continue by drawing a begin token $t_b \sim P_b$ and finally draw an end token $t_e \sim P_e$ within a window of up to $\ell_{\max}$ tokens to the right of t_b. If the end token is to be sampled first, we proceed conversely. We also propose empty spans, i.e., the removal of existing spans without an replacement.

For the distributions P_b and P_e, we reuse our boundary detection models from Section 3.1. For each class $x \in \{b, e\}$ we form a distribution

$$P_x(t) \propto \exp(\text{score}_x(t)/T_x) \quad (2)$$

over the tokens t of a document using the scores from Equation 1. T_x is a temperature hyperparameter. Temperature controls the pronouncedness of peaks in the distribution. Higher temperature flattens the distribution and encourages the selection of tokens with lower scores. This is useful when exploration of the sample space is desired.

Algorithm 2 SEMIMARKOV inference algorithm

Input: Document d; probability distributions for begin and end (P_b, P_e); feature function for spans $\boldsymbol{g}$; maximum span length $\ell_{\max}$; number of proposals N
Output: Set of content spans $\mathcal{S}$

$\mathcal{S} \leftarrow \varnothing$
$\boldsymbol{\theta} \leftarrow \mathbf{0}$
for $n = 1$ to N **do**
  $(t_b, t_e) \leftarrow$ DRAWPROPOSAL(P_b, P_e)
  score $\leftarrow \boldsymbol{\theta} \cdot \boldsymbol{g}(t_b, t_e)$
  $\mathcal{O} \leftarrow$ OVERLAPPING(t_b, t_e)
  score$_{\mathcal{O}} \leftarrow \sum_{(t'_b, t'_e) \in \mathcal{O}} \boldsymbol{\theta} \cdot \boldsymbol{g}(t'_b, t'_e)$
  if score $>$ score$_{\mathcal{O}}$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \mathcal{O}$ ▷ remove overlapping
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{(t_b, t_e)\}$ ▷ accept proposal
    if ISTRAINING **and** $\neg$CORRECT(t_b, t_e) **then**
      PERCEPTRONUPDATE ▷ wrongly accepted
  else
    REJECT(t_b, t_e)
    if ISTRAINING **and** CORRECT(t_b, t_e) **then**
      PERCEPTRONUPDATE ▷ wrongly rejected

The proposed candidates enter into the decision algorithm shown in Algorithm 2. As shown, the candidates are scored using a linear model (again as defined in Equation 1). We use the features of the previous models (Table 1 and 2) on the begin and end tokens. As we now judge complete span assignments rather than local label assignments to tokens, we can add a new span-global feature function $\boldsymbol{g}(t_b, t_e)$. We introduce the features shown in Table 3. If the candidate's score is higher than the sum of scores of all spans overlapping with it, we accept it and remove all overlapping ones.

This model architecture can be seen as a modification of the pipeline of the GREEDY model (cf. Figure 2). We again detect cues and boundaries, but then make an informed decision for combining begin and end candidates. In addition, the sampler makes "soft" selections of begin and end tokens based on the model scores rather than simply accepting the classifier decisions.

For training, we again use perceptron updates (cf. Section 3.2). If the model accepts a wrong

	Setting	Direct			Indirect			Mixed			Overall		
		P	R	F	P	R	F	P	R	F	P	R	F
strict	Pareti (2015) as reported therein	94	88	91	78	56	65	67	60	63	80	63	71
	CRF (own re-implementation)	94	93	94	73	58	64	81	68	74	79_{g}	67	72
	GREEDY	92	91	91	69	59	64	72	64	68	75	67	71
	SEMIMARKOV	93	94	94	73	65	69	81	66	73	$\mathbf{79}_{g}$	$\mathbf{71}^{c}_{g}$	$\mathbf{75}^{c}_{g}$
	Combination: CRF+SEMIMARKOV	94	93	94	73	64	69	81	68	74	$\mathbf{79}_{g}$	$\mathbf{71}^{c}_{g}$	$\mathbf{75}^{c}_{g}$
partial	Pareti (2015) as reported therein	99	93	96	91	66	77	91	81	86	93	73	82
	CRF (own re-implementation)	98	96	97	87	70	77	94	83	88	90_{g}	77	83
	GREEDY	97	95	96	83	76	79	93	85	89	88	81^{c}	84
	SEMIMARKOV	97	95	96	83	75	79	92	81	86	88	80	84
	Combination: CRF+SEMIMARKOV	98	96	97	83	75	79	94	83	88	88	81^{c}	84^{c}

Table 4: Results on the test set of PARC3. Best overall strict results in bold. Models as in Figure 2. g: significantly better than GREEDY; c: significantly better than CRF (both with $\alpha = 0.05$).

G1. Numbers of named entities, lowercased tokens, commas, and pronouns inside the span
G2. Binned percentage of tokens that depend on a cue
G3. Location of the closest cue (left/right?), percentage of dependents on that cue
G4. Number of cues overlapped by the span
G5. Is there a cue before the first token and/or after the last token of the span (within the same sentence)? first or after the last token of the span?, and their conjunction
G6. Do both the first and the last token depend on a cue?
G7. Binned length of the span
G8. Does the span match a sentence exactly/off by one token?
G9. Number of sentences covered by the span
G10. Does the span match one or more constituents exactly?
G11. Is the span direct, indirect, or mixed?
G12. Is the # of quotation marks in the span odd or even?
G13. Is the span is direct and does it contain more than two quotation marks?

Table 3: Global features for content span detection

span, we perform a negative update (Line 12 in Algorithm 2). If a correct span is rejected, we make a positive update (Line 16). We iterate over the documents in random order for a fixed number E of epochs. As the sampling procedure takes long to fully label documents, we employ GREEDY to make initial assignments. This does not constitute additional supervision, as the sampler can remove any initial span and thus refute the initialization. This reduces runtime without affecting the result in practice.

4 Experimental Setup

Data We use the Penn Attribution Relations Corpus, version 3 (henceforth *PARC3*), by Pareti (2015).[5] It contains AR annotations on the Wall Street Journal part of the Penn Treebank (2,294 news documents). As in related work, we use sections 1–22 as training set, section 23 as test set, and section 24 as development set. We perform the same preprocessing as Pareti: We use gold tokenization, lemmatization, part-of-speech tags, constituency parses, gold named entity annotations (Weischedel and Brunstein, 2005), and Stanford parser dependency analyses (Manning et al., 2014).

Evaluation We report precision, recall, and micro-averaged F_1, adopting the two metrics introduced by Pareti et al. (2013): *Strict match* considers cases as correct where the boundaries of the spans match exactly. *Partial match* measures correctness as the ratio of overlap of the predicted and true spans. In both cases, we report numbers for each of the three quotation types (*direct*, *indirect*, *mixed*) and their micro averages. Like Pareti (2015), we exclude single-token content spans from the evaluation. To test for statistical significance of differences, we use the approximate randomization test (Noreen, 1989) at a significance level of $\alpha = 0.05$.

Implementation and Hyperparameters We use the CRF implementation in MALLET (McCallum, 2002). We optimize all hyperparameters of the models on the development set. Our best models use positive margins of $\tau_+ = 25$ for the boundary and $\tau_+ = 15$ for the span models, favoring recall. The SEMIMARKOV sampler uses a temperature of $T_x = 10$ for all classes. We perform 15 epochs of training after which the models have converged, and draw 1,000 samples for each document. For the GREEDY model, we obtain the best results with $d_{\max} = 30$ and $\ell_{\max} = 55$. For the SEMIMARKOV sampler, $\ell_{\max} = 75$ is optimal.

[5] Note that the data and thus the results differ from those previously published in (Pareti et al., 2013).

The high values mirror the presence of very long spans in the data.

5 Results

Cue We first evaluate the cue classifier. We obtain an F_1 of 86 %, with both precision and recall at 86 %, which is very close to the 85 % F_1 of Pareti.

CRF Table 4 summarizes the content span results. First, we compare Pareti's results to our reimplementation (the rows denoted with *Pareti (2015)* and *CRF*). There are some differences in how well the model performs on certain types of spans: while our precision is lower for indirect spans, it is higher on mixed spans. Additionally, our implementation generally has higher recall than Pareti's. Her system includes several features using proprietary lists (such as a manually curated list of titles) we were unable to obtain, and complex feature templates that we may interpret differently. We suspect that these differences are due to the typical replication problems in NLP (cf. Fokkens et al. (2013)). Overall, however, our model performs quite similarly to Pareti's, with our model scoring an overall F_1 of 72 % (vs. Pareti's 71 %) and a partial F_1 of 83 % (vs. 82 %).

GREEDY Next, we compare the GREEDY model to the CRF. We find its overall performance to be comparable to the CRF, confirming our expectations. While strict precision is statistically significantly lower for GREEDY (75 % vs. 79 %), strict recall is not significantly different (bot at 67 %). Considering partial matches, GREEDY has significantly higher recall (81 % vs. 77 %) but significantly lower precision (88 % vs. 90 %) than the CRF, with an overall comparable F_1. This result bolsters our hypothesis that the CRF learn only a small amount of useful sequence information. Although GREEDY ignores label sequences in training completely, it is able to compete with the CRF. Furthermore, the partial match result that GREEDY is a particularly good choice if the main interest is the approximate location of content spans in a document: The simpler model architecture makes it easier and more efficient to train and apply. The caveat is that GREEDY is particularly bad at locating mixed spans (as indicated by a precision of only 72 %): Quotation marks are generally good indicators for span boundaries and are often returned as false positives by the boundary detection models, so GREEDY tends to incorrectly pick them.

SEMIMARKOV Overall, the SEMIMARKOV model outperforms the CRF significantly in terms of strict recall (71 % vs. 67 %) and F_1 (75 % vs. 72 %), while precision remains unaffected (at 79 %). The model performs particularly well on indirect quotations (increasing F_1 by 5 points to 69 %), the most difficult category, where local context is insufficient. Meanwhile, on partial match, the SEMIMARKOV model has a comparable recall (80 vs. 77 %), but significantly lower precision (88 % vs. 90 %). The overall partial F_1 results are not significantly different. The improvement on the strict measures supports our intuition that better features help in particular in identifying the *exact* boundaries of quotations, a task that evidently profits from global information.

Model Combination The complementary strengths of the CRF and SEMIMARKOV (CRF detects direct quotations well, SEMIMARKOV indirect quotations) suggest a simple model combination algorithm based on the surface form of the spans: First take all *direct* and *mixed* spans predicted by the CRF; then add all *indirect* spans from the SEMIMARKOV model (except for those which would overlap). This result is our overall best model under strict evaluation, although it is not significantly better than the SEMIMARKOV model. Considering partial match, its results are essentially identical to the SEMIMARKOV model.

6 Analysis

We now proceed to a more detailed analysis of the performance of the three models (CRF, GREEDY, and SEMIMARKOV) and their differences in order to gain insights into the nature of the quotation detection task. In the interest of readability, we organize this section by major findings instead of the actual analyses that we have performed, and adduce for each finding all relevant analysis results.

Finding 1: Variation in length does not explain the differences in model performance. A possible intuition about our models it that the improvement of SEMIMARKOV over CRF is due to a better handling of longer quotations. However, this is not the case. Figure 3 shows the recall of the three models for quotations binned by lengths. The main patterns hold across all three models: Medium-length spans are the easiest to detect. Short spans are difficult to detect as they are often part of discontinuous content spans. Long spans are also

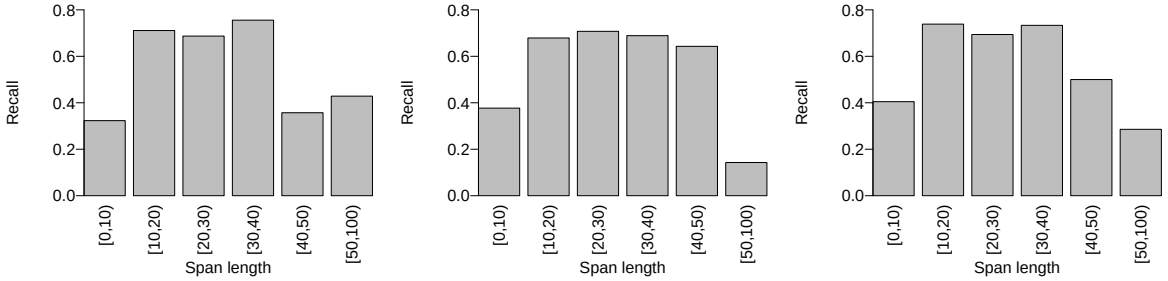


Figure 3: Strict recall by span length for CRF (left), GREEDY (center), and SEMIMARKOV model (right)

Category	Count		
	B	I	E
looking left	27	14	7
looking right	1	13	30
cue	11	10	7
other lexical	31	21	22
structural/syntactic	27	44	35
punctuation	31	25	36

Table 5: Categories of top positive and negative CRF features for begin (B), inside (I), and end (E)

difficult since any wrong intermediary decision can falsify the prediction. In fact, the CRF model is even the best model among the three for very long spans (which are rare). Those spans exceed the 55 and 75 token limits $\ell_{\max}$ of the GREEDY and SEMIMARKOV models. Intuitively, for the CRF, most spans are long: even spans which are short in comparison to other quotations are longer than the window within which the CRF operates. This is why span length does not have an influence.

Finding 2: Quotations are mostly defined by their immediate external context. A feature analysis of the CRF model reveals that many important features refer to material *outside* the quotation itself. For each label (B, I, E), we collect the 50 features with the highest positive and negative values, respectively. We first identify the subset of those features that looks look left or right. As the upper part of Table 5 shows, a substantial number of B (begin) features look to the left, and a number of E (end) features look to the right. Thus, these features do not look at the quotation itself, but at its immediate external context.

We next divide the features into four broad categories (cues, other lexical information, structural and syntactic features, and punctuation including quotation marks). The results in the lower part of Table 5 show that the begin and end classes rely on a range of categories, including lexical, cue and punctuation *outside* the quotation. The situation is different for inside tokens (I), where most features express structural and syntactic properties of the quotation such as the length of a sentence and its syntactic relation to a cue. Together, these observations suggest that one crucial piece of information about quotations is their lexical and orthographic context: the factors that mark a quotation as a quotation. Another crucial piece are internal structural properties of the quotation, while lexical properties of the quotation are not very important: which makes sense, since almost anything can be quoted.

The feature analysis is bolstered by an error analysis of the false negatives in the high-precision low-recall CRF. The first reason for false negatives is indeed the occurrence of infrequent cues which the cue model fails to identify (e.g., *read* or *acknowledge*). The second one is that the model does attempt to learn syntactic features, but that the structural features that can be learned by the CRF (such as C7, C10 or S4) can model only local windows of the quality of the quotation, but not its global quality. This leads us to our third finding.

Finding 3: Simple models cannot capture dependencies between begin and end boundaries well. Given the importance of cues, as evidenced by our Finding 2, we can ask whether the boundary of the quotation that is adjacent to its associated cue ("cue-near") is easier to identify than the other boundary ("cue-far") whose context is less informative. To assess this question, we evaluate the recall of individual boundary detection at the token level. For the CRF, "cue-far" boundaries of spans indeed tend to be more difficult to detect than "cue-near" ones. The results in Table 6 show that both the GREEDY and the CRF model show a marked asym-

	GREEDY	CRF	SEMIMARKOV
cue-near	76	74	76
cue-far	72	71	75

Table 6: Recall on boundaries by cue position

metry and perform considerably worse (3 % and 4 %, respectively) on the cue-far boundary. This asymmetry is considerably weaker for the SEMIMARKOV model, where both boundary types are recognized almost on par. The reason behind this finding is that neither the GREEDY model nor the CRF can condition the choice of the cue-far boundary on the cue-near boundary or on global properties of the quotation – the GREEDY model, because its choices are completely independent, and the CRF model, because its choices are largely independent due to the Markov assumption.

Finding 4: The SEMIMARKOV model benefits the most from its ability to handle global features about content spans. This leads us to our final finding about why the SEMIMARKOV model outperforms the CRF – whether it is the model architecture itself, or the new global features that it allows us to formulate. We perform an ablation study whose results are shown in Figure 4. We begin with only the token-level features on the begin, end, and interior tokens of the span, as introduced in Section 2, i.e., the features that the CRF has at its disposal. We find that this model performs on par with the CRF, thus the model architecture on its own does not help. We then incrementally add the feature templates containing count statistics of the internal tokens (Template G1 in Table 3) and advanced cue information (G2–G6). Both give the model incremental boosts. Adding syntactic coherence features (G7–G13) completes our full feature set and yields the best results.

Thus, the difference comes from features that describe global properties of the quotation. One of the most informative (negative) features is the conjunction from G6. It enforces the constraint that each content span is associated with a single cue. As in the CRF, the actual content of a content span does not play a large role. The only semantic features the model considers concern the presence of named entities within the span.

These observations are completed by analysis of the quotation spans that were correctly detected by the SEMIMARKOV model, but not the CRF (in terms of strict recall). We find a large amount of spans with highly ambiguous cue-near tokens such as *to* (10 % of the cases) *that* (16 %). We find that often the errors are also related to the frequency or location of cues. As an example, in the sentence

> [...] he has said [that when he was on the winning side in the 1960s, he knew that the tables might turn in the future]$_{\text{CONT}}$.[6]

the CRF model incorrectly splits the content span at the second cue candidate *knew*. This is, however, an embedded quotation that the model should ignore. In contrast, the SEMIMARKOV model makes use of the fact the tokens of the span depend on the same cue, and predicts the span correctly. For these tokens, the distinction between reported speech and factual descriptions is difficult. Arguably, it is the global features that help the model make its call.

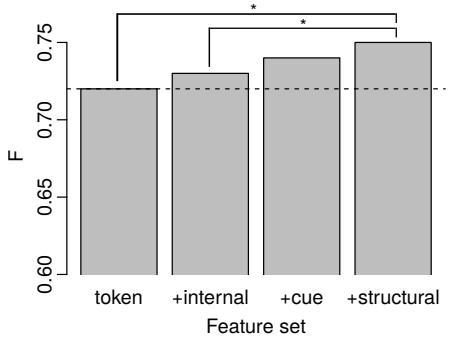


Figure 4: Strict F_1 for different feature sets in the SEMIMARKOV model. *: Difference statistically significant. Dashed line: CRF result.

7 Related Work

Quotation detection has been tackled with a number of different strategies. Pouliquen et al. (2007) use a small set of rules which has high precision but low recall on multilingual text. Krestel et al. (2008) also pursue a rule-based approach, focusing on the roles of cue verbs and syntactic markers. They evaluate on a small set of annotated WSJ documents and again report high precision but low recall. Pareti et al. (2013) develop the state-of-the-art sequence labeling approach discussed in this paper.

Our sampling approach builds on that of Zhang et al. (2015), who pursue a similar strategy for parsing, PoS tagging, and sentence segmentation. Similar semi-Markov model approaches have been used for other applications, e.g. by Yang and Cardie

[6] PARC, `wsj_2347`

(2012) and Klinger and Cimiano (2013) for sentiment analysis. They also predict spans by sampling, but they draw proposals based on the token or syntactic level. This is not suitable for quotation detection as we deal with much longer spans.

8 Conclusion

We have considered the task of quotation detection, starting from the hypothesis that linear-chain CRFs cannot take advantage of all available sequence information due to its Markov assumptions. Indeed, our analyses find that the features most important to recognize a quotation consider its direct context of orthographic evidence (such as quotation marks) and lexical evidence (such as cue words). A simple, greedy algorithm using non-sequential models of quotation boundaries rivals the CRF's performance. For further improvements, we introduce a semi-Markov model capable of taking into account *global* information about the complete span not available to a linear-chain CRF, such as the presence of cues on both sides of the quotation candidate. This leads to a significant improvement of 3 points F_1 over the state of the art.

On a more general level, we believe that quotation detection is interesting as a representative of tasks involving long sequences, where Markov assumptions become inappropriate. Other examples of such tasks include the identification of chemical compound names (Krallinger et al., 2015) and the detection of annotator rationales (Zaidan and Eisner, 2008). We have shown that a more expressive semi-Markov model which avoids these assumptions can improve performance. More expressive models however come with harder inference problems which are compounded when applied to long-sequence tasks. The informed sampling algorithm we have described performs such efficient inference for our semi-Markov quotation detection model.

Acknowledgments

This work was funded in part by the DFG through the Sonderforschungsbereich 732. We thank Silvia Pareti for kindly providing the PARC dataset as well as for much information helpful for replicating her results. Further thanks go to Anders Björkelund and Kyle Richardson for discussion and comments.

References

Apoorv Agarwal, Augusto Corvalan, Jacob Jensen, and Owen Rambow. 2012. Social network analysis of alice in wonderland. In *Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature*, pages 88–96, Montréal, Canada, June. Association for Computational Linguistics.

Ricardo Barandela, José Salvador Sánchez, Vicente García, and Edgar Rangel. 2003. Strategies for learning in class imbalance problems. *Pattern Recognition*, 36(3):849–851.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1–8, Philadelphia, PA.

David Elson, Nicholas Dames, and Kathleen McKeown. 2010. Extracting social networks from literary fiction. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 138–147, Uppsala, Sweden.

Manaal Faruqui and Sebastian Pado. 2012. Towards a model of formal and informal address in English. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 623–633, Avignon, France.

Antske Fokkens, Marieke van Erp, Marten Postma, Ted Pedersen, Piek Vossen, and Nuno Freire. 2013. Offspring from reproduction problems: What replication failure teaches us. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1691–1701, Sofia, Bulgaria.

Roman Klinger and Philipp Cimiano. 2013. Bidirectional inter-dependencies of subjective expressions and targets and their value for a joint model. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 848–854, Sofia, Bulgaria.

Martin Krallinger, Florian Leitner, Obdulia Rabal, Miguel Vazquez, Julen Oyarzabal, and Alfonso Valencia. 2015. CHEMDNER: The drugs and chemical names extraction challenge. *Journal of Cheminformatics*, 7(Suppl 1):S1.

Ralf Krestel, Sabine Bergler, and René Witte. 2008. Minding the source: Automatic tagging of reported speech in newspaper articles. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 2823–2828, Marrakech, Morocco.

Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz S. Kandola. 2002. The perceptron algorithm with uneven margins. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 379–386, Sydney, Australia.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of ACL System Demonstrations*, pages 55–60, Baltimore, MD.

Andrew K. McCallum, 2002. *MALLET: A Machine Learning for Language Toolkit*. User's manual.

Eric T. Nalisnick and Henry S. Baird. 2013. Character-to-character sentiment analysis in shakespeare's plays. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 479–483, Sofia, Bulgaria, August. Association for Computational Linguistics.

Eric W. Noreen. 1989. *Computer intensive methods for hypothesis testing: An introduction*. Wiley, New York.

Silvia Pareti, Tim O'Keefe, Ioannis Konstas, James R. Curran, and Irena Koprinska. 2013. Automatically detecting and attributing indirect quotations. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 989–999, Seattle, WA.

Silvia Pareti. 2015. *Attribution: A Computational Approach*. Ph.D. thesis, University of Edinburgh.

Bruno Pouliquen, Ralf Steinberger, and Clive Best. 2007. Automatic detection of quotations in multilingual news. In *Proceedings of Recent Advances in Natural Language Processing*, pages 487–492, Borovets, Bulgaria.

Sunita Sarawagi and William W. Cohen. 2004. Semi-markov conditional random fields for information extraction. In *Proceedings of Advances in Neural Information Processing Systems*, pages 1185–1192, Vancouver, BC.

Ralph Weischedel and Ada Brunstein. 2005. BBN pronoun coreference and entity type corpus. Linguistic Data Consortium, Philadelphia.

Michael Wick, Khashayar Rohanimanesh, Kedar Bellare, Aron Culotta, and Andrew McCallum. 2011. Samplerank: Training factor graphs with atomic gradients. In *Proceedings of the 28th International Conference on Machine Learning*, pages 777–784, Bellevue, WA.

Bishan Yang and Claire Cardie. 2012. Extracting opinion expressions with semi-Markov conditional random fields. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1335–1345, Jeju Island, South Korea.

Omar Zaidan and Jason Eisner. 2008. Modeling annotators: A generative approach to learning from annotator rationales. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 31–40, Honolulu, HI.

Yuan Zhang, Chengtao Li, Regina Barzilay, and Kareem Darwish. 2015. Randomized greedy inference for joint segmentation, POS tagging and dependency parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 42–52, Denver, CO.

Speech Act Modeling of Written Asynchronous Conversations with Task-Specific Embeddings and Conditional Structured Models

Shafiq Joty and **Enamul Hoque**
ALT Research Group
Qatar Computing Research Institute — HBKU, Qatar Foundation
{sjoty, mprince}@qf.org.qa

Abstract

This paper addresses the problem of speech act recognition in written asynchronous conversations (e.g., fora, emails). We propose a class of conditional structured models defined over arbitrary graph structures to capture the conversational dependencies between sentences. Our models use sentence representations encoded by a long short term memory (LSTM) recurrent neural model. Empirical evaluation shows the effectiveness of our approach over existing ones: (*i*) LSTMs provide better task-specific representations, and (*ii*) the global joint model improves over local models.

1 Introduction

Asynchronous conversations, where participants communicate with each other at different times (e.g., fora, emails), have become very common for discussing events, issues, queries and life experiences. In doing so, participants interact with each other in complex ways, performing certain communicative acts like asking questions, requesting information or suggesting something. These are called **speech acts** (Austin, 1962).

For example, consider the excerpt of a forum conversation from our corpus in Figure 1. The participant who posted the first comment C_1, describes his situation by the first two sentences and then asks a *question* in the third sentence. Other participants respond to the query by *suggesting* something or *asking* for clarification. In this process, the participants get into a conversation by taking turns, each of which consists of one or more speech acts. The two-part structures across posts like 'question-answer' and 'request-grant' are called **adjacency pairs** (Schegloff, 1968).

C_1: My son wish to do his bachelor degree in Mechanical Engineering in an affordable Canadian university.
Human: st, Local: st, Global: st

The info. available in the net and the people who wish to offer services are too many and some are misleading.
Human: st, Local: st, Global: st

The preliminary preparations,eligibility,the require funds etc., are some of the issues which I wish to know from any panel members of this forum .. (truncated)
Human: ques, Local: st, Global: st

C_3 (truncated)...take a list of canadian universities and then create a table and insert all the relevant information by reading each and every program info on the web.
Human: sug, Local: sug, Global: sug

Without doing a research my advice would be to apply to UVIC .. for the following reasons .. (truncated)
Human: sug, Local: sug, Global: sug

UBC is good too... but it is expensive particularly for international students due to tuition .. (truncated)
Human: sug, Local: sug, Global: sug

most of them accept on-line or email application.
Human: st, Local: st, Global: st

Good luck !!
Human: pol, Local: pol, Global: pol

C_4 snakyy21: UVIC is a short form of? I have already started researching for my brother and found "College of North Atlantic" and .. (truncated)
Human: ques, Local: st, Global: ques

but not sure about the reputation..
Human: st, Local: res, Global: st

C_5 thank you for sharing useful tips will follow your advise.
Human: pol, Local: pol, Global: pol

Figure 1: Example conversation with **Human** annotations and automatic predictions by a **Local** classifier and a **Global** classifier. The labels **st**, **ques**, **sug**, and **pol** refers to *Statement*, *Question*, *Suggestion*, and *Polite* speech acts, respectively.

Identification of speech acts is an important step towards deep conversation analysis in these media (Bangalore et al., 2006), and has been shown to be useful in many downstream applications including summarization (McKeown et al., 2007) and question answering (Hong and Davison, 2009).

Previous attempts to automatic (sentence-level)

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1746–1756,
Berlin, Germany, August 7-12, 2016.

speech act recognition in asynchronous conversation (Qadir and Riloff, 2011; Jeong et al., 2009; Tavafi et al., 2013; Oya and Carenini, 2014) suffer from at least one of the two major flaws.

Firstly, they use bag-of-word (BOW) *representation* (e.g., unigram, bigram) to encode lexical information in a sentence. However, consider the *suggestion* sentences in the example. Arguably, a model needs to consider the structure (e.g., word order) and the compositionality of phrases to identify the right speech act. Furthermore, BOW representation could be quite sparse and may not generalize well when used in classification models.

Secondly, existing approaches mostly disregard conversational dependencies between sentences. For instance, consider the example again, where we tag the sentences with the human annotations ('Human') and with the predictions of a local ('Local') classifier that considers word order for sentence representation but classifies each sentence separately. Prediction errors are underlined and highlighted in red. Notice the first and second sentences of comment 4, which are tagged mistakenly as *statement* and *response*, respectively, by our best local classifier. We hypothesize that some of the errors made by the local classifier could be corrected by employing a global joint model that performs a collective classification taking into account the conversational dependencies between sentences (e.g., adjacency relations).

However, unlike synchronous conversations (e.g., phone, meeting), modeling conversational dependencies between sentences in asynchronous conversation is challenging, especially in those where explicit thread structure (reply-to relations) is missing, which is also our case. The conversational flow often lacks sequential dependencies in its temporal order. For example, if we arrange the sentences as they arrive in the conversation, it becomes hard to capture any dependency between the act types because the two components of the adjacency pairs can be far apart in the sequence. This leaves us with one open research question: how to model the dependencies between sentences in a single comment and between sentences across different comments? In this paper, we attempt to address this question by designing and experimenting with conditional structured models over arbitrary graph structure of the conversation.

More concretely, we make the following contributions. Firstly, we propose to use Recurrent Neural Network (RNN) with Long Short Term Memory (LSTM) hidden layer to perform composition of phrases and to represent sentences using distributed condensed vectors (i.e., embeddings). We experiment with both unidirectional and bidirectional RNNs. Secondly, we propose conditional structured models in the form of pairwise Conditional Random Field (Murphy, 2012) over arbitrary conversational structures. We experiment with different variations of this model to capture different types of interactions between sentences inside the comments and across the comments. These models use the LSTM encoded vectors as feature vectors for performing the classification task jointly. As a secondary contribution, we also present and release a forum dataset annotated with a standard speech act tagset.

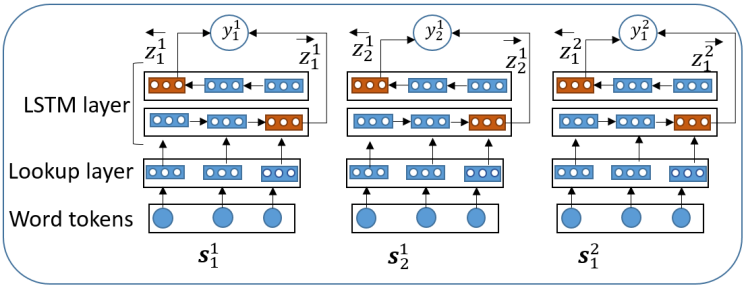
We train our models on different settings using synchronous and asynchronous corpora, and evaluate on two forum datasets. Our main findings are: (*i*) LSTM RNNs provide better representation than BOW; (*ii*) Bidirectional LSTMs, which encode a sentence using two vectors provide better representation than the unidirectional ones; and (*iii*) Global joint models improve over local models given that it considers the right graph structure. The source code and the new dataset are available at `http://alt.qcri.org/tools/speech-act/`

2 Our Approach

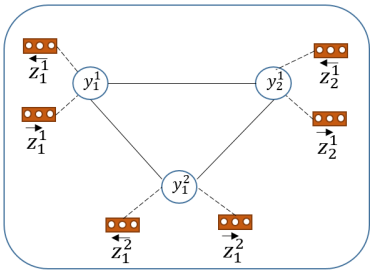
Let $\mathbf{s}_m^n$ denote the m-th sentence of comment n in a conversation. Our framework works in two steps as demonstrated in Figure 2. First, we use a recurrent neural network (RNN) to compose sentence representations semantically from their words and to represent them with distributed condensed vectors $\mathbf{z_m^n}$, i.e., sentence embeddings (Figure 2a). In the second step, a multivariate (graphical) model, which operates on the sentence embeddings, captures conversational dependencies between sentences in the conversation (Figure 2b). In the following, we describe the two steps in detail.

2.1 Sentence Representation

One of our main hypotheses is that a sentence representation method should consider the word order of the sentence. To this end, we use an LSTM RNN (Hochreiter and Schmidhuber, 1997) to encode a sentence into a vector by processing its words sequentially, at each time step combining



(a) Bidirectional LSTM-based RNN model



(b) Fully connected CRF model

Figure 2: Our two-step framework for speech act recognition in asynchronous conversation: (a) a bidirectional LSTM encodes each sentence $\mathbf{s}_m^n$ into a condensed vector $\mathbf{z_m^n}$ and classifies them separately; (b) a fully-connected CRF that takes the encoded vectors as input and performs joint learning and inference.

the current input with the previous hidden state. Figure 4b demonstrates the process for three sentences. Each word in the vocabulary V is represented by a D dimensional vector in a shared lookup table $L \in \mathbb{R}^{|V| \times D}$. L is considered a model parameter to be learned. We can initialize L randomly or by pretrained word embedding vectors like word2vec (Mikolov et al., 2013a).

Given an input sentence $\mathbf{s} = (w_1, \cdots, w_T)$, we first transform it into a feature sequence by mapping each token $w_t \in \mathbf{s}$ to an index in L. The lookup layer then creates an input vector $\mathbf{x_t} \in \mathbb{R}^D$ for each token w_t. The input vectors are then passed to the LSTM recurrent layer, which computes a compositional representation $\overrightarrow{\mathbf{h}}_t$ at every time step t by performing nonlinear transformations of the current input $\mathbf{x}_t$ and the output of the previous time step $\overrightarrow{\mathbf{h}}_{t-1}$. Specifically, the recurrent layer in a LSTM RNN is constituted with hidden units called *memory blocks*. A memory block is composed of four elements: (*i*) a memory cell c (a neuron) with a self-connection, (*ii*) an input gate i to control the flow of input signal into the neuron, (*iii*) an output gate o to control the effect of the neuron activation on other neurons, and (*iv*) a forget gate f to allow the neuron to adaptively reset its current state through the self-connection. The following sequence of equations describe how the memory blocks are updated at every time step t:

$$\mathbf{i}_t = \text{sigh}(U_i \mathbf{h}_{t-1} + V_i \mathbf{x}_t + \mathbf{b}_i) \quad (1)$$

$$\mathbf{f}_t = \text{sigh}(U_f \mathbf{h}_{t-1} + V_f \mathbf{x}_t + \mathbf{b}_f) \quad (2)$$

$$\mathbf{c}_t = \mathbf{i}_t \odot \tanh(U_c \mathbf{h}_{t-1} + V_c \mathbf{x}_t) + \mathbf{f}_t \odot \mathbf{c}_{t-1} \quad (3)$$

$$\mathbf{o}_t = \text{sigh}(U_o \mathbf{h}_{t-1} + V_o \mathbf{x}_t + \mathbf{b}_o) \quad (4)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \quad (5)$$

where U_k and V_k are the weight matrices between two consecutive hidden layers, and between the input and the hidden layers, respectively, which are associated with gate k (input, output, forget and cell); and and $\mathbf{b}_k$ is the corresponding bias vector. The symbols sigh and tanh denote hard sigmoid and hard tan, respectively, and the symbol $\odot$ denotes a element-wise product of two vectors.

LSTM by means of its specifically designed gates (as opposed to simple RNNs) is capable of capturing long range dependencies. We can interpret $\mathbf{h}_t$ as an intermediate representation summarizing the past. The output of the last time step $\overrightarrow{\mathbf{h}}_T = \mathbf{z}$ thus represents the sentence, which can be fed to the output layer of the neural network (Fig. 4b) or to other models (e.g, a fully-connected CRF in Fig. 2b) for classification. The output layer of our LSTM-RNN uses a `softmax` for multiclass classification. Formally, the probability of k-th class for classification into K classes is

$$p(y = k|\mathbf{s}, \theta) = \frac{exp\ (\mathbf{w}_k^T \mathbf{z})}{\sum_{k=1}^{K} exp\ (\mathbf{w}_k^T \mathbf{z})} \quad (6)$$

where $\mathbf{w}$ are the output layer weights.

Bidirectionality The RNN described above encodes information that it gets only from the past. However, information from the future could also be crucial for recognizing speech acts. This is specially true for longer sentences, where a unidirectional LSTM can be limited in encoding the necessary information into a single vector. Bidirectional RNNs (Schuster and Paliwal, 1997) capture dependencies from both directions, thus provide two different views of the same sentence. This amounts to having a backward counterpart for each of the equations from 1 to 5. For classification, we use the concatenated vector $[\overrightarrow{\mathbf{h}}_T, \overleftarrow{\mathbf{h}}_T]$,

where $\overrightarrow{\mathbf{h}_T}$ and $\overleftarrow{\mathbf{h}_T}$ are the encoded vectors summarizing the past and the future, respectively.

2.2 Conditional Structured Model

Given the vector representation of the sentences in an asynchronous conversation, we explore two different approaches to learn classification functions. The first and the traditional approach is to learn a local classifier ignoring the structure in the output and to use it for predicting the label of each sentence separately. This is the approach we took above when we fed the output layer of the LSTM RNN with the sentence-level embeddings. However, this approach does not model the conversational dependency (e.g., *adjacency* relations between question-answer and request-accept pairs).

The second approach, which we adopt in this paper, is to model the dependencies between the output variables (labels) while learning the classification functions jointly by optimizing a global performance criterion. We represent each conversation by a graph $G{=}(V, E)$. Each node $i{\in}V$ is associated with an input vector $\mathbf{z}_i = \mathbf{z}_m^n$, representing the features of the sentence $\mathbf{s}_m^n$, and an output variable $y_i{\in}\{1, 2, \cdots, K\}$, representing the class label. Similarly, each edge $(i, j){\in}E$ is associated with an input feature vector $\phi(\mathbf{z}_i, \mathbf{z}_j)$, derived from the node-level features, and an output variable $y_{i,j}{\in}\{1, 2, \cdots, L\}$, representing the state transitions for the pair of nodes. We define the following conditional joint distribution:

$$p(\mathbf{y}|\mathbf{v}, \mathbf{w}, \mathbf{z}) = \frac{1}{Z(\mathbf{v}, \mathbf{w}, \mathbf{z})} \prod_{i \in V} \psi_n(y_i|\mathbf{z}, \mathbf{v}) \prod_{(i,j) \in E} \psi_e(y_{i,j}|\mathbf{z}, \mathbf{w}) \quad (7)$$

where ψ_n and ψ_e are node and the edge *factors*, and $Z(.)$ is the global normalization constant that ensures a valid probability distribution. We use a log-linear representation for the factors:

$$\psi_n(y_i|\mathbf{z}, \mathbf{v}) = \exp(\mathbf{v}^T \phi(y_i, \mathbf{z})) \quad (8)$$

$$\psi_e(y_{i,j}|\mathbf{z}, \mathbf{w}) = \exp(\mathbf{w}^T \phi(y_{i,j}, \mathbf{z})) \quad (9)$$

where $\phi(.)$ is a feature vector derived from the inputs and the labels. This model is essentially a pairwise conditional random field or PCRF (Murphy, 2012). The global normalization allows CRFs to surmount the so-called *label bias* problem (Lafferty et al., 2001), allowing them to take long-range interactions into account. The log likelihood for one data point $(\mathbf{z}, \mathbf{y})$ (i.e., a conversation) is:

$$f(\theta) = \sum_{i \in V} \mathbf{v}^T \phi(y_i, \mathbf{z}) + \sum_{(i,j) \in E} \mathbf{w}^T \phi(y_{i,j}, \mathbf{z}) - \log Z(\mathbf{v}, \mathbf{w}, \mathbf{z}) \quad (10)$$

This objective is convex, so we can use gradient-based methods to find the global optimum. The gradients have the following form:

$$f'(\mathbf{v}) = \sum_{i \in V} \phi(y_i, \mathbf{z}) - \mathbb{E}[\phi(y_i, \mathbf{z})] \quad (11)$$

$$f'(\mathbf{w}) = \sum_{(i,j) \in E} \phi(y_{i,j}, \mathbf{z}) - \mathbb{E}[\phi(y_{i,j}, \mathbf{z})] \quad (12)$$

where $\mathbb{E}[\phi(.)]$ denote the expected feature vector.

Training and Inference Traditionally, CRFs have been trained using offline methods like limited-memory BFGS (Murphy, 2012). Online training of CRFs using stochastic gradient descent (SGD) was proposed by Vishwanathan et al. (2006). Since RNNs are trained with online methods, to compare our two methods, we use SGD to train our CRFs. Algorithm 1 in the Appendix gives a pseudocode of the training procedure.

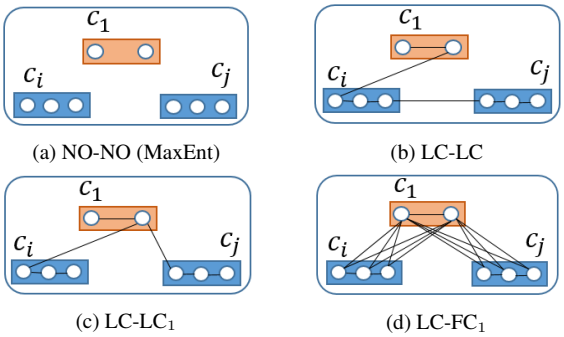
We use Belief Propagation or BP (Pearl, 1988) for inference in our graphical models. BP is guaranteed to converge to an exact solution if the graph is a tree. However, exact inference is intractable for graphs with loops. Despite this, it has been advocated by Pearl (1988) to use BP in loopy graphs as an approximation; see also (Murphy, 2012), page 768. The algorithm is then called "loopy" BP, or LBP. Although LBP gives approximate solutions for general graphs, it often works well in practice (Murphy et al., 1999), outperforming other methods such as mean field (Weiss, 2001).

Variations of Graph Structures One of the main advantages of our pairwise CRF is that we can define this model over arbitrary graph structures, which allows us to capture conversational dependencies at various levels. We distinguish between two types of dependencies: (*i*) *intra-comment*, which defines how the labels of the sentences in a comment are connected; and (*ii*) *across-comment*, which defines how the labels of the sentences across comments are connected.

Table 1 summarizes the connection types that we have explored in our models. Each configuration of intra- and across- connections yields a different pairwise CRF model. Figure 3 shows four such CRFs with three comments — C_1 being the first comment, and C_i and C_j being two other comments in the conversation.

Tag	Connection type	Applicable to
NO	No connection between nodes	intra & across
LC	Linear chain connection	intra & across
FC	Fully connected	intra & across
FC_1	Fully connected with first comment only	across
LC_1	Linear chain with first comment only	across

Table 1: Connection types in CRF models.



(a) NO-NO (MaxEnt) (b) LC-LC

(c) $LC\text{-}LC_1$ (d) $LC\text{-}FC_1$

Figure 3: CRFs over different graph structures.

Figure 3a shows the structure for **NO-NO** configuration, where there is no link between nodes of both intra- and across- comments. In this setting, the CRF model is equivalent to MaxEnt. Figure 3b shows the structure for **LC-LC**, where there are linear chain relations between nodes of both intra- and across- comments. The linear chain across comments refers to the structure, where the last sentence of each comment is connected to the first sentence of the comment that comes next in the temporal order (i.e., posting time). Figures 3c shows the CRF for **LC-LC**$_1$, where sentences inside a comment have linear chain connections, and the last sentence of the first comment is connected to the first sentence of the other comments. Similarly, Figure 3d shows the graph structure for **LC-FC**$_1$ configuration, where sentences inside comments have linear chain connections, and sentences of the first comment are fully connected with the sentences of the other comments.

3 Corpora

There exist large corpora of utterances annotated with speech acts in synchronous spoken domains, e.g., Switchboard-DAMSL or SWBD (Jurafsky et al., 1997) and Meeting Recorder Dialog Act or MRDA (Dhillon et al., 2004). However, such large corpus does not exist in asynchronous domains. Some prior work (Cohen et al., 2004; Ravi and Kim, 2007; Feng et al., 2006; Bhatia et al., 2014) tackles the task at the comment level, and uses task-specific tagsets. In contrast, in this work we are interested in identifying speech acts at the sentence level, and also using a standard tagset like the ones defined in SWBD and MRDA.

	TA	BC3
Total number of conv.	200	39
Avg. nb of comments per conv.	4.02	6.54
Avg. nb of sentences per conv.	18.56	34.15
Avg. nb of words per sentence	14.90	12.61

Table 2: Statistics about TA and BC3 corpora.

Tag	Description	TA	BC3	MRDA
SU	Suggestion	7.71%	5.48%	5.97%
R	Response	2.4%	3.75%	15.63%
Q	Question	14.71%	8.41%	8.62%
P	Polite	9.57%	8.63%	3.77%
ST	Statement	65.62%	73.72%	66.00%

Table 3: Distribution of speech acts in our corpora.

More recent studies attempt to solve the task at the sentence level. Jeong et al. (2009) first created a dataset of TripAdvisor (TA) forum conversations annotated with the standard 12 act types defined in MRDA. They also remapped the BC3 email corpus (Ulrich et al., 2008) according to this tagset. Table 10 in the Appendix presents the tags and their relative frequency in the two datasets. Subsequent studies (Joty et al., 2011; Tavafi et al., 2013; Oya and Carenini, 2014) use these datasets. We also use these datasets in our work. Table 2 shows some basic statistics about these datasets. On average, BC3 conversations are longer than TA in both number of comments and number of sentences.

Since these datasets are relatively small in size, we group the 12 acts into 5 coarser classes to learn a reasonable classifier.[1] More specifically, all the question types are grouped into one general class *Question*, all response types into *Response*, and appreciation and polite mechanisms into *Polite* class. Also since deep neural models like LSTM RNNs require a lot of training data, we also utilize the MRDA meeting corpus. Table 3 shows the label distribution of the resultant datasets. Statement is the most dominant class, followed by Question, Polite and Suggestion.

QC3 Conversational Corpus Since both TA and BC3 are quite small to make a general comment about model performance in asynchronous

[1] Some prior work (Tavafi et al., 2013; Oya and Carenini, 2014) also took the same approach.

Speech Act	Distribution	κ
Suggestion	17.38%	0.86
Response	5.24%	0.43
Question	12.59%	0.87
Polite	6.13%	0.75
Statement	58.66%	0.78

Table 4: Corpus statistics for QC3.

Corpora	Type	Train	Dev.	Test
QC3	asynchronous	1252	157	156
TA	asynchronous	2968	372	371
BC3	asynchronous	1065	34	133
MRDA	synchronous	50865	8366	10492
Total	asyn. + sync.	56150	8929	11152

Table 5: Number of sentences in train, development and test sets for different datasets.

conversation, we have created a new dataset called Qatar Computing Conversational Corpus or QC3.

We selected 50 conversations from a popular community question answering site named Qatar Living[2] for our annotation. We used 3 conversations for our pilot study and used the remaining 47 for the actual study. The resultant corpus on average contains 13.32 comments and 33.28 sentences per conversation, and 19.78 words per sentence.

Two native speakers of English annotated each conversation using a web-based annotation framework. They were asked to annotate each sentence with the most appropriate speech act tag from the list of 5 speech act types. Since this task is not always obvious, we gave them detailed annotation guidelines with real examples. We use Cohens Kappa κ to measure the agreement between the annotators. Table 4 presents the distribution of the speech acts and their respective κ values.After Statement, Suggestion is the most frequent class, followed by Question and Polite. The κ varies from 0.43 (for Response) to 0.87 (for Question).

Finally, in order to create a consolidated dataset, we collected the disagreements and employed a third annotator to resolve those cases.

4 Experiments and Analysis

In this section we present our experimental settings, results and analysis. We evaluate our models on the two forum corpora QC3 and TA. For performance comparison, we use both *accuracy* and *macro-averaged* F_1 score. Accuracy gives the overall performance of a classifier but could be biased to most populated ones. Macro-averaged F_1 weights equally every class and is not influenced by class imbalance. Statistical significance tests are done using an *approximate randomization* test based on the accuracy.[3] We used SIGF V.2 (Padó, 2006) with 10,000 iterations.

[2]http://www.qatarliving.com/

[3]Significance tests operate on individual instances rather than individual classes; thus not applicable for macro F_1.

Because of the noise and informal nature of conversational texts, we performed a series of preprocessing steps. We normalize all characters to their lower-cased forms, truncate elongations to two characters, spell out every digit and URL. We further tokenized the texts using the CMU TweetNLP tool (Gimpel et al., 2011).

In the following, we first demonstrate the effectiveness of LSTM RNNs for learning representations of sentences automatically to identify their speech acts. Then in subsection 4.2, we show the usefulness of pairwise CRFs for capturing conversational dependencies in speech act recognition.

4.1 Effectiveness of LSTM RNNs

To show the effectiveness of LSTMs for learning sentence representations, we split each of our asynchronous corpora randomly into 70% *sentences* for training, 10% for development, and 20% for testing. For MRDA, we use the same train-test-dev split as Jeong et al. (2009). Table 5 summarizes the resultant datasets.

We compare the performance of LSTMs with that of MaxEnt (**ME**) and Multi-layer Perceptron (**MLP**) with one hidden layer.[4] Both ME and MLP were fed with the bag-of-word (**BOW**) representations of the sentence, i.e., vectors containing binary values indicating the presence or absence of a word in the training set vocabulary.

We train the models by optimizing the cross entropy using the gradient-based online learning algorithm ADAM (Kingma and Ba, 2014).[5] The learning rate and other parameters were set to the values as suggested by the authors. To avoid overfitting, we use dropout (Srivastava et al., 2014) of hidden units and early stopping based on the loss on the development set.[6] Maximum number of epochs was set to 25 for RNNs and 100 for ME and MLP. We experimented with $\{0.0, 0.2, 0.4\}$

[4]More hidden layers worsened the performance.

[5]Other algorithms (SGD, Adagrad) gave similar results.

[6]l_1 and l_2 regularization on weights did not work well.

dropout rates, $\{16, 32, 64\}$ minibatch sizes, and $\{100, 150, 200\}$ hidden layer units in MLP and in LSTMs. The vocabulary (V) in LSTMs was limited to the most frequent $P\%$ ($P \in \{85, 90, 95\}$) words in the training corpus. We initialize the word vectors in the loop-up table L in one of two ways: (*i*) by sampling randomly from the small uniform distribution $\mathcal{U}(-0.05, 0.05)$, and (*ii*) by using pretrained 300 dimensional Google word embeddings from Mikolov et al. (2013b). The dimension for random initialization was set to 128.

We experimented with four LSTM variations: (*i*) U-LSTM$_r$, referring to unidirectional with random initialization; (*ii*) U-LSTM$_p$, referring to unidirectional with pretrained initialization; (*iii*) B-LSTM$_r$, referring to bidirectional with random initialization; and (*iv*) B-LSTM$_p$, referring to bidirectional with pretrained initialization.

Table 6 shows the results for different models for the data splits in Table 5. The first two rows show the best results reported so far on the MRDA corpus from (Jeong et al., 2009) for classifying into 12 act types. The first row shows the results of the model that uses *n-grams* and the second row shows the results using *all* the features including speaker, part-of-speech, and dependency structure. Our LSTM RNNs and their n-gram model therefore use the same word sequence information. To compare our results with the state of the art, we ran our models on MRDA for both 5-class and 12-class classification tasks. The results are shown at the right most part of Table 6.

Notice that all of our LSTMs achieve state of the art results and B-LSTM$_p$ achieves even significantly better with 99% confidence level. This is remarkable since our LSTMs learn the sentence representation automatically from the word sequence and do not use any hand-engineered features.

Now consider the asynchronous domains QC3 and TA, where we show the results of our models based on 5-fold cross validation, in addition to the random (20%) testset. The 5-fold setting allows us to get more general performance of the models on a particular corpus. The comparison between our LSTMs shows that: (*i*) pretrained Google vectors provide better initialization to LSTMs than the random ones; (*ii*) bidirectional LSTMs outperform their unidirectional counterparts. When we compare these results with those of our baselines, the results are disappointing; the ME and MLP using BOW outperform LSTMs by a good margin.

	SU	R	Q	P	ST
SU	34	0	1	0	27
R	0	4	0	2	12
Q	0	0	64	0	13
P	0	0	1	35	6
ST	8	1	3	4	311

(a) B-LSTM$_p$

	SU	R	Q	P	ST
SU	21	1	1	0	39
R	0	6	0	1	11
Q	0	0	63	0	14
P	0	0	1	32	9
ST	8	2	0	2	316

(b) MLP

Figure 4: Confusion matrices for (a) B-LSTM$_p$ and (b) MLP on the testsets of QC3 and TA.

However, this is not surprising since deep neural networks like LSTMs have a lot of parameters, for which they require a lot of data to learn from.

To validate our claim, we create another *training* setting CAT by merging the training and development sets of the four corpora in Table 5 (see the Train and Dev. columns in the last row); the testset for each dataset however remains the same. Table 7 shows the results of the baselines and the B-LSTM$_p$ on the QC3 and TA testsets. In both datasets, B-LSTM$_p$ outperforms ME and MLP significantly. When we compare these results with those in Table 6, we notice that B-LSTM$_p$, by virtue of its distributed and condensed representation, generalizes well across different domains. In contrast, ME and MLP, because of their BOW representation, suffer from data diversity of different domains. These results also confirm that B-LSTM$_p$ gives better sentence representation than BOW, when it is given enough data.

To analyze further the cases where B-LSTM$_p$ makes a difference, Figure 4 shows the corresponding confusion matrices for B-LSTM$_p$ and MLP on the concatenated testsets of QC3 and TA. It is noticeable that B-LSTM$_p$ is less affected by class imbalance and it can detect more *suggestions* than MLP. This indicates that LSTM RNNs can model the grammar of the sentence when composing the words into phrases sequentially.

4.2 Effectiveness of CRFs

To demonstrate the effectiveness of CRFs for capturing inter-sentence dependencies in an asynchronous conversation, we create another dataset setting called CON, in which the random splits are done at the *conversation* (as opposed to sentence) level for the asynchronous corpora. This is required because our CRF models perform joint learning and inference based on a full conversation. As presented in Table 8, this setting contains 197 and 24 conversations for training and devel-

	QC3		TA		MRDA	
	Testset	5 folds	Testset	5 folds	5 classes	12 classes
Jeong et al. (ng)	-	-	-	-	-	57.53 (83.30)
Jeong et al. (All)	-	-	-	-	-	59.04 (83.49)
ME	55.12 (**75.64**)	50.23 (71.37)	61.4 (85.44)	59.23 (84.85)	65.25 (83.95)	57.79 (82.84)
MLP	**61.30** (74.36)	**54.57 (71.63)**	**68.17 (85.98)**	**62.41 (85.02)**	68.12 (84.24)	58.19 (83.24)
U-LSTM$_r$	51.57 (73.55)	48.64 (65.94)	56.54 (83.24)	56.39 (83.83)	71.29 (85.38)	58.72 (83.34)
U-LSTM$_p$	49.41 (70.97)	50.26 (65.62)	63.12(83.78)	59.10 (83.13)	72.32 (85.19)	59.05 (84.06)
B-LSTM$_r$	50.75 (72.26)	48.41 (66.19)	58.88 (82.97)	56.23 (83.34)	71.69 (85.62)	58.33 (83.49)
B-LSTM$_p$	53.22 (71.61)	51.59 (68.50)	60.73 (82.97)	59.68 (84.07)	72.02 (85.33)	**60.12 (84.46*)**

Table 6: Macro-averaged F_1 and raw accuracy (in parenthesis) for baselines and LSTM variants on the testset and 5-fold splits of different corpora. For MRDA, we use the same train-test-dev split as (Jeong et al., 2009). Accuracy significantly superior to state-of-the-art is marked with *.

	QC3 (Testset)	TA (Testset)
ME	50.64 (71.15)	72.49 (84.10)
MLP	58.60 (74.36)	73.07 (86.29)
B-LSTM$_p$	**66.40 (80.65*)**	**73.14 (87.01*)**

Table 7: Results on CAT dataset.

	Train	Dev	Test
QC3	38 (1332)	4 (111)	5 (122)
TA	160 (2957)	20 (310)	20 (444)
Total	197 (4289)	24 (421)	25 (566)

Table 8: Setting for CON dataset. The numbers inside parentheses indicate the number of sentences.

	QC3	TA
ME$_b$	56.67 (67.21)	63.29 (84.23)
B-LSTM$_p$	65.15 (77.87)	66.93 (85.13)
ME$_e$	59.94 (77.05)	59.55 (85.14)
CRF (LC-NO)	62.20 (77.87)	60.30 (85.81)
CRF (LC-LC)	62.35 (78.69)	60.30 (85.81)
CRF (LC-LC$_1$)	**65.94 (80.33*)**	61.58 (86.54)
CRF (LC-FC$_1$)	61.18 (77.87)	60.00 (85.36)
CRF (FC-FC)	64.54 (79.51*)	**61.64 (86.81*)**

Table 9: Results of CRFs on CON dataset.

opment, respectively.[7] The testsets contain 5 and 20 conversations for QC3 and TA, respectively.

As baselines, we use three models: (*i*) **ME**$_b$, a MaxEnt using BOW representation; (*ii*) **B-LSTM**$_p$, which is now trained on the concatenated set of sentences from MRDA and CON training sets; and (*iii*) **ME**$_e$, a MaxEnt using sentence embeddings extracted from the B-LSTM$_p$, i.e., the sentence embeddings are used as feature vectors.

We experiment with the CRF variants in Table 1. The CRFs are trained on the CON training set using the sentence embeddings that are extracted by applying the B-LSTM$_p$ model, as was done with ME$_e$. Table 9 shows our results. We notice that CRFs generally outperform MEs in accuracy. This indicates that there are conversational dependencies between the sentences in a conversation.

When we compare between CRF variants, we notice that the model that does not consider any link across comments perform the worst; see CRF (LC-NO). A simple linear chain connection between sentences in their temporal order does not improve much (CRF (LC-LC)), which indicates that the widely used linear chain CRF (Lafferty et al., 2001) is not the most appropriate model for capturing conversational dependencies in these conversations. The CRF (LC-LC$_1$) is one of the best performing models and perform significantly (with 99% confidence) better than B-LSTM$_p$.[8] This model considers linear chain connections between sentences inside comments and only to the first comment. Note that both QC3 and TA are forum sites, where participants in a conversation interact mostly with the person who posts the first comment asking for some information. This is interesting that our model can capture this aspect.

Another interesting observation is that when we change the above model to consider relations with every sentence in the first comment (CRF (LC-FC$_1$)), this degrades the performance. This could be due to the fact that the information seeking person first explains her situation, and then asks for the information. Others tend to respond to the requested information rather than to her situation. The CRF (FC-FC) also yields as good results as CRF (LC-LC$_1$). This could be attributed to the robustness of the fully-connected CRF, which learns

[7] We use the concatenated sets as train and dev. sets.

[8] Significance was computed on the concatenated testset.

from all possible relations.

To see some real examples in which CRF by means of its global learning and inference makes a difference, let us consider the example in Figure 1 again. We notice that the two sentences in comment C_4 were mistakenly identified as Statement and Response, respectively, by the B-LSTM$_p$ local model. However, by considering these two sentences together with others in the conversation, the global CRF (FC-FC) model could correct them.

5 Related Work

Three lines of research are related to our work: (*i*) semantic compositionality with LSTM RNNs, (*ii*) conditional structured models, and (*iii*) speech act recognition in asynchronous conversations.

LSTM RNNs for composition Li et al. (2015) compare recurrent neural models with recursive (syntax-based) models for several NLP tasks and conclude that recurrent models perform on par with the recursive for most tasks (or even better). For example, recurrent models outperform recursive on sentence level sentiment classification. This finding motivated us to use recurrent models rather than recursive. The application of LSTM RNNs to speech act recognition is novel to the best of our knowledge. LSTM RNNs have also been applied to sequence tagging in opinion mining (Irsoy and Cardie, 2014; Liu et al., 2015).

Conditional structured models There has been an explosion of interest in CRFs for solving structured output problems in NLP; see (Smith, 2011) for an overview. Linear chain (for sequence labeling) and tree structured CRFs (for parsing) are the common ones in NLP. However, speech act recognition in asynchronous conversation posits a different problem, where the challenge is to model arbitrary conversational structures. In this work we propose a general class of models based on pairwise CRFs that work on arbitrary graph structures.

Speech act recognition in asynchronous conversation Jeong et al. (2009) use semi-supervised boosting to tag the sentences in email and forum discussions with speech acts by adapting knowledge from spoken conversations. Other sentence-level approaches use supervised classifiers and sequence taggers (Qadir and Riloff, 2011; Tavafi et al., 2013; Oya and Carenini, 2014).

Cohen et al. (2004) first use the term *email speech act* for classifying emails based on their acts (deliver, meeting). Their classifiers do not capture any contextual dependencies between the acts. To model contextual dependencies, Carvalho and Cohen (2005) use a collective classification approach with two different classifiers, one for content and one for context, in an iterative algorithm. Our approach is similar in spirit to their approach with three crucial differences: (*i*) our CRFs are globally normalized to surmount the *label bias* problem, where their classifiers are normalized locally; (*ii*) the graph structure of the conversation is given in their case, which is not the case with ours; and (*iii*) their approach works at the comment level, where we work at the sentence level.

6 Conclusions and Future Work

We have presented a two-step framework for speech act recognition in asynchronous conversation. A LSTM RNN first composes sentences into vector representations by considering the word order. Then a pairwise CRF jointly models the inter-sentence dependencies in the conversation. We experimented with different LSTM variants (uni- vs. bi-directional, random vs. pretrained initialization), and different CRF variants depending on the underlying graph structure. We trained our models on many different settings using synchronous and asynchronous corpora and evaluated on two forum datasets, one of which is presented in this work.

Our results show that LSTM RNNs provide better representations but requires more data, and global joint models improve over local models given that it considers the right graph structure.

In the future, we would like to combine CRFs with LSTMs for doing the two steps jointly, so that the LSTMs can learn the embeddings using the global thread-level feedback. This would require the backpropagation algorithm to take error signals from the loopy BP inference. We would also like to apply our models to conversations, where the graph structure is extractable using the meta data or other clues, e.g., the fragment quotation graphs for email threads (Carenini et al., 2008).

Acknowledgments

We thank Aseel Ghazal for her effort in creating the QC3 corpus. This work is part of the Interactive sYstems for Answer Search (IYAS) project.

References

John Langshaw Austin. 1962. How to do things with words. *Harvard University Press*.

Srinivas Bangalore, Giuseppe Di Fabbrizio, and Amanda Stent. 2006. Learning the structure of task-driven human-human dialogs. In *Proceedings of the 44nd Annual Meeting on Association for Computational Linguistics, ACL'06*, pages 201–208. ACL.

Sumit Bhatia, Prakhar Biyani, and Prasenjit Mitra. 2014. Summarizing online forum discussions – can dialog acts of individual messages help? In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2127–2131, Doha, Qatar, October. ACL.

Giuseppe Carenini, Raymond T. Ng, and Xiaodong Zhou. 2008. Summarizing emails with conversational cohesion and subjectivity. In *Proceedings of the 46nd Annual Meeting on Association for Computational Linguistics, ACL'08*, pages 353–361, OH. ACL.

Vitor R. Carvalho and William W. Cohen. 2005. On the collective classification of email "speech acts". In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 345–352, New York, NY, USA. ACM Press.

William W. Cohen, Vitor R. Carvalho, and Tom M. Mitchell. 2004. Learning to classify email into "speech acts". In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 309–316.

Rajdip Dhillon, Sonali Bhagat, Hannah Carvey, and Elizabeth Shriberg. 2004. Meeting Recorder Project: Dialog Act Labeling Guide. Technical report, ICSI Tech. Report.

Donghui Feng, Erin Shaw, Jihie Kim, and Eduard Hovy. 2006. Learning to detect conversation focus of threaded discussions. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 208–215, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. ACL.

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

Liangjie Hong and Brian D. Davison. 2009. A classification-based approach to question answering in discussion boards. In *32nd Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, pages 171–178, Boston, USA. ACM Press.

Ozan Irsoy and Claire Cardie. 2014. Opinion mining with deep recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 720–728, Doha, Qatar. ACL.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Semi-supervised speech act recognition in emails and forums. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1250–1259, Singapore. ACL.

Shafiq Joty, Giuseppe Carenini, and Chin-Yew Lin. 2011. Unsupervised Modeling of Dialog Acts in Asynchronous Conversations. In *Proceedings of the twenty second International Joint Conference on Artificial Intelligence*, IJCAI'11, pages 1–130, Barcelona.

Dan Jurafsky, Liz Shriberg, and Debra Biasca. 1997. Switchboard SWBD-DAMSL Shallow-Discourse-Function Annotation Coders Manual, Draft 13. Technical report, University of Colorado at Boulder & +SRI International.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2304–2314, Lisbon, Portugal, September. ACL.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1433–1443, Lisbon, Portugal, September. ACL.

Kathleen McKeown, Lokesh Shrestha, and Owen Rambow. 2007. Using question-answer pairs in extractive summarization of email conversations. In *CICLing*, pages 542–550.

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

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

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 467–475, Stockholm, Sweden. Morgan Kaufmann Publishers Inc.

Kevin Murphy. 2012. *Machine Learning A Probabilistic Perspective*. The MIT Press.

Tatsuro Oya and Giuseppe Carenini. 2014. Extractive summarization and dialogue act modeling on email threads: An integrated probabilistic approach. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 133–140, Philadelphia, PA, U.S.A. ACL.

Sebastian Padó, 2006. *User's guide to `sigf`: Significance testing by approximate randomisation*.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ashequl Qadir and Ellen Riloff. 2011. Classifying sentences as speech acts in message board posts. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 748–758, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Sujith Ravi and Jihie Kim. 2007. Profiling Student Interactions in Threaded Discussions with Speech Act Classifiers. In *Proceedings of AI in Education Conference (AIED 2007)*.

Emanuel A. Schegloff. 1968. Sequencing in conversational openings. *American Anthropologist*, 70(6):1075–1095.

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

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Synthesis Lectures on Human Language Technologies. Morgan and Claypool, May.

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

Maryam Tavafi, Yashar Mehdad, Shafiq Joty, Giuseppe Carenini, and Raymond Ng. 2013. Dialogue act recognition in synchronous and asynchronous conversations. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, page 117–121, Metz, France. ACL.

Jan Ulrich, Gabriel Murray, and Giuseppe Carenini. 2008. A publicly available annotated corpus for supervised email summarization. In *AAAI'08 EMAIL Workshop*, Chicago, USA. AAAI.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Mark W. Schmidt, and Kevin P. Murphy. 2006. Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 969–976, Pittsburgh, USA. ACM.

Yair Weiss. 2001. Comparing the mean field method and belief propaga- tion for approximate inference in mrfs. In *Advanced Mean Field Methods*. MIT Press.

A Appendix

Algorithm 1: Online learning algorithm for conditional random fields

1. Initialize the model parameters $\mathbf{v}$ and $\mathbf{w}$;
2. **repeat**
   - **for** *each thread* $G = (V, E)$ **do**
     - a. Compute node and edge factors $\psi_n(y_i|\mathbf{z}, \mathbf{v})$ and $\psi_e(y_{i,j}|\mathbf{z}, \mathbf{w})$;
     - b. Infer node and edge marginals using sum-product loopy BP;
     - c. Update: $\mathbf{v} = \mathbf{v} - \eta \frac{1}{|V|} f'(\mathbf{v})$;
     - d. Update: $\mathbf{w} = \mathbf{w} - \eta \frac{1}{|E|} f'(\mathbf{w})$;
   - **end**
   - **until** *convergence*;

Tag	Description	BC3	TA
S	Statement	69.56%	65.62%
P	Polite mechanism	6.97%	9.11%
QY	Yes-no question	6.75%	8.33%
AM	Action motivator	6.09%	7.71%
QW	Wh-question	2.29%	4.23%
A	Accept response	2.07%	1.10%
QO	Open-ended question	1.32%	0.92%
AA	Acknowledge and appreciate	1.24%	0.46%
QR	Or/or-clause question	1.10%	1.16%
R	Reject response	1.06%	0.64%
U	Uncertain response	0.79%	0.65%
QH	Rhetorical question	0.75%	0.08%

Table 10: Dialog act tags and their relative frequencies in the BC3 and TA corpora.

Situation entity types: automatic classification of clause-level aspect

Annemarie Friedrich[1] **Alexis Palmer**[2] **Manfred Pinkal**[1]

[1]Department of Computational Linguistics, Saarland University, Germany
[2]Leibniz ScienceCampus, Dept. of Computational Linguistics, Heidelberg University, Germany
{afried,pinkal}@coli.uni-saarland.de palmer@cl.uni-heidelberg.de

Abstract

This paper describes the first robust approach to automatically labeling clauses with their situation entity type (Smith, 2003), capturing aspectual phenomena at the clause level which are relevant for interpreting both semantics at the clause level and discourse structure. Previous work on this task used a small data set from a limited domain, and relied mainly on words as features, an approach which is impractical in larger settings. We provide a new corpus of texts from 13 genres (40,000 clauses) annotated with situation entity types. We show that our sequence labeling approach using distributional information in the form of Brown clusters, as well as syntactic-semantic features targeted to the task, is robust across genres, reaching accuracies of up to 76%.

1 Introduction

Clauses in text have different aspectual properties, and thus contribute to the discourse in different ways. Distinctions that have been made in the linguistic and semantic theory literature include the classification of states, events and processes (Vendler, 1957; Bach, 1986), and whether clauses introduce particular eventualities or report regularities generalizing either over events or members of a kind (Krifka et al., 1995). Such aspectual distinctions are relevant to natural language processing tasks requiring text understanding such as information extraction (Van Durme, 2010) or temporal processing (Costa and Branco, 2012).

In this paper, we are concerned with automatically identifying the **type** of a **situation entity (SE)**, which we assume to be expressed by a clause. Specifically, we present a system for automatically labeling clauses using the inventory of SE types shown in Figure 1 (Smith, 2003, 2005; Palmer et al., 2007). The original motivation for the above inventory of SE types is the observation that different *modes of discourse*, a classification of linguistic properties of text at the passage level, have different distributions of SE types. For example, EVENTs and STATEs are predominant in *narrative* passages, while GENERIC SENTENCEs occur frequently in *information* passages.

STATE: *The colonel owns the farm.*
EVENT: *John won the race.*
REPORT: "...", *said Obama.*
GENERIC SENTENCE: Generalizations over kinds. *The lion has a bushy tail.*
GENERALIZING SENTENCE: Generalizations over events (*habituals*). *Mary often fed the cat last year.*
QUESTION: *Who wants to come?*
IMPERATIVE: *Hand me the pen!*

Figure 1: SE types, adapted from Smith (2003).

A previous approach to automatically labeling SE types (Palmer et al., 2007) – referred to here as UT07 – captures interesting insights, but is trained and evaluated on a relatively small amount of text (about 4300 clauses), mainly from one rather specialized subsection of the Brown corpus. The data shows a highly skewed distribution of SE types and was annotated in an intuitive fashion with only moderate agreement. In addition, the UT07 system relies mostly on part-of-speech tags and words as features. The latter are impractical when dealing with larger data sets and capture most of the corpus vocabulary, overfitting the model to the data set. Despite this overfitting, the system's accuracy is only around 53%.

We address these shortcomings, developing a robust system that delivers high performance compared to the human upper bound across a range of genres. Our approach uses features which increase

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1757–1768,
Berlin, Germany, August 7-12, 2016.

robustness: Brown clusters and syntactic-semantic features. Our models for labeling texts with the aspectual properties of clauses in the form of SE types reach accuracies of up to 76%.

In an oracle experiment, Palmer et al. (2007) show that including the gold labels of the previous clauses as features into their maximum entropy model is beneficial. We implement the first true sequence labeling model for SE types, using conditional random fields to find the globally-best sequence of labels for the clauses in a document. Performance increases by around 2% absolute compared to predicting labels for clauses separately; much of this effect stems from the fact that GENERIC SENTENCEs often occur together.

Moving well beyond the single-domain setting, our models are trained and evaluated on a multi-genre corpus of approximately 40,000 clauses from MASC (Ide et al., 2008) and Wikipedia which have been annotated with substantial agreement. We train and test our models both within genres and across genres, highlighting differences between genres and creating models that are robust across genres. Both the corpus and the code for an SE type labeler are freely available.[1] These form the basis for future research on SE types and related aspectual phenomena and will enable the inclusion of SE type information as a preprocessing step into various NLP tasks.

2 Linguistic background

The inventory of SE types proposed by Smith (2003) consists of three high-level categories, each with two subtypes. *Eventualities* comprise EVENT and STATE, categories for clauses representing actual happenings, states of the world, or attributes of entities or situations. *General Statives* include GENERIC SENTENCE and GENERALIZING SENTENCE and reflect regularities in the world or general information predicated over classes or kinds. Finally, *Abstract Entities* (Figure 2) have the subtypes FACT and PROPOSITION. Although *Abstract Entities* are part of the label inventory for UT07, we treat them in a separate identification step, for reasons discussed in Section 7. The inventory was expanded by Palmer et al. (2007) to include three additional types: REPORT, QUESTION and IMPERATIVE. The latter two categories were added to accommodate exhaustive annotation of text; REPORT is a subtype of event for attributions of quoted speech.

FACT: Objects of knowledge.
I know that Mary refused the offer.
PROPOSITION: Objects of belief.
I believe that Mary refused the offer.

Figure 2: *Abstract Entity* SE types.

Two parts of a clause provide important information for determining the SE type (Friedrich and Palmer, 2014b): a clause's **main verb** and its **main referent**. The latter is loosely defined as the main entity that the segment is about; in English this is usually the subject. For example, main referents of GENERIC SENTENCEs are kinds or classes as in "*Elephants are huge*", while the main referents of *Eventualities* and GENERALIZING SENTENCEs are particular individuals ("*John is short*"). For English, the main verb is the non-auxiliary verb ranked highest in the dependency parse (e.g. "*kiss*" in "*John has kissed Joe*"). STATEs and EVENTs differ in the *fundamental lexical aspectual class* (Siegel and McKeown, 2000) of their main verbs (e.g. **dynamic** in "*She filled the glass with water*" vs. **stative** in "*Water fills the glass*"). While fundamental lexical aspectual class is a word-sense level attribute of the clause's main verb, *habituality* is a property of the entire clause which is helpful to determine the clause's SE type. For example, EVENT and GENERALIZING SENTENCE differ in habituality (e.g. **episodic** in "*John cycled to work yesterday*" vs. **habitual** in "*John cycles to work*").

Like habituality, SE types are a categorization at the clause level. Properties of the clause such as modals, negation, or the perfect influence the SE type: for instance, "*John might win*" is treated as a STATE as it describes a possible state of the world rather than an EVENT. Such coercions happen only for clauses which, without the trigger for aspectual shift, would be EVENTs; other SEs retain their type even under coercions such as negation, e.g., "*Elephants are not small*" is a GENERIC SENTENCE. SE types aim to capture how clauses behave in discourse, and the types STATE and EVENT are aspectual rather than ontological categories. The types reflect not so much semantic content of a clause as its manner of presentation, and all parts of a clause contribute to determining its SE type.

[1] Corpora, annotation manual and code available at www.coli.uni-saarland.de/projects/sitent

3 Related work

SE types model aspect at the clause level; thus they are most closely related to other works performing automatic classification for various aspect-related phenomena of the verb or the clause. For example, Vendler classes (Vendler, 1957) ascribe four categories as lexical properties of verbs, distinguishing states from three types of events (accomplishment, achievement, and activity), differing according to temporal and aspectual properties (e.g. telicity and punctuality). The work of Siegel and McKeown (2000) is a major inspiration in computational work on modeling these linguistic phenomena, introducing the use of linguistic indicators (see Section 5.1). Hermes et al. (2015) model Vendler classes computationally on a verb-type level for 95 different German verbs, combining distributional vectors with supervised classification. Zarcone and Lenci (2008) investigate both supervised and unsupervised classification frameworks for occurrences of 28 Italian verbs, and Friedrich and Palmer (2014a) predict lexical aspectual class for English verbs in context.

The only previous approach to automatic classification of SE types comes from Palmer et al. (2007). This system (UT07) uses word and POS tag features as well as a number of lexical features adopted from theoretical work on aspectual classification. The model is described in Section 6.1.

Another related body of work has to do with determining event class as a precursor to temporal relation classification. The inventory of event classes, described in detail in the TimeML annotation guidelines (Saurí et al., 2006), combines semantic (REPORTING, PERCEPTION), aspectual (ASPECTUAL, STATE, OCCURRENCE), and intensional (I_ACTION, I_STATE) properties of events.

Finally, there are close connections to systems which predict genericity of noun phrases (Reiter and Frank, 2010; Friedrich and Pinkal, 2015a), and habituality of clauses (Mathew and Katz, 2009; Friedrich and Pinkal, 2015b).

4 Data sets

The experiments presented in this paper make use of two data sets labeled with SE types.

Brown data. This data set consists of 20 texts from the *popular lore* section of the Brown corpus (Francis and Kučera, 1979), manually segmented into 4391 clauses and marked by two annotators in an intuitive way with κ=0.52 (Cohen, 1960). Final labels were created via adjudication. The texts are essays and personal stories with topics ranging from maritime stories to marriage advice.

corpus	tokens	SEs	Fleiss' κ
MASC	357078	30333	0.69
Wikipedia	148040	10607	0.66

Table 1: SE-labeled corpora: **size** and **agreement**.

SE type	MASC	Wiki	Fleiss κ*
STATE	49.8	24.3	0.67
EVENT	24.3	18.9	0.74
REPORT	4.8	0.9	0.80
GENERIC	7.3	49.7	0.68
GENERALIZING	3.8	2.5	0.43
QUESTION	3.3	0.1	0.91
IMPERATIVE	3.2	0.2	0.94
undecided	2.4	2.1	-

Table 2: **Distribution of SE types** in gold standard (%). *Krippendorff's diagnostics.

MASC and Wikipedia. Our main data set consists of documents from MASC (Ide et al., 2010) and Wikipedia. The MASC data covers 12 of the written genres (see Table 11). Texts are split into clauses using SPADE (Soricut and Marcu, 2003) with some heuristic post-processing, and the clauses are labeled by three annotators independently. Annotators, all student assistants with basic linguistic training, were given an extensive annotation manual. Table 1 reports agreement over types in terms of Fleiss' κ (Fleiss, 1971). As we do not force annotators to assign a label to each clause, we compute κ using all pairings of labels where both annotators assigned an SE type. The gold standard is constructed via majority voting.

Table 2 shows the distribution of SE types. The largest proportion of segments in MASC are STATEs, while the largest proportion in Wikipedia are GENERIC SENTENCEs. The Wikipedia data was collected to supplement MASC, which contains few generics and no data from an encyclopedic genre. Within MASC, the various genres' distributions of SE types differ as well, and agreement scores also vary: some genres contain many instances of easily classifiable SE types, while others (e.g., essays or journal) are more difficult to annotate (more details in Section 6.6).

The rightmost column of Table 2 shows the values for Krippendorff's diagnostics (Krippendorff, 1980), a tool for determining which categories hu-

group	explanation	examples
mv	features describing the SE's **main verb** & its arguments	tense, lemma, lemma of object, auxiliary, WordNet sense and hypernym sense, progressive, POS, perfect, particle, voice, linguistic indicators
mr	features describing the **main referent**, i.e., the NP denoting the main verb's subject	lemma, determiner type, noun type, number, WordNet sense and supersense, dependency relations linked to this token, person, countability, bare plural
cl	features describing entire **clause** that invokes the SE	presence of adverbs / prepositional clauses, conditional, modal, whether subject before verb, negated, verbs embedding the clause

Table 3: Overview of feature set **B**. The full and detailed list is available (together with the implementation) at `http://www.coli.uni-saarland.de/projects/sitent`.

mans had most difficulties with. For each category, one computes κ for an artificial set-up in which all categories except one are collapsed into an artificial OTHER category. A high value indicates that annotators can distinguish this SE type well from others. GENERALIZING SENTENCEs are most difficult to agree upon. For all frequently occurring types as well as QUESTIONs and IMPERATIVEs, agreement is substantial.

Agreement on QUESTION and IMPERATIVEs is not perfect even for humans, as identifying them requires recognizing cases in reported speech, which is not a trivial task (e.g., Brunner, 2013). To illustrate another difficult case, consider the example "You must never confuse faith", which was marked as both IMPERATIVE and GENERIC SENTENCE, by different annotators.

5 Method

This section describes the feature sets and classification methods used in our approach, which models SE type labeling as a supervised sequence labeling task.

5.1 Feature sets

Our feature sets are designed to work well on large data sets, across genres and domains. Features are grouped into two sets: **A** consists of standard NLP features including POS tags and Brown clusters. Set **B** targets SE labeling, focusing on syntactic-semantic properties of the main verb and main referent, as well as properties of the clause which indicate its aspectual nature. Texts are pre-processed with Stanford CoreNLP (Manning et al., 2014), including tokenization, POS tagging (Toutanova et al., 2003) and dependency parsing (Klein and Manning, 2002) using the UIMA-based DKPro framework (Ferrucci and Lally, 2004; Eckart de Castilho and Gurevych, 2014).

A-pos: part-of-speech tags. These features count how often each POS tag occurs in a clause.

A-bc: Brown cluster features. UT07 relies mostly on words and word/POS tag pairs. These simple features work well on the small Brown data set, but the approach quickly becomes impractical with increasing corpus size. We instead turn to distributional information in the form of Brown clusters (Brown et al., 1992), which can be learned from raw text and represent word classes in a hierarchical way. Originally developed in the context of n-gram language modeling, they aim to assign words to classes such that the average mutual information of the words in the clusters is maximized. We use existing, freely-available clusters trained on news data by Turian et al. (2010) using the implementation by Liang (2005).[2] Clusterings with 320 and 1000 Brown clusters work best for our task. We use one feature per cluster, counting how often a word in the clause was assigned to this cluster (0 for most clusters).

B-mv: main verb. Using dependency parses, we extract the verb ranked highest in the clause's parse as the main verb, and extract the set of features listed in Table 3 for that token. Features based on WordNet (Fellbaum, 1998) use the most frequent sense of the lemma. Tense and voice information is extracted from sequences of POS tags using a set of rules (Loaiciga et al., 2014). Linguistic indicators (Siegel and McKeown, 2000) are features collected per verb type over a large parsed background corpus, encoding how often a verb type occurred with each linguistic marker, e.g., in past tense or with an *in*-PP. We use values collected from Gigaword (Graff et al., 2003); these are freely available at our project web site (Friedrich and Palmer, 2014a).

B-mr: main referent. We extract the grammatical subject of the main verb (i.e., `nsubj` or `nsubjpass`) as the clause's main referent. While the main verb must occur within the clause, the

[2] `http://metaoptimize.com/projects/wordreprs`

main referent may be a token either within or outside the clause. In the latter case, it still functions as the clause's main referent, as in most cases it can be considered an implicit argument within the clause. Table 3 lists the features extracted for the main referent.

B-cl: clause. These features (see also Table 3) describe properties at the clause level, capturing both grammatical phenomena such as word order and lexical phenomena including presence of particular adverbials or prepositional phrases, as well as semantic information such as modality. If the clause's main verb is embedded in a `ccomp` relation, we also use features describing the respective governing verb.

5.2 Classification / sequence labeling model

Our core modeling assumption is to view a document as a sequence of SE type labels, each associated with a clause; this motivates the choice of using a conditional random field (CRF, Lafferty et al. (2001)) for label prediction. The conditional probability of label sequence $\vec{y}$ given an observation sequence $\vec{x}$ is given by:

$$P(\vec{y}|\vec{x}) = \frac{1}{Z(\vec{x})} exp(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i(y_{j-1}, y_j, \vec{x}, j)),$$

with $Z(\vec{x})$ being a normalization constant (see also Klinger and Tomanek (2007)). λ_i, the weights of the feature functions, are learned via L-BGFS (Wright and Nocedal, 1999).

We create a linear chain CRF using the CRF++ toolkit[3] with default parameters, applying two forms of feature functions: $f_i(y_j, x_j)$ and $f_i(y_{j-1}, y_j)$. The former consists of indicator functions for combinations of SE type labels and each of the features listed above. The latter type of feature function (also called "bigram" features in CRF++ terminology) gets instantiated as indicator functions for each combination of labels, thereby enabling the model to take sequence information into account. When using only the former type of feature function, our classifier is equivalent to a maximum entropy (MaxEnt) model.

Side remark: pipeline approach. Feature set **B** is inspired by previous work on two subtasks of assigning an SE type to a clause (see Section 3): (a) identifying the genericity of a noun phrase in its clausal context, and (b) identifying whether a clause is episodic, habitual or static. This information can in turn be used to determine a clause's SE type label in a rule-based way, e.g., GENERALIZING SENTENCES are habitual clauses with a non-generic main referent. As our corpus is also annotated with this information, we also trained separate models for these subtasks and assigned the SE type label accordingly. However, such a pipeline approach is not competitive with the model trained directly on SE types (see Section 6.3).

[3] `https://code.google.com/p/crfpp`

6 Experiments

Here we present our experiments on SE type classification, beginning with a (near) replication of the UT07 system, and moving on to evaluate our new approach from multiple perspectives.

6.1 Replication and extension of UT07

As a first step, we implement a system similar to UT07, which relies on the features summarized in Table 4. For W and T features, we set a frequency threshold of 7 occurrences. Feature set L comprises sets of predicates assumed to correlate with particular SE types, and whether or not the clause contains a modal or finite verb. Set G includes all verbs of the clause and their POS-tags. UT07 additionally uses CCG supertags and grammatical function information. The UT07 system approximates a sequence labeling model by adding the predicted labels of previous clauses as *lookback* (LB) features. To parallel their experiments, we train both MaxEnt and CRF models, as explained in Section 5.2. Results on the Brown data, with the same training/test split, appear in Table 5. Unlike the LB model, our CRF predicts the label sequence jointly and outperforms UT07 on the Brown data by up to 7% accuracy. We assume that the performance boost in the MaxEnt setting is at least partially due to having better parses.

In sum, on the small Brown data set, a CRF approach successfully leverages sequence information, and a simple set of features works well. Preliminary experiments applying our new features on Brown data yield no improvements, suggesting that word-based features overfit this domain.

W	words
T	POS tags, word/POS tag combinations
L	linguistic cues
G	grammatical cues

Table 4: Features used in baseline UT07.

features	Palmer et al. (2007) MaxEnt	LB	our implementation MaxEnt	CRF
W	45.4	46.6	48.8	47.0
WT	49.9	52.4	52.9	53.7
WTL	48.9	50.5	51.6	55.8
WTLG	**50.6**	**53.1**	**55.8**	**60.0**

Table 5: **Accuracy** on Brown. Test set majority class (STATE) is 35.3%. LB = results for best look-back settings in MaxEnt. 787 test instances.

6.2 Experimental settings

We develop our models using **10-fold cross validation (CV)** on 80% (counted in terms of the number of SEs) of the MASC and Wikipedia data (a total of 32,855 annotated SEs), keeping the remaining 20% as a held-out test set. Development and test sets each contain distinct sets of documents; the documents of each MASC genre and of Wikipedia are distributed over the folds. We report results in terms of macro-average precision, macro-average recall, macro-average F1-measure (harmonic mean of macro-average precision and macro-average recall), and accuracy. We apply McNemar's test (McNemar, 1947) with $p < 0.01$ to test significance of differences in accuracy. In the following tables, we mark numerically-close scores with the same symbols if they are found to be significantly different.

Upper bound: human performance. Labeling clauses with their SE types is a non-trivial task even for humans, as there are many borderline cases (see Sections 4 and 8). We compute an upper bound for system performance by iterating over all clauses: for each pair of human annotators, two entries are added to a co-occurrence matrix (similar to a confusion matrix), with each label serving once as "gold standard" and once as the "prediction." From this matrix, we can compute scores in the same manner as for system predictions. Precision and recall scores are symmetric in this case, and accuracy corresponds to observed agreement.

6.3 Impact of feature sets

We now compare various configurations of our CRF-based SE type labeler, experimenting with the feature sets as described in Section 5.1. Table 6 shows the results for 10-fold CV on the dev portion of the MASC+Wiki corpus.

Each feature set on its own outperforms the majority class baseline. Of the individual feature groups, **bc** and **mv** have the highest predictive power; both capture lexical information of the main verb. Using sets **A** and **B** individually results in similar scores; their combination increases accuracy on the dev set by an absolute 3.6-4.3%. Within **A** and **B**, each subgroup contributes to the increase in performance (not shown in table).

feature set	P	R	F	acc.
maj. class (STATE)	6.4	14.3	8.8	45.0
A	70.1	61.4	65.4	∗†72.1
pos	49.3	40.3	44.3	58.7
bc	67.5	55.8	61.1	∗70.6
B	69.5	62.7	66.9	⋆‡72.8
mr	36.4	26.8	30.9	51.7
mv	62.3	52.4	56.9	⋆70.8
cl	53.3	41.2	46.6	52.8
A+B	**74.1**	**68.6**	**71.2**	‡†**76.4**
upper bound (humans)	78.6	78.6	78.6	79.6

Table 6: **Impact of different feature sets.** Wiki+MASC dev set, CRF, 10-fold CV.

feature set	P	R	F	acc.
maj. class (STATE)	6.4	14.3	8.8	44.7
A	67.6	60.6	63.9	∗69.8
B	69.9	61.7	65.5	†71.4
A+B	**73.4**	**65.5**	**69.3**	∗†**74.7**

Table 7: Results on MASC+Wiki **held-out test set** (7937 test instances).

Finally, having developed exclusively on the dev set, we run the system on the held-out test set, training on the entire dev set. Results (in Table 7) show the same tendencies as for the dev set: each feature set contributes to the final score, and the syntactic-semantic features targeted at classifying SE types (i.e. **B**) are helpful.

Ablation. To gain further insight, we ablate each feature subgroup from the full system, see Table 8. Again, **bc** features and **mv** features are identified as the most important ones. The other feature groups partially carry redundant information when combining **A** and **B**. Next, we rank features by their information gain with respect to the SE types. In Table 3, the features of each group are ordered by this analysis. Ablating single features from the full system does not result in significant performance losses. However, using only selected, top features for our system decreased performance, possibly because some features cover rare but important cases, and because the feature selection algorithm does not take into account the information features may provide regarding tran-

feature set	P	R	F	acc.
A+B	**74.1**	**68.6**	**71.2**	**76.4**
- bc	71.3	65.7	68.4	74.5
- pos	73.4	67.4	70.2	76.0
- mr	73.7	67.4	70.4	75.9
- mv	72.3	64.2	68.0	73.6
- cl	73.1	67.6	70.2	76.0

Table 8: **Impact of feature groups: ablation** Wiki+MASC dev set, CRF, 10-fold CV. All accuracies for ablation settings are significantly different from A+B.

sitions (Klinger and Friedrich, 2009). In addition, CRFs are known to be able to deal with a large number of potentially dependent features.

Side remark: pipeline approach. We here use the subset of SEs labeled as EVENT, STATE, GENERIC SENTENCE or GENERALIZING SENTENCE because noun phrase genericity and habituality is not labeled for IMPERATIVE and QUESTION, and REPORT is identified lexically based on the main verb rather than these semantic features. Models for subtasks of SE type identification, i.e., (a) genericity of noun phrases and (b) habituality reach accuracies of (a) 86.8% and (b) 83.6% (on the relevant subset). Applying the labels output by these two systems as (the only) features in a rule-based way using a J48 decision tree (Witten et al., 1999) results in an accuracy of 75.5%, which is lower than 77.2%, the accuracy of the CRF which directly models SE types (when using only the above four types).

6.4 Impact of amount of training data

Next we test how much training data is required to get stable results for SE type classification. Figure 3 shows accuracy and F1 for 10-fold CV using **A+B**, with training data downsampled to different extents in each run by randomly removing documents. Up to the setting which uses about 60% of the training data, performance increases steadily. Afterwards, the curves start to level off. We conclude that robust models can be learned from our corpus. Adding training data, especially in-domain data, will, as always, be beneficial.

6.5 Impact of sequence labeling approach

Palmer et al. (2007) suggest that SE types of nearby clauses are a useful source of information. We further test this hypothesis by comparing our sequence labeling model (CRF) to two additional

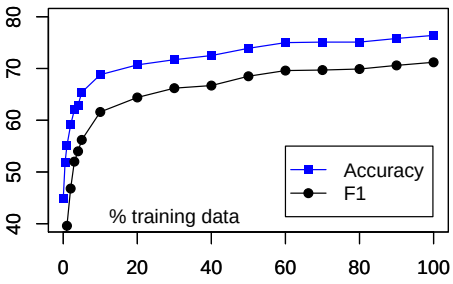


Figure 3: **Learning curve** for MASC+Wiki dev.

models: (1) a MaxEnt model, which labels clauses in isolation, and (2) a MaxEnt model including the correct label of the preceding clause (seq-oracle), simulating an upper bound for the impact of sequence information on our system.

Table 9 shows the results. Scores for GENERALIZING SENTENCE are the lowest as this class is very infrequent in the data set. The most striking improvement of the two sequence labeling settings over MaxEnt concerns the identification of GENERIC SENTENCES. These often "cluster" in texts (Friedrich and Pinkal, 2015b) and hence their identification profits from using sequence information. The results for seq-oracle show that the sequence information is useful for STATE, GENERIC and GENERALIZING SENTENCES, but that no further improvement is to be expected from this method for the other SE types. We conclude that the CRF model is to be preferred over the MaxEnt model; in almost all of our experiments it performs significantly better or equally well.

SE type	MaxEnt	CRF	seq-oracle
STATE	79.1	**80.6**	81.7
EVENT	77.5	**78.6**	78.3
REPORT	78.2	**78.9**	78.3
GENERIC	61.3	**68.3**	73.5
GENERALIZING	25.0	**29.4**	38.1
IMPERATIVE	72.3	**75.3**	74.7
QUESTION	84.4	**84.4**	83.8
macro-avg P	71.5	**74.1**	75.5
macro-avg R	66.1	**68.6**	70.4
macro-avg F1	68.7	**71.2**	73.9
accuracy	∗74.1	∗†**76.4**	†77.9

Table 9: **Impact of sequence information:** (**F1** by SE type): CRF, Masc+Wiki, 10-fold CV.

6.6 Impact of genre

In this section, we test to what extent our models are robust across genres. Table 10 shows F1-scores for each SE type for two settings: the 10-

SE type	genre-CV	10-fold CV	humans
STATE	78.2	80.6	82.8
EVENT	77.0	78.6	80.5
REPORT	76.8	78.9	81.5
GENERIC	44.8	68.3	75.1
GENERALIZING	27.4	29.4	45.8
IMPERATIVE	70.8	75.3	93.6
QUESTION	81.8	84.4	90.7
macro-avg F1	66.6	71.2	78.6
accuracy	∗71.8	∗76.4	79.6

Table 10: **Impact of in-genre training data**. F1-score by SE type, CRF, MASC+Wiki dev.

fold CV setting as explained in section 6.2, and a **genre-CV** setting, simulating the case where no in-genre training data is available, treating each genre as one cross validation fold. As expected, in the latter setting, both overall accuracy and macro-average F1 are lower compared to the case when in-genre training data is available. Nevertheless, our model is able to capture the nature of SE types across genres: the prediction of STATE, EVENT, REPORT and QUESTION is relatively stable even in the case of not having in-genre training data. An EVENT seems to be easily identifiable regardless of the genre. GENERIC SENTENCE is a problematic case; in the genre-CV setting, its F1-score drops by 23.5%. The main reason for this is that the distribution of SE types in Wikipedia differs completely from the other genres (see section 4). Precision for GENERIC SENTENCE is at 70.5% in the genre-CV setting, but recall is only 32.8% (compared to 70.1% and 66.6% in the 10-fold CV setting). Genericity seems to work differently in the various genres: most generics in Wikipedia clearly refer to kinds (e.g., lions or plants), while many generics in essays or letters are instances of more abstract concepts or generic *you*.

Results by genre. Next, we drill down in the evaluation of our system by separately inspecting results for individual genres. Table 11 shows that performance differs greatly depending on the genre. In some genres, the nature of SE types seems clearer to our annotators than in others, and this is reflected in the system's performance. The majority class is GENERIC SENTENCE in wiki, and STATE in all other genres. In the 'same genre' setting, 10-fold CV was performed within each genre. Adding out-of-genre training data improves macro-average F1 especially for genres with low scores in the 'same genre' setting. This boost is

		training data			
	maj. class	same genre	all	humans	
genre	%	F1	F1	F1	κ
blog	57.6	57.3	**64.9**	72.9	0.62
email	68.6	63.6	**66.4**	67.0	0.65
essays	49.4	33.5	**62.1**	64.6	0.54
ficlets	44.7	60.2	**65.7**	81.7	0.80
fiction	45.8	63.0	**66.0**	76.7	0.77
govt-docs	60.9	26.6	**67.6**	72.6	0.57
jokes	34.9	66.2	**69.8**	82.0	0.77
journal	59.3	35.8	**59.8**	63.7	0.52
letters	57.3	51.9	**65.1**	68.0	0.66
news	52.2	54.6	**64.1**	78.6	0.75
technical	57.7	31.4	**59.4**	54.7	0.55
travel	25.9	39.9	**58.1**	48.9	0.59
wiki	51.6	53.1	**63.0**	69.2	0.66

Table 11: **Macro-avg. F1 by genre**, CRF, 10-fold CV. Majority class given in % of clauses.

due to adding training data for types that are infrequent in that genre. Accuracy (not shown in table) improves significantly for blog, essays, govt-docs, jokes, and journal, and does not change for the remaining genres. We conclude that it is extremely beneficial to use our full corpus for training, as robustness of the system is increased, especially for SE types occurring infrequently in some genres.

7 Identification of Abstract Entities

Our system notably does not address one of Smith's main SE categories: *Abstract Entities*, introduced in Section 2. These SEs are expressed as clausal arguments of certain predicates such as (canonically) *know* or *believe*. Note that the *Abstract Entity* subtypes FACT and PROPOSITION do not imply that a clause's propositional content is true or likely from an objective point of view, they rather indicate that the clause's content is introduced to the discourse as an object of knowledge or belief, respectively. Following Smith (2003), we use PROPOSITION in a different sense than the usual meaning of "proposition" in semantics - naturally situation entities of any type may have propositional content. Smith's use of the term (and thus ours too) contrasts PROPOSITION with FACT - our PROPOSITIONS are simply sentences presented as a belief (or with uncertain evidential status) of the writer or speaker. This use of "proposition" also occurs in linguistic work by Peterson (1997) on factive vs. propositional predicates.

During the creation of the corpus, annotators

were asked to give one label out of the SE types included in our classification task, and to mark the clause with one of the *Abstract Entity* subtypes in addition if applicable. Analysis of the data shows that our annotators frequently forgot to mark clauses as *Abstract Entities*, which makes it difficult to model these categories correctly. As a first step toward resolving this issue, we implement a filter which automatically identifies *Abstract Entities* by looking for clausal complements of certain predicates. The list of predicates is compiled using WordNet synonyms of *know*, *think*, and *believe*, as well as predicates extracted from FactBank (Sauri and Pustejovsky, 2009) and TruthTeller (Lotan et al., 2013). Many of the clauses automatically identified as *Abstract Entities* are cases that annotators missed during annotation. We thus performed a post-hoc evaluation, presenting these clauses in context to annotators and asking whether the clause is an *Abstract Entity*. The so-estimated precision of our filter is 85.8% (averaged over 3 annotators). Agreement for this annotation task is $\kappa = 0.54$, with an observed agreement of 88.7%. Our filter finds 80% of the clauses labeled as *Abstract Entity* by at least one annotator in the gold standard; this is *approximately* its recall.

8 Conclusion

We have presented a system for automatically labeling clauses with their SE type which is mostly robust to changes in genre and which reaches accuracies of up to 76%, comparing favorably to the human upper bound of 80%. The system benefits from capturing contexual effects by using a linear chain CRF with label bigram features. In addition, the distributional and targeted syntactic-semantic features we introduce enable SE type prediction for large and diverse data sets. Our publicly available system can readily be applied to any written English text, making it easy to explore the utility of SE types for other NLP tasks.

Discussion. Our annotation scheme and guidelines for SE types (Friedrich and Palmer, 2014b) follow established traditions in linguistics and semantic theory. When applying these to a large number of natural texts, though, we came across a number of borderline cases where it is not easy to select just one SE type label. As we have reported before (Friedrich et al., 2015), the most difficult case is the identification of GENERIC SENTENCEs, which are defined as making a statement about a kind or class. We find that making this task becomes particularly difficult for argumentative essays (Becker et al., to appear).

Future work. A next major step in our research agenda is to integrate SE type information into various applications, including argument mining, temporal reasoning, and summarization. Together with the mode of progression through the text, e.g., temporal or spatial, SE type distribution is a key factor for a reader or listener's intuitive recognition of the discourse mode of a text passage. Therefore the automatic labeling of clauses with their SE type is a prerequisite for automatically identifying a passage's discourse mode, which in turn has promising applications in many areas of NLP, as the mode of a text passage has implications for the linguistic phenomena to be found in the passage. Examples include temporal processing of text (Smith, 2008), summarization, or machine translation (for work on genres see van der Wees et al., 2015). Here we focus on the automatic identification of SE types, leaving the identification of discourse modes to future work.

The present work, using the SE type inventory introduced by Smith (2003), is also the basis for research on more fine-grained aspectual type inventories. Among others, we plan to create subtypes of the STATE label, which currently subsumes clauses stativized by negation, modals, lexical information or other aspectual operators. Distinguishing these is relevant for temporal relation processing or veridicality recognition.

Acknowledgments

We thank the anonymous reviewers and Andrea Horbach for their helpful comments related to this work, and our annotators Melissa Peate Sørensen, Christine Bocionek, Kleo-Isidora Mavridou, Fernando Ardente, Damyana Gateva, Ruth Kühn and Ambika Kirkland. This research was supported in part by the MMCI Cluster of Excellence of the DFG, and the first author is supported by an IBM PhD Fellowship. The second author is funded by the Leibniz ScienceCampus *Empirical Linguistics and Computational Language Modeling*, supported by Leibniz Association grant no. SAS-2015-IDS-LWC and by the Ministry of Science, Research, and Art of Baden-Württemberg.

References

Emmon Bach. 1986. The algebra of events. *Linguistics and philosophy* 9(1):5–16.

Maria Becker, Alexis Palmer, and Anette Frank. to appear. Argumentative texts and clause types. In *Proceedings of the 3rd Workshop on Argument Mining, ACL 2016*.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics* 18(4):467–479.

Annelen Brunner. 2013. Automatic recognition of speech, thought, and writing representation in german narrative texts. *Literary and linguistic computing* 28(4):563–575.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20(1):37–46.

Francisco Costa and António Branco. 2012. Aspectual type and temporal relation classification. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Richard Eckart de Castilho and Iryna Gurevych. 2014. A broad-coverage collection of portable NLP components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*. Dublin, Ireland.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

David Ferrucci and Adam Lally. 2004. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering* 10(3-4):327–348.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76(5):378.

W. Nelson Francis and Henry Kučera. 1979. *A Standard Corpus of Present-Day Edited American English, for use with Digital Computers*. Brown University.

Annemarie Friedrich and Alexis Palmer. 2014a. Automatic prediction of aspectual class of verbs in context. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Baltimore, USA.

Annemarie Friedrich and Alexis Palmer. 2014b. Situation entity annotation. *In Proceedings of the Linguistic Annotation Workshop VIII* .

Annemarie Friedrich, Alexis Palmer, Melissa Peate Srensen, and Manfred Pinkal. 2015. Annotating genericity: a survey, a scheme, and a corpus. In *Proceedings of the 9th Linguistic Annotation Workshop (LAW IX)*. Denver, Colorado, US.

Annemarie Friedrich and Manfred Pinkal. 2015a. Automatic recognition of habituals: a three-way classification of clausal aspect. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Lisbon, Portugal.

Annemarie Friedrich and Manfred Pinkal. 2015b. Discourse-sensitive Automatic Identification of Generic Expressions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL)*. Beijing, China.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia* .

Jürgen Hermes, Michael Richter, and Claes Neuefind. 2015. Automatic induction of German aspectual verb classes in a distributional framework. In *Proceedings of the International Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*.

Nancy Ide, Collin Baker, Christiane Fellbaum, and Charles Fillmore. 2008. MASC: The manually annotated sub-corpus of American English .

Nancy Ide, Christiane Fellbaum, Collin Baker, and Rebecca Passonneau. 2010. The manually annotated sub-corpus: A community resource for and by the people. In *Proceedings of the ACL 2010 conference short papers*.

Dan Klein and Christopher D Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*.

Roman Klinger and Christoph M Friedrich. 2009. Feature subset selection in conditional random fields for named entity recognition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP)*.

Roman Klinger and Katrin Tomanek. 2007. Classical probabilistic models and conditional random fields. *TU Dortmund Algorithm Engineering Report* .

Manfred Krifka, Francis Jeffrey Pelletier, Gregory N. Carlson, Alice ter Meulen, Godehard Link, and Gennaro Chierchia. 1995. Genericity: An Introduction. *The Generic Book* pages 1–124.

Klaus Krippendorff. 1980. *Content analysis: An introduction to its methodology*. Sage.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*.

Percy Liang. 2005. *Semi-supervised learning for natural language*. Ph.D. thesis, University of California Berkeley.

Sharid Loaiciga, Thomas Meyer, and Andrei Popescu-Belis. 2014. English-French Verb Phrase Alignment in Europarl for Tense Translation Modeling. In *The Ninth Language Resources and Evaluation Conference (LREC)*.

Amnon Lotan, Asher Stern, and Ido Dagan. 2013. TruthTeller: Annotating Predicate Truth. In *Proceedings of NAACL 2013*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*.

Thomas A. Mathew and Graham E. Katz. 2009. Supervised categorization for habitual versus episodic sentences. In *Sixth Midwest Computational Lingustics Colloquium*. Indiana University, Bloomington, Indiana.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12(2):153–157.

Alexis Palmer, Elias Ponvert, Jason Baldridge, and Carlota Smith. 2007. A sequencing model for situation entity classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*.

Philip L Peterson. 1997. On representing event reference. In *Fact Proposition Event*, Springer, pages 65–90.

Nils Reiter and Anette Frank. 2010. Identifying Generic Noun Phrases. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Uppsala, Sweden.

Roser Saurí, Jessica Littman, Bob Knippen, Rober Gaizauskas, Andrea Setzer, and James Pustejovsky. 2006. TimeML Annotation Guidelines, Version 1.2.1. Technical report.

Roser Sauri and James Pustejovsky. 2009. Factbank 1.0 ldc2009t23. Web Download. Philadelphia: Linguistic Data Consortium.

Eric V Siegel and Kathleen R McKeown. 2000. Learning methods to combine linguistic indicators: Improving aspectual classification and revealing linguistic insights. *Computational Linguistics* 26(4):595–628.

Carlota S Smith. 2003. *Modes of discourse: The local structure of texts*, volume 103. Cambridge University Press.

Carlota S Smith. 2005. Aspectual entities and tense in discourse. In *Aspectual inquiries*, Springer, pages 223–237.

Carlota S Smith. 2008. Time with and without tense. In *Time and modality*, Springer, pages 227–249.

Radu Soricut and Daniel Marcu. 2003. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*.

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

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*.

Marlies van der Wees, Arianna Bisazza, Wouter Weerkamp, and Christof Monz. 2015. What's

in a Domain? Analyzing Genre and Topic differences in Statistical Machine Translation. In *Proceedings of the 53rd Meeting of the Association for Computational Linguistics (ACL)*.

Benjamin D Van Durme. 2010. *Extracting implicit knowledge from text*. Ph.D. thesis, University of Rochester.

Zeno Vendler. 1957. Verbs and times. *The philosophical review* pages 143–160.

Ian H Witten, Eibe Frank, Len Trigg, Mark Hall, Geoffrey Holmes, and Sally Jo Cunningham. 1999. Weka: Practical machine learning tools and techniques with java implementations .

Stephen Wright and Jorge Nocedal. 1999. Numerical optimization. *Springer Science* 35:67–68.

Alessandra Zarcone and Alessandro Lenci. 2008. Computational models of event type classification in context. In *Proceedings of LREC2008*.

Learning Prototypical Event Structure from Photo Albums

Antoine Bosselut[†], **Jianfu Chen**[‡], **David Warren**[‡], **Hannaneh Hajishirzi**[†] and **Yejin Choi**[†]
[†]Computer Science & Engineering, University of Washington, Seattle, WA
{antoineb, hannaneh, yejin}@cs.washington.edu
[‡]Department of Computer Science, Stony Brook University, Stony Brook, NY
{jianchen, warren}@cs.stonybrook.edu

Abstract

Activities and events in our lives are structural, be it a vacation, a camping trip, or a wedding. While individual details vary, there are characteristic patterns that are specific to each of these scenarios. For example, a wedding typically consists of a sequence of events such as walking down the aisle, exchanging vows, and dancing. In this paper, we present a data-driven approach to learning event knowledge from a large collection of photo albums. We formulate the task as constrained optimization to induce the prototypical temporal structure of an event, integrating both visual and textual cues. Comprehensive evaluation demonstrates that it is possible to learn multimodal knowledge of event structure from noisy web content.

1 Introduction

Many common scenarios in our lives, such as a wedding or a camping trip, show characteristic structural patterns. As illustrated in Figure 1, these patterns can be sequential, such as in a wedding, where exchanging vows generally happens before cutting the cake. In other scenarios, there may be a set of composing events, but no prominent temporal relations. A camping trip, for example, might include events such as hiking, which can happen either before or after setting up a tent.

This observation on the prototypical patterns in everyday scenarios goes back to early artificial intelligence research. Scripts (Schank and Abelson, 1975), an early formalism, were developed to encode the necessary background knowledge to support an inference engine for common sense reasoning in limited domains. However, early approaches based on hand-coded symbolic representations proved to be brittle and difficult to scale.

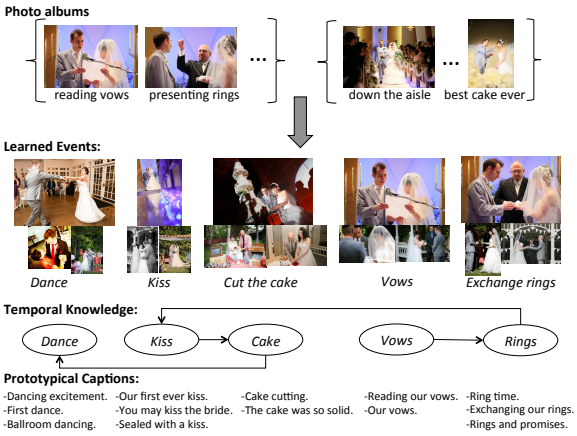


Figure 1: We collect photo albums of common scenarios (e.g., weddings) and cluster their images and captions to learn the hierarchical events that make up these scenarios. We use constrained optimization to decode the temporal order of these events, and we extract the prototypical descriptions that define them.

An alternative direction in recent years has been statistical knowledge induction, i.e., learning script or common sense knowledge bottom-up from large-scale data. While most prior work is based on text (Pichotta and Mooney, 2014; Jans et al., 2012; Chambers and Jurafsky, 2008; Chambers, 2013), recent work begins exploring the use of images as well (Bagherinezhad et al., 2016; Vedantam et al., 2015).

In this paper, we present the first study for learning knowledge about common life scenarios (e.g., weddings, camping trips) from a large collection of online photo albums with time-stamped images and their captions. The resulting dataset includes 34,818 time-stamped photo albums corresponding to 12 distinct event scenarios with 1.5 million images and captions (see Table 1 for more details).

We cast unsupervised learning of event structure as a sequential multimodal clustering prob-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1769–1779,
Berlin, Germany, August 7-12, 2016.

lem, which requires solving two subproblems concurrently: identifying the boundaries of events and assigning identities to each of these events. We formulate this process as constrained optimization, where constraints encode the temporal event patterns that are induced directly from the data. The outcome is a statistically induced prototypical structure of events characterized by their visual and textual representations.

We evaluate the quality and utility of the learned knowledge in three tasks: temporal event ordering, segmentation prediction, and multimodal summarization. Our experimental results show the performance of our model in predicting the order of photos in albums, partitioning photo albums into event sequences, and summarizing albums.

2 Overview

The high-level goal of this work is unsupervised induction of the prototypical event structure of common scenarios from multimodal data. We assume a two-level structure: high-level events, which we refer to as *scenarios* (e.g., wedding, funeral), are given, and low-level events (e.g., dance, kiss, vows), which we refer to as *events*, are to be automatically induced. In this section, we provide the overview of the paper (Section 2.1), and introduce our new dataset (Section 2.2).

2.1 Approach

Given a large collection of photo albums corresponding to a scenario, we want to learn three aspects of event knowledge by (1) identifying events common to the given scenario (Section 4.1), (2) learning temporal relations across events (Section 4.2), and (3) extracting prototypical captions for each event (Section 4.3).

To induce the prototypical event structure, an important subproblem we consider is individual photo album analysis, where the task is (1) partitioning each photo album into a sequence of segments, and (2) assigning the event identity to each segment. We present an inference model based on Integer Linear Programming (ILP) in Section 3 to perform both segmentation and event identification simultaneously, in consideration of the learned knowledge that we describe in Section 4.

Finally, we evaluate the utility of the automatically induced knowledge in the context of three concrete tasks: temporal ordering of photos (Section 6.1), album segmentation (Section 6.2), and photo album summarization (Section 6.3).

scenario	# of albums	# of images
WEDDING	4689	192K
MARATHON	3961	158K
COOKING	1168	36K
FUNERAL	781	28K
BARBECUE	735	22K
BABY BIRTH	688	21K
PARIS TRIP	4603	306K
NEW YORK TRIP	4205	267K
CAMPING	4063	159K
THANKSGIVING	5928	153K
CHRISTMAS	3449	98K
INDEPENDENCE DAY	548	22K
TOTAL	34,818	1.5M

Table 1: Dataset Statistics: the number of albums and images compiled for each scenario. The middle horizontal line separates the scenarios we predict have a well-defined temporal structure (top) from those we predict do not (bottom).

2.2 Dataset

For this study, we have compiled a new corpus of multimodal photo albums across 12 distinct scenarios. It comprises of 34,818 albums containing 1.5 million pairs of online photographs and their textual descriptions. Table 1 shows the list of scenarios and the corresponding data statistics. We choose six scenarios (the top half of Table 1) that we expect have an inherent temporal event structure that can be learned and six that we expect do not (the bottom half of Table 1).

The dataset is collected using the Flickr API[1,2]. We use the scenario names and variations of them (e.g., Paris Trip, Paris Vacation) as queries for images. We then form albums from these images by grouping images by user, sorting them by timestamp, and extracting groups that are within a contained time frame (e.g., 24 hours for a wedding, 5 days for a trip). For all images, we extract the first sentences of the corresponding textual descriptions as captions and also store their timestamps. This data is publicly available at `https://www.cs.washington.edu/projects/nlp/protoevents`.

3 Inference Model for Multimodal Event Segmentation and Identification

Given a photo album, the goal of the inference is to assign events to photos and to segment albums by event. More formally, given a sequence of M photos $P = \{p_1, \ldots, p_M\}$, and N learned events $E = \{e_1, \ldots, e_N\}$, the task is to assign each photo to a

[1]https://www.flickr.com/services/api/

[2]https://pypi.python.org/pypi/flickrapi/1.4.5

Figure 2: The events learned in Section 4 are assigned to photos based on textual (A^c) and visual (A^v) affinities, which encode how well a photo represents an event (ϕ_{event}). Segmentation scores (ϕ_{seg}) between adjacent photos encourage similar photos to be assigned the same event. Local transition, P_L, and global pairwise ordering, P_G, probabilities encode the learned temporal knowledge between events. $\phi_{temporal}$ encourages event assignments toward a learned temporal structure of the scenario.

single event. The event assignment can be viewed as a latent variable for each photo. We formulate a constrained optimization (depicted in Figure 2) that maximizes the objective function, F, which consists of three scoring components: (a) event assignment scores ϕ_{event} (Section 3.1), (b) segmentation scores ϕ_{seg} (Section 3.2), and (c) temporal knowledge scores $\phi_{temporal}$ (Section 3.3):

$$F = \phi_{event} + \phi_{seg} + \phi_{temporal} \quad (1)$$

Decision Variables. The binary decision variable $\mathbf{X}_{i,k}$ indicates that photo p_i is assigned to event e_k. The binary decision variable $\mathbf{Z}_{i,j,k,l}$ indicates that photos p_i and p_j are assigned to events e_k and e_l, respectively:

$$\mathbf{Z}_{i,j,k,l} := \mathbf{X}_{i,k} \wedge \mathbf{X}_{j,l} \quad (2)$$

3.1 Event Assignment Scores

Event assignment scores quantify the textual and visual affinity between a photo p_i and an event e_k. Affinities are measures of representation similarity between photos and events. These scores push photos displaying a certain event to be assigned to that event. For now we assume the textual affinity matrix $\mathbf{A}^c \in [0,1]^{M\times N}$ and the visual affinity matrix $\mathbf{A}^v \in [0,1]^{M\times N}$ are given. We describe how we obtain these affinity matrices in Section 4.1. Event assignment scores are defined as the weighted sum of both textual and visual affinity:

$$\phi_{event} = \sum_{i=1}^{M}\sum_{k=1}^{N}\left(\gamma_{ce}\mathbf{A}^c_{i,k} + \gamma_{ve}\mathbf{A}^v_{i,k}\right)\mathbf{X}_{i,k} \quad (3)$$

where $\mathbf{X}_{i,k}$ is a photo-event assignment decision variable, and γ_{ce} and γ_{ve} are hyperparameters that balance the contribution of both affinities.

3.2 Segmentation Scores

Segmentation scores quantify textual and visual similarities between adjacent photos. These scores encourage similar adjacent photos to be assigned to the same event. We define a similarity score between adjacent photos equal to the weighted sum of their textual ($\mathbf{b}^c$) and visual ($\mathbf{b}^v$) similarities:

$$\phi_{seg} = \sum_{i=1}^{M-1}\sum_{k=1}^{N}\left(\gamma_{cs}\mathbf{b}^c_i + \gamma_{vs}\mathbf{b}^v_i\right)\mathbf{Z}_{i,i+1,k,k} \quad (4)$$

where $\mathbf{b}^c$, $\mathbf{b}^v \in [0,1]^{(M-1)\times 1}$ are vectors of textual and visual similarity scores between adjacent photos whose i^{th} element corresponds to the similarity score between photos p_i and p_{i+1}, $\mathbf{Z}$ is a decision variable defined by Equation 2, and γ_{cs} and γ_{vs} are hyperparameters balancing the contribution of both types of similarity. The similarity scores in the $\mathbf{b}$ vectors are computed using cosine similarity of the feature representations of adjacent images in both the textual and visual modes.

3.3 Temporal Knowledge Scores

Temporal knowledge scores quantify the compatibilities across different event assignments in terms of their relative ordering. For now, we assume two types of temporal knowledge matrices are given: $\mathbf{L} \in [0,1]^{N\times N}$ which stores local transition probabilities for every pair of events, e_k and e_l, and $\mathbf{G} \in [0,1]^{N\times N}$ which stores global pairwise ordering probabilities for every pair of events, e_k

and e_l. We describe how we obtain these temporal knowledge matrices in Section 4.2. The temporal knowledge score, defined below, encourages the inference model to assign events that are compatible with the learned temporal knowledge:

$$\phi_{temporal} = \gamma_{lp} \sum_{i=0}^{M} \sum_{k,l=1}^{N} \mathbf{L}_{k,l} \mathbf{Z}_{i,i+1,k,l} \quad (5)$$
$$+ \gamma_{gp} \sum_{i=1}^{M} \sum_{j=i}^{M} \sum_{k,l=1}^{N} \mathbf{G}_{k,l} \mathbf{Z}_{i,j,k,l}$$

where $\mathbf{Z}$ is a decision variable defined by Equation 2, and γ_{lp} and γ_{gp} are hyperparameters that balance the contribution of local and global temporal knowledge in the objective.

3.4 Constraints

We include hard constraints that force each photo to be assigned to exactly one event:

$$\sum_{k=1}^{N} \mathbf{X}_{i,k} = 1 \quad (6)$$

The number of these constraints is linear in the number of photos in an album. We also include hard constraints to ensure consistencies among binary decision variables $\mathbf{X}$ and $\mathbf{Z}$:

$$\frac{1}{2}(\mathbf{X}_{i,k} + \mathbf{X}_{j,l}) - \mathbf{Z}_{i,j,k,l} \geq 0 \quad (7)$$

which states that $\mathbf{Z}_{i,j,k,l}$ can be 1 only if both $\mathbf{X}_{i,k}$ and $\mathbf{X}_{j,l}$ are 1. The number of constraints for segmentation scores and local transition probabilities is $O(MN^2)$ because they model interactions between adjacent photos for all event pairs. The number of these constraints for global pairwise ordering probabilities is $O(M^2N^2)$ because they model interactions between all pairs of photos in an album for all event pairs.

4 Learned Event Knowledge

We learn base events for each scenario by clustering photos from training albums related to that scenario (Figure 3). As described in Section 3, these base events and their temporal knowledge are incorporated in a joint model for event induction in unseen albums.

4.1 Learned Event Representation

We perform k-means clustering over captions to create a base event model. We perform text-only clustering at first since visual cues are significantly noisier. Because not all photos have informative captions, it is expected that this base clustering will form meaningful clusters only over a subset of the data. For each scenario, the largest cluster corresponds to the "miscellaneous" cluster as the captions in it tend to be relatively uninformative about specific events. This cluster is excluded when computing temporal knowledge probabilities (Section 4.2).

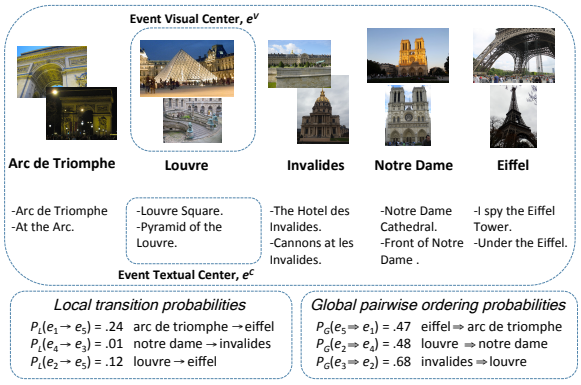


Figure 3: Photos are clustered by their captions. We can compute the visual, e_k^v, and caption, e_k^c, centers for all the clusters, as well as the local transition, P_L, and global pairwise ordering, P_G, probabilities between these events based on the sequential patterns they exhibit in the training set.

The visual and textual representations of an event are computed using the average of the visual and textual features, respectively, of photos assigned to that event. We compute each textual affinity $\mathbf{A}_{i,k}^c$ in the event assignment scores (Equation 3) as the cosine similarity between the textual features of the caption for photo p_i and the textual representation of event e_k. For textual features, we extract noun and verb unigrams using TurboTagger (Martins et al., 2013) and weigh them by their discriminativeness relative to their scenario, $P(S|w)$. Given scenario S and word w, $P(S|w)$ is defined as the number of albums for the scenario the word occurs in divided by the total number of albums in that scenario. The visual affinity $\mathbf{A}_{i,k}^v$ is the similarity between the visual features of photo p_i and the visual representation of event e_k. For visual features, we use the convolutional features from the final layer activations of the 16-layer VGGNet model (Simonyan and Zisserman, 2015).

4.2 Temporal Knowledge

Local transition probabilities. These probabilities, denoted as P_L, encode an expected sequence of events using temporal patterns among adjacent

Wedding		Camping		Funeral	
aisle	Walking down the aisle Bride walking down the aisle	*tent*	Inside our tent Setting up the tent	*service*	Graveside service The service
vow	Exchanging vows Reading the vows Reciting vows to each other	*fire*	Building the Fire Getting the Fire going Around the Fire	*pay respect*	Paying Respects Respect
dance	First Dance Everybody Dancing Dancing the Night Away	*sunset*	Watching the Sunset Sunset from camp Sunset on the first night	*goodbye*	Saying Goodbye

Table 2: Sample learned events and prototypical captions

photos. We model P_L for each pair of events as a multinomial distribution,

$$P_L(e_k \rightarrow e_l) = \frac{C(e_k \rightarrow e_l)}{\sum_{m=1}^{N} C(e_k \rightarrow e_m)} \quad (8)$$

where C is the observed counts of that specific event transition. This is the likelihood that an event e_k is immediately followed by event e_l.

Global pairwise ordering probabilities. These probabilities, denoted as P_G, encode global structural patterns about events. We model P_G for each pair of events as a binomial distribution by computing the likelihood that an event occurs before another at any point in an album,

$$P_G(e_k \Rightarrow e_l) = \frac{C(e_k \Rightarrow e_l)}{C(e_k \Rightarrow e_l) + C(e_l \Rightarrow e_k)} \quad (9)$$

where $C(e_k \Rightarrow e_l)$ is the observed counts of e_k occurring anytime before e_l in all photo albums. These global probabilities model relations among events assigned to all photos in the album, not just events assigned to photos that are adjacent to one another. This distinction is important because these probabilities can encode global patterns between events and are not limited to modeling a sequential event chain.

We use these learned temporal probabilities, P_L and P_G, in matrices $\mathbf{L}$ and $\mathbf{G}$ from $\phi_{temporal}$ (Equation 5). These matrices are used to index local transition probabilities and global pairwise ordering probabilities for pairs of events when computing temporal knowledge scores in the inference model (Section 3.3).

4.3 Prototypical Captions

After clustering the photos, the representative language of the captions in each cluster begins to tell a story about each scenario. The event names are automatically extracted using the most common content words among captions in the cluster. For each cluster, we also compile *prototypical captions* by extracting captions whose lemmatized forms are frequently observed throughout multiple albums in the scenario. Sample events and their prototypical captions from three scenarios are displayed in Table 2.

5 Experimental Setup

Data split. For scenarios with more than 1000 albums, we use 100 albums for each of the development and test sets and use the rest for training. For scenarios with less than 1000 albums, we use 50 albums for each of the development and test sets, and the rest for training.

Implementation details. We optimize our objective function using integer linear programming (Roth and Yih, 2004) with the Gurobi solver (Inc., 2015). For computational efficiency, temporally close sets of consecutive photos are treated as one unit during the optimization. We use these units to reduce the number of variables and constraints in the model from a function of the number of photos to a function of the number of units. We form these units heuristically by merging images agglomeratively when their timestamps are within a certain range of the closest image in a unit. When merging photos, the textual affinity of each unit for a particular event is the maximum affinity for that event among photos in that unit. The visual affinity of each unit is the average of all affinities for that event among photos in that unit. The textual and visual similarities of consecutive units are defined in terms of the similarities between the two photos at the units' boundary. Temporal information for events not aligned well with a particular unit should not influence the objective, so we include temporal scores only for unit-event pairs which have both textual and visual event assignment scores greater than 0.05.

Hyperparameters. We tune the hyperparameters using grid search on the development set. In models where the corresponding objective components are included, we set $\gamma_{ce} = 1$, $\gamma_{ve} = 1$, $\gamma_{cs} = .5$, $\gamma_{vs} = .15$, $\gamma_{lp} = 1$, and $\gamma_{gp} = \frac{4}{Q}$ (where Q is the

Model	Wedding	Baby Birth	Marathon	Cooking	Funeral	Barbecue	Indep. Day	Camping	Thanksgiving	Paris Trip	NY Trip	Christmas
k-MEANS	52.7	52.5	53.8	53.0	50.5	50.2	53.2	52.3	51.4	**53.1**	**50.3**	51.4
NO TEMPORAL	58.6	66.3	62.6	56.5	50.8	51.7	**58.0**	52.6	54.3	51.7	49.1	50.9
FULL MODEL	**60.0**	**66.5**	**64.5**	**63.2**	**53.1**	**58.6**	56.0	**55.5**	**56.1**	52.3	48.5	**52.4**

Table 3: Temporal ordering pairwise photo results. The metric reported is accuracy, the percentage of time the correct photo is picked as coming first based on the event assigned to it. Scenarios with an expected temporal structure are in the left half of the table.

number of event units). For k-means clustering, we use 10 random restarts and 40 cluster centers for the WEDDING, CAMPING, PARIS TRIP, and NY TRIP scenarios. For all other scenarios, we use 30 cluster centers.

6 Experimental Results

We evaluate the performance of our model on three tasks. The first task evaluates the effect of learned temporal knowledge in predicting the correct order of photos in an unseen album (Section 6.1). The second task evaluates the model's ability to segment albums into logical groupings (Section 6.2). The third task evaluates the quality of prototypical captions and their use in photo album summarization (Section 6.3).

6.1 Temporal Ordering of Photos

We evaluate the model's ability to capture the temporal relationships between events in the scenario. Given two randomly selected photos p_i and p_j from an album, the task is to predict which of the photos appears earlier in the album using their event assignments. We compare the full model that assigns events to photos using ILP (Section 3) with two baselines: k-MEANS , which assigns events to photos using k-means clustering over captions (Section 4), and NO TEMPORAL: a variant of the full model that does not use temporal knowledge scores ($\phi_{temporal}$ in Equation 1) for optimization.

We run each method over a test photo album, in which the events e_k and e_l are assigned to the photos p_i and p_j, respectively. We then use the learned global pairwise ordering probabilities (Section 4.2) to predict which photo appears earlier in the album. We report the accuracy of each method in predicting the order of photos compared to the actual order of photos in the albums. We perform this experiment 50 times for each album and average the number of correct choices across every album and every trial.

Results. Table 3 reports the results of the full model compared to the baselines. The results show that temporal knowledge generally helps in predicting photo ordering. We observe that the full model achieves higher scores for scenarios for which we expect would have a sequential structure (e.g., WEDDING, BABY BIRTH, MARATHON). Conversely, the full model achieves lower overall scores in non-sequential scenarios (e.g., PARIS TRIP, NEW YORK TRIP). Qualitatively, we notice interesting temporal patterns such as the fact that during a marathon, the starting line occurs before the medal awards with 92.3% probability, or that Parisian tourists have a 24% chance ($\sim$10$\times$ higher than random chance) of visiting the Eiffel Tower immediately after the Arc de Triomphe (a high local transition probability that correctly implies their real world proximity).

6.2 Album Segmentation

Our model partitions photos in albums into coherent events. The album segmentation evaluation tests if the model recovers the same sequences of photos that a human would identify in a photo album as events.

Evaluation. We had an impartial annotator label where they thought events began and ended in 10 candidate albums of greater than 100 photos for three scenarios: WEDDING, FUNERAL, CAMPING. We evaluate how well our model can replicate these boundaries with two metrics. The first metric is the F_1 score of recovering the same boundaries annotated by humans. The second metric is d, the difference between the number of events segmented by the model compared to the annotated albums. We report results for exact event boundaries as well as relaxed boundaries where the start of an event can be r photos away from the start of an annotated event, where r is the relaxation coefficient. For reference, we note that albums in the wedding scenario were dual annotated and the agreement between annotators is 56.9% for $r = 0$ and 77.5% for $r = 2$.

Results. Table 4 shows comparison of the the full ILP model with same baselines we described before, k-MEANS and NO TEMPORAL. The table shows that the full model generally outperforms the k-MEANS baseline for all three scenarios.

In the WEDDING scenario, the F_1 score for the full

Model	r	WEDDING		FUNERAL		CAMPING	
		F_1	d	F_1	d	F_1	d
k-MEANS		27.1	32.9	27.9	29.6	**31.2**	46.0
NO TEMPORAL	0	32.0	-5.6	**35.9**	**.9**	22.0	-15.6
FULL MODEL		**37.8**	**1.3**	32.2	4.2	27.5	**-10.1**
k-MEANS		40.8	32.9	38.3	29.6	46.2	46.0
NO TEMPORAL	2	49.6	-5.6	**57.6**	**.9**	35.4	-15.8
FULL MODEL		**57.5**	**1.3**	51.6	5.0	**51.4**	**-10.1**

Table 4: Segmentation results for our full model. F_1 scores how often our model recovers the same boundaries annotated by humans. d is the average difference between the number of events identified by the model in an album and marked by annotators. r is the relaxation coefficient.

Feature Group Excluded	P	R	F_1	d
FULL MODEL	36.7	**42.8**	**37.8**	1.3
- Visual Event Affinity	37.7	37.0	35.3	-1.7
- Textual Segmentation	37.1	41.5	37.4	.8
- Visual Segmentation	35.1	42.1	36.5	1.7
- Local Ordering Probs.	36.9	40.3	36.9	**.2**
- Global Ordering Probs.	**40.5**	25.0	29.5	-5.8

Table 5: Ablation study of objective function components for the wedding scenario. P, R, and F_1 are the precision, recall and F-measure of recovering the same boundaries annotated by humans. d is the average difference between the number of events identified by our models and the annotators.

model is consistently higher. The k-MEANS baseline oversamples the number of events in albums, which is indicated by an average d significantly greater than 0. For the FUNERAL scenario, the NO TEMPORAL baseline outperforms the full model. We attribute this difference to the smaller data subset (see Table 1) making it harder to learn the temporal relations in the scenario, which makes the contributions of the local and global temporal probabilities unexpected. In the CAMPING scenario, the F_1 score for the k-MEANS baseline is higher than that of the full model when $r = 0$. At a high-level, CAMPING is a scenario we expect has less of a known structure compared to other scenarios and may be harder to segment into its events.

Ablation Study. Table 5 depicts the performance of ablations of the full model for the wedding scenario. Results show that removing any component of the objective functions yields lower recall and F_1 scores than the full model for $r = 0$. The exception is removing local ordering probabilities, which yields a higher d. These observations support the hypothesis that all of the components of the objective function contribute to segmenting the album into subsequences of photos depicting the same event. Particularly, we note the degradation when removing the global ordering probabilities, indicating that approaches which model only local event transitions such as hidden Markov models would not be suitable for this task.

6.3 Photo Album Summarization

The final experiment evaluates how our learned prototypical captions can improve downstream tasks such as summarization and captioning.

6.3.1 Summaries

The goal of a good summary is to select the most salient pictures of an album. In our setting, a good summary should have a high coverage of the events in an album and choose the photos that most appropriately depict these events. Given a photo budget b, we choose a subset of photos that aims for these goals. To summarize a test album, we run our model over the entire album. This will yield h unique events assigned to the photos in the album. For each of these h events, we choose the photo with the highest event assignment score for that event (Equation 3) to be in the summary. If $h > b$, we count the number of photos in the training set assigned to each of the h events and choose the photos corresponding to the b events with the largest membership of photos in the training set. If $h < b$, we complete the summary with $b - h$ photos from the "miscellaneous" event that are spaced evenly throughout the album. Finally, we replace the caption of each selected photo with a prototypical caption (Section 4.3) for the assigned event.

Baseline. We evaluate against two baselines. The first baseline, KTH, involves including a photo in the summary every $k = M/b$ photos. The second baseline, k-MEANS, uses the events assigned to photos from k-means clustering and then picks b photos in the same manner as our main model.

Evaluation. We evaluate the summaries produced by each method with a human evaluation using Amazon Mechanical Turk (AMT). We use albums from the test set that contain more than 40 photos for the wedding scenario. For each album, at random, we present two summaries generated by two algorithms. AMT workers are instructed to choose the better summary considering both the images and the captions. For each comparison of two summaries for an album, we aggregate answers from three workers by majority voting. We set $b = 7$. The number of assigned events in an album, h, varies by album.

Results. As seen in Table 6, the summary from the full model is preferred 57.7% of the time compared to the KTH baseline. The summaries generated using the full model perform slightly better than the summaries from k-MEANS. We attribute

Figure 4: Example summaries from the wedding, Paris trip, and baby birth scenarios. In cases where the album had less events than b, the additionally chosen photos are outlined in red. These photos do not have their caption replaced by a prototypical captions and merely fill out the summary.

Method	Selection Rates	
FULL MODEL vs. KTH	**57.7**	42.3
FULL MODEL vs. k-MEANS	**53.8**	47.2
k-MEANS vs. KTH	**53.8**	47.2

Table 6: Summarization results. The selection rates indicate the percentage of time the corresponding method in the left-most column was picked.

Method	Scenario Relevance	Image Relevance	Grammar
LSTM	**4.90**	2.85	3.74
FULL MODEL	4.55	**3.66**	**4.08**
RAW CAPTIONS	4.10	4.36	4.28

Table 7: Captioning results. We evaluate the caption quality of the prototypical captions of the full model, those generated by an LSTM trained on the raw captions, and original captions. Captions were evaluated on 3 metrics: grammatical correctness, how relevant they were to the scenario, and how relevant they were to their assigned image.

the superior performance of the full model to the fact that it redistributes photos with noisy captions throughout the events, allowing for a larger sample to estimate visual representations of events, yielding more accurate visual affinity measurements to choose the summarization photos. As can be seen from qualitative examples in Figure 4, the photos chosen and the captions assigned cover key events that would occur during the scenario and describe them in a coherent way. Additional examples are available at `https://www.cs.washington.edu/projects/nlp/protoevents`.

6.3.2 Prototypical Captions

We also evaluate the quality of the prototypical captions assigned to every photo in the summaries. For each album, we use the same sets of b photos from the full model in the summarization task and evaluate the quality of the prototypical captions paired with that group of photos.

Evaluation. We evaluate the quality of captions assigned to every photo by asking AMT workers to rate the captions on three different metrics: grammaticality, relevance to the scenario to which the image belongs, and relevance to its paired image. Five AMT workers rate each group of b photos on a five point Likert scale for each metric. We compare the prototypical captions for every photo in the summary with captions generated by an LSTM model[3] trained on every photo-caption pair in the training set for a scenario. We also compare with the original raw captions for each image in the summary. Because we chose photos with the highest event assignment scores (Equation 3) to be in the summary, the raw captions for this evaluation are cleaner and more descriptive than most captions in the dataset.

Results. Our model outperforms the LSTM-generated captions in the image relevance and grammaticality scores, but did worse in scenario

[3]We use a single-layer encoder-decoder LSTM. The cell state and the input embedding dimensions are 256. Visual inputs are the final layer convolutional features of the VGG-16 model and are fine-tuned during training. We use RMSprop to train the network with a base learning rate of .0001 and 30% dropout. We train the model for 45 epochs on a single NVIDIA Titan X GPU with mini batch size 100. To decode, we use beam search with beam size 5.

relevance. We attribute this result to LSTM captions having little caption variation because the model learns frequency statistics without any knowledge of latent events. Almost all LSTM captions mention the words *bride, wedding*, or *groom*, yielding a very high scenario score for the caption, even if that caption is grammatically incorrect or irrelevant to the image. As expected the raw captions have high relevance to the original image, and they are grammatical, but can be less relevant to the corresponding scenario.

7 Related Work

Previous studies have explored unsupervised induction of salient content structure in newswire texts (Barzilay and Lee, 2004), temporal graph representations (Bramsen et al., 2006), and storyline extraction and event summarization (Xu et al., 2013). Another line of research finds the common event structure from children's stories (McIntyre and Lapata, 2009), where the learned plot structure is used to stochastically generate new stories (Goyal et al., 2010; Goyal et al., 2013). Our work similarly aims to learn the typical temporal patterns and compositional elements that define common scenarios, but with multimodal integration.

Compared to studies that learn narrative schemas from natural language (Pichotta and Mooney, 2014; Jans et al., 2012; Chambers and Jurafsky, 2009; Chambers, 2013; Cassidy et al., 2014), or compile script knowledge from crowdsourcing (Regneri et al., 2010), our work explores a new source of knowledge that allows grounded event learning with temporal dimensions, resulting in a new dataset of scenario types that are not naturally accessible from newswire or literature.

While recent studies have explored videos and photo streams as a source of discovering complex events and learning their sequential patterns (Kim and Xing, 2014; Kim and Xing, 2013; Tang et al., 2012; Tschiatschek et al., 2014), their focus was mostly on the visual modality. Zhang et al. (2015) explored multimodal information extraction focusing specifically on identifying video clips that referred to the same event in television news. This contrasts to the goal of our study that aims to learn the temporal structure by which common scenarios unfold.

Integrating language and vision has attracted increasing attention in recent years across diverse tasks such as image captioning (Karpathy and Fei-Fei, 2015; Vinyals et al., 2015; Fang et al., 2015; Xu et al., 2015; Chen et al., 2015), cross modal semantic modeling (Lazaridou et al., 2015), information extraction (Morency et al., 2011; Rosas et al., 2013; Zhang et al., 2015; Izadinia et al., 2015), common-sense knowledge (Vedantam et al., 2015; Bagherinezhad et al., 2016), and visual storytelling (Huang et al., 2016). Our work is similar to both common sense knowledge learning and visual story completion. Our model learns commonsense knowledge on the hierarchical and temporal event structure from scenario-specific multimodal photo albums, which can be viewed as visual stories about common life events.

Recent work focused on photo album summarization using visual (Sadeghi et al., 2015) and multimodal representations (Sinha et al., 2011). Our work identifies the nature of common events in scenarios and learns their timelines and characteristic forms.

8 Conclusion

We introduce a novel exploration to learn script-like knowledge from photo albums. We model stochastic event structure to learn both the event representations (textual and visual) and the temporal relations among those events. Our event induction method incorporates learned knowledge about events, partitions photo albums into segments, and assigns events to those segments. We show the significance of our model in learning and using learned knowledge for photo ordering, album segmentation, and summarization. Finally, we provide a dataset depicting 12 scenarios with $\sim$1.5 M images for future research. Future directions could include exploring nuances in the type of temporal knowledge that can be learned across different scenarios.

Acknowledgements

We thank the anonymous reviewers for many insightful comments and members of UW NLP for feedback and support. The work is in part supported by NSF grants IIS-1408287, IIS-1524371, IIS-1616112, DARPA under the CwC program through the ARO (W911NF-15-1-0543), the Allen Institute for AI (66-9175), the Allen Distinguished Investigator Award, and gifts by Google and Facebook.

References

Hessam Bagherinezhad, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. 2016. Are elephants bigger than butterflies? reasoning about sizes of objects. In *Proceedings of the Conference in Artificial Intelligence (AAAI)*.

Regina Barzilay and Lillian Lee. 2004. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *NAACL-HLT*.

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing temporal graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 189–198. Association for Computational Linguistics.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An annotation framework for dense event ordering. In *ACL*.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *ACL*.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 602–610. Association for Computational Linguistics.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Jianfu Chen, Polina Kuznetsova, David S Warren, and Yejin Choi. 2015. Déja image-captions: A corpus of expressive descriptions in repetition. In *NAACL-HLT*.

Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K. Srivastava, Li Deng, Piotr Dollar, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2015. From captions to visual concepts and back. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Amit Goyal, Ellen Riloff, and Hal Daumé III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 77–86. Association for Computational Linguistics.

Amit Goyal, Ellen Riloff, et al. 2013. A computational model for plot units. *Computational Intelligence*, 29(3):466–488.

Ting-Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling. In *NAACL*.

Gurobi Optimization Inc. 2015. Gurobi optimizer reference manual.

Hamid Izadinia, Fereshteh Sadeghi, Santosh K Divvala, Hannaneh Hajishirzi, Yejin Choi, and Ali Farhadi. 2015. Segment-phrase table for semantic segmentation, visual entailment and paraphrasing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10–18.

Bram Jans, Steven Bethard, Ivan Vuli, and Marie-Francine Moens. 2012. Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 336–344.

Andrej Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Gunhee Kim and Eric P Xing. 2013. Jointly aligning and segmenting multiple web photo streams for the inference of collective photo storylines. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 620–627. IEEE.

Gunhee Kim and Eric Xing. 2014. Reconstructing storyline graphs for image recommendation from web community photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3882–3889.

Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. 2015. Combining language and vision with a multimodal skip-gram model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 153–163, Denver, Colorado, May–June. Association for Computational Linguistics.

André FT Martins, Miguel B Almeida, and Noah A Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *ACL*.

Neil McIntyre and Mirella Lapata. 2009. Learning to tell tales: A data-driven approach to story generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 217–225. Association for Computational Linguistics.

Louis-Philippe Morency, Rada Mihalcea, and Payal Doshi. 2011. Towards multimodal sentiment analysis: Harvesting opinions from the web. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 169–176. ACM.

Karl Pichotta and Raymond J Mooney. 2014. Statistical script learning with multi-argument events. In *EACL*, volume 14, pages 220–229.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988. Association for Computational Linguistics.

Verónica Pérez Rosas, Rada Mihalcea, and Louis-Philippe Morency. 2013. Multimodal sentiment analysis of spanish online videos. *IEEE Intelligent Systems*, (3):38–45.

Dan Roth and Wen-tau Yih. 2004. A linear programming formulation for global inference in natural language tasks. Technical report, DTIC Document.

Fereshteh Sadeghi, J Rafael Tena, Ali Farhadi, and Leonid Sigal. 2015. Learning to select and order vacation photographs. In *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*, pages 510–517. IEEE.

Roger C Schank and Robert P Abelson. 1975. *Scripts, plans, and knowledge*. Yale University.

K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Pinaki Sinha, Sharad Mehrotra, and Ramesh Jain. 2011. Summarization of personal photologs using multidimensional content and context. In *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*, page 4. ACM.

Kevin Tang, Li Fei-Fei, and Daphne Koller. 2012. Learning latent temporal structure for complex event detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1250–1257. IEEE.

Sebastian Tschiatschek, Rishabh K Iyer, Haochen Wei, and Jeff A Bilmes. 2014. Learning mixtures of submodular functions for image collection summarization. In *Advances in Neural Information Processing Systems*, pages 1413–1421.

R. Vedantam, X. Lin, T. Batra, C. L. Zitnick, and D. Parikh. 2015. Learning common sense through visual abstraction. In *Proceedings of the International Conference in Computer Vision (ICCV)*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Shize Xu, Shanshan Wang, and Yan Zhang. 2013. Summarizing complex events: a cross-modal solution of storylines extraction and reconstruction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1281–1291.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron C. Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 2048–2057.

Tongtao Zhang, Hongzhi Li, Heng Ji, and Shih-Fu Chang. 2015. Cross-document event coreference resolution based on cross-media features. In *Proc. Conference on Empirical Methods in Natural Language Processing*.

Cross-Lingual Image Caption Generation

Takashi Miyazaki*
Yahoo Japan Corporation
Tokyo, Japan
takmiyaz@yahoo-corp.jp

Nobuyuki Shimizu*
Yahoo Japan Corporation
Tokyo, Japan
nobushim@yahoo-corp.jp

Abstract

Automatically generating a natural language description of an image is a fundamental problem in artificial intelligence. This task involves both computer vision and natural language processing and is called "image caption generation." Research on image caption generation has typically focused on taking in an image and generating a caption in English as existing image caption corpora are mostly in English. The lack of corpora in languages other than English is an issue, especially for morphologically rich languages such as Japanese. There is thus a need for corpora sufficiently large for image captioning in other languages. We have developed a Japanese version of the MS COCO caption dataset and a generative model based on a deep recurrent architecture that takes in an image and uses this Japanese version of the dataset to generate a caption in Japanese. As the Japanese portion of the corpus is small, our model was designed to transfer the knowledge representation obtained from the English portion into the Japanese portion. Experiments showed that the resulting bilingual comparable corpus has better performance than a monolingual corpus, indicating that image understanding using a resource-rich language benefits a resource-poor language.

1 Introduction

Automatically generating image captions by describing the content of an image using natural language sentences is a challenging task. It is especially challenging for languages other than English due to the sparsity of annotated resources in the target language. A promising solution to this problem is to create a comparable corpus. To support the image caption generation task in Japanese, we have annotated images taken from the MS COCO caption dataset (Chen et al., 2015b) with Japanese captions. We call our corpus the "YJ Captions 26k Dataset." While the size of our dataset is comparatively large with 131,740 captions, it greatly trails the 1,026,459 captions in the MS COCO dataset. We were thus motivated to transfer the resources in English (source language) to Japanese and thereby improve image caption generation in Japanese (target language). In natural language processing, a task involving transferring information across languages is known as a cross-lingual natural language task, and well known tasks include cross-lingual sentiment analysis (Chen et al., 2015a), cross-lingual named entity recognition (Zirikly and Hagiwara, 2015), cross-lingual dependency parsing (Guo et al., 2015), and cross-lingual information retrieval (Funaki and Nakayama, 2015).

Existing work in the cross-lingual setting is usually formulated as follows. First, to overcome the language barrier, create a connection between the source and target languages, generally by using a dictionary or parallel corpus. Second, develop an appropriate knowledge transfer approach to leverage the annotated data from the source language for use in training a model in the target language, usually supervised or semi-supervised. These two steps typically amount to automatically generating and expanding the pseudo-training data for the target language by exploiting the knowledge obtained from the source language.

We propose a very simple approach to cross-lingual image caption generation: exploit the English corpus to improve the performance of image caption generation in another language. In this ap-

* Both authors contributed equally to this work.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1780–1790,
Berlin, Germany, August 7-12, 2016.

proach, no resources besides the images found in the corpus are used to connect the languages, and we consider our dataset to be a comparable corpus. Paired texts in a comparable corpus describe the same topic, in this case an image, but unlike a parallel corpus, the texts are not exact translations of each other. This unrestrictive setting enables the model to be used to create image caption resources in other languages. Moreover, this model scales better than creating a parallel corpus with exact translations of the descriptions.

Our transfer model is very simple. We start with a neural image caption model (Vinyals et al., 2015) and pretrain it using the English portion of the corpus. We then remove all of the trained neural network layers except for one crucial layer, the one closest to the vision system. Next we attach an untrained Japanese generation model and train it using the Japanese portion of the corpus. This results in improved generation in Japanese compared to using only the Japanese portion of the corpus. To the best of our knowledge, this is the first paper to address the problem of cross-lingual image caption generation.

Our contribution is twofold. First, we have created and plan to release the first ever significantly large corpus for image caption generation for the Japanese language, forming a comparable corpus with existing English datasets. Second, we have created a very simple model based on neural image caption generation for Japanese that can exploit the English portion of the dataset. Again, we are the first to report results in cross-lingual image caption generation, and our surprisingly simple method improves the evaluation metrics significantly. This method is well suited as a baseline for future work on cross-lingual image caption generation.

The paper is organized as follows. In the next section, we describe related work in image caption generation and list the corpora currently available for caption generation. Then in Section 3 we present the statistics for our corpus and explain how we obtained them. We then explain our model in Section 4 and present the results of our experimental evaluation in Section 5. We discuss the results in Section 6, and conclude in Section 7 with a summary of the key points.

2 Related Work

Recent advances in computer vision research have led to halving the error rate between 2012 and 2014 at the Large Scale Visual Recognition Challenge (Russakovsky et al., 2015), largely driven by the adoption of deep neural networks (Krizhevsky et al., 2012; Simonyan and Zisserman, 2014; Donahue et al., 2014; Sharif Razavian et al., 2014). Similarly, we have seen increased adaptation of deep neural networks for natural language processing. In particular, sequence-to-sequence training using recurrent neural networks has been successfully applied to machine translation (Cho et al., 2014; Bahdanau et al., 2015; Sutskever et al., 2014; Kalchbrenner and Blunsom, 2013).

These developments over the past few years have led to renewed interest in connecting vision and language. The encoder-decoder framework (Cho et al., 2014) inspired the development of many methods for generating image captions since generating an image caption is analogous to translating an image into a sentence.

Since 2014, many research groups have reported a significant improvement in image caption generation due to using a method that combines a convolutional neural network with a recurrent neural network. Vinyals et al. used a convolutional neural network (CNN) with inception modules for visual recognition and long short-term memory (LSTM) for language modeling (Vinyals et al., 2015). Xu et al. introduced an attention mechanism that aligns visual information and sentence generation for improving captions and understanding of model behavior (Xu et al., 2015). The interested reader can obtain further information elsewhere (Bernardi et al., 2016).

These developments were made possible due to a number of available corpora. The following is a list of available corpora that align images with crowd-sourced captions. A comprehensive list of other kinds of corpora connecting vision and language, e.g., visual question answering, is available elsewhere (Ferraro et al., 2015).

1. UIUC Pascal Dataset (Farhadi et al., 2010) includes 1,000 images with 5 sentences per image; probably one of the first datasets.
2. Abstract Scenes Dataset (Clipart) (Zitnick et al., 2013) contains 10,020 images of children playing outdoors associated with 60,396 descriptions.

3. Flickr 30K Images (Young et al., 2014) extends Flickr datasets (Rashtchian et al., 2010) and contains 31,783 images of people involved in everyday activities.
4. Microsoft COCO Dataset (MS COCO) (Lin et al., 2014; Chen et al., 2015b) includes about 328,000 images of complex everyday scenes with common objects in naturally occurring contexts. Each image is paired with five captions.
5. Japanese UIUC Pascal Dataset (Funaki and Nakayama, 2015) is a Japanese translation of the UIUC Pascal Dataset.

To the best of our knowledge, there are no large datasets for image caption generation except for English. With the release of the YJ Captions 26k dataset, we aim to remedy this situation and thereby expand the research horizon by exploiting the availability of bilingual image caption corpora.

3 Statistics for Data Set

In this section we describe the data statistics and how we gathered data for the YJ Captions 26k dataset. For images, we used the Microsoft COCO dataset (Chen et al., 2015b). The images in this dataset were gathered by searching for pairs of 80 object categories and various scene types on Flickr. They thus tended to contain multiple objects in their natural context. Objects in the scene were labeled using per-instance segmentations. This dataset contains pictures of 91 basic object types with 2.5 million labeled instances. To collect Japanese descriptions of the images, we used Yahoo! Crowdsourcing [1], a microtask crowdsourcing service operated by Yahoo Japan Corporation.

Given 26,500 images taken from the training part of the MS COCO dataset, we collected 131,740 captions in total. The images had on average 4.97 captions; the maximum number was 5 and the minimum was 3. On average, each caption had 23.23 Japanese characters. We plan to release the YJ Captions 26k dataset [2].

3.1 Crowdsourcing Procedure

Our captions were human generated using Yahoo! Crowdsourcing. As this crowdsourcing platform is operated in Japan, signing up for the service and participating require Japanese proficiency. Thus, we assumed that the participants were fluent in Japanese.

Figure 1: User Interface

First, we posted a pilot task that asked the participants to describe an image. We then examined the results and selected promising participants (comprising a "white list") for future task requests. That is, only the participants on the white list could see the next task. This selection process was repeated, and the final white list included about 600 participants. About 150 of them regularly participated in the actual image caption collection task. We modified the task request page and user interface on the basis of our experience with the pilot task. In order to prevent their fatigue, the tasks were given in small batches so that the participants were unable to work over long hours.

In our initial trials, we tried a direct translation of the instructions used in the MS-COCO English captions. This however did not produce Japanese captions comparable to those in English. This is because people describe what appears unfamiliar to them and do not describe things they take for granted. Our examination of the results from the pilot tasks revealed that the participants generally thought that the pictures contained non-Japanese people and foreign places since the images originated from Flickr and no scenery from Japan was included in the image dataset. When Japanese

[1] http://crowdsourcing.yahoo.co.jp
[2] http://research-lab.yahoo.co.jp/software/index.html

crowds are shown pictures with scenery in the US or Europe in MS-COCO dataset, the scenes themselves appear exotic and words such as 'foreign' and 'oversea' would be everywhere in the descriptions. As such words are not common in the original dataset, and to make the corpus nicer complement to the English dataset and to reduce the effects of such cultural bias, we modified the instructions: "2. Please give only factual statements"; "3. Please do not specify place names or nationalities." We also strengthened two sections in the task request page and added more examples.

The interface is shown in Figure 1. The instructions in the user interface can be translated into English as "Please explain the image using 16 or more Japanese characters. Write a single sentence as if you were writing an example sentence to be included in a textbook for learning Japanese. Describe all the important parts of the scene; do not describe unimportant details. Use correct punctuation. Write a single sentence, not multiple sentences or a phrase."

Potential participants are shown task request pages, and the participants select which crowdsourcing task(s) to perform. The task request page for our task had the following instructions (English translation):

1. Please explain an image using 16 or more Japanese characters. Please write a single sentence as if you were writing an example sentence to be included in a textbook for learning Japanese.
   (a) Do not use incorrect Japanese.
   (b) Use a polite style of speech (*desu*/*masu* style) as well as correct punctuation.
   (c) Write a single complete sentence that ends with a period. Do not write just a phrase or multiple sentences.
2. Please give only factual statements.
   (a) Do not write about things that might have happened or might happen in the future. Do not write about sounds.
   (b) Do not speculate. Do not write about something about which you feel uncertain.
   (c) Do not state your feelings about the scene in the picture. Do not use an overly poetic style.
   (d) Do not use a demonstrative pronoun such as 'this' or 'here.'
3. Please do not specify place names or nationalities.
   (a) Please do not give proper names.
4. Please describe all the important parts of the scene; do not describe unimportant details.

Together with the instructions, we provided 15 examples (1 good example; 14 bad examples).

Upon examining the collected data, manual checks of first 100 images containing 500 captions revealed that 9 captions were clearly bad, and 12 captions had minor problems in descriptions. In order to further improve the quality of the corpus, we crowdsourced a new data-cleaning task. We showed each participant an image and five captions that describe the image and asked to fix them.

The following is the instructions (English translation) for the task request page for our data-cleaning task.

1. There are five sentences about a hyper-linked image, and several sentences require fixes in order to satisfy the conditions below. Please fix the sentences, and while doing so, tick a checkbox of the item (condition) being fixed.
2. The conditions that require fixes are:
   (a) Please fix typographical errors, omissions and input-method-editor conversion misses.
   (b) Please remove or rephrase expressions such as 'oversea', 'foreign' and 'foreigner.'
   (c) Please remove or rephrase expressions such as 'image', 'picture' and 'photographed.'
   (d) Please fix the description if it does not match the contents of the image.
   (e) Please remove or rephrase subjective expressions and personal impressions.
   (f) If the statement is divided into several sentences, please make it one sentence.
   (g) If the sentence is in a question form, please make it a declarative sentence.
   (h) Please rewrite the entire sentence if meeting all above conditions requires extensive modifications.
   (i) If there are less than 16 characters, please provide additional descriptions so that the sentence will be longer than 16 characters.

For each condition, we provided a pair of examples (1 bad example and 1 fixed example).

To gather participants for the data-cleaning task, we crowdsourced a preliminary user qualification task that explained each condition requiring fixes in the first half, then quizzed the participants in the second half. This time we obtained over 900 qualified participants. We posted the data-cleaning task to these qualified participants.

The interface is shown in Figure 2. The instructions in the user interface are very similar to the task request page, except that we have an additional checkbox:

(j) All conditions are satisfied and no fixes were necessary.

We provided these checkboxes to be used as a checklist, so as to reduce failure by compensating for potential limits of participants' memory and attention, and to ensure consistency and completeness in carrying out the data-cleaning task.

For this data-cleaning task, we had 26,500 images totaling 132,500 captions checked by 267 participants. The number of fixed captions are

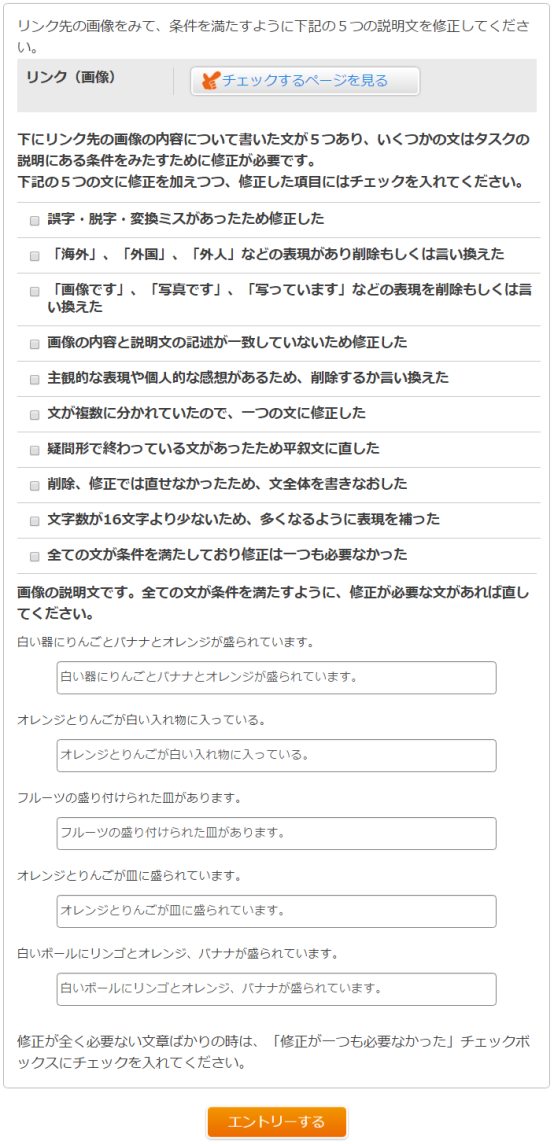


Figure 2: Data Cleaning Task User Interface

45,909. To our surprise, a relatively large portion of the captions were fixed by the participants. We suspect that in our data-cleaning task, the condition (e) was especially ambiguous for the participants, and they errored on the cautious side, fixing "a living room" to just "a room", thinking that a room that looks like a living room may not be a living room for the family who occupies the house, for example. Another example includes fixing "beautiful flowers" to just "flowers" because beauty is in the eye of the beholder and thought to be subjective. The percentage of the ticked checkboxes is as follows: (a) 27.2%, (b) 5.0%, (c) 12.3%, (d) 34.1%, (e) 28.4%, (f) 3.9%, (g) 0.3%, (h) 11.6%, (i) 18.5%, and (j) 24.0%. Note that a checkbox is ticked if there is at least one sentence out of five that meets the condition. In machine learning, this setting is called multiple-instance multiple-label problem (Zhou et al., 2012). We cannot directly infer how many captions correspond to a condition ticked by the participants.

After this data-cleaning task, we further removed a few more bad captions that came to our attention. The resulting corpus finally contains 131,740 captions as noted in the previous section.

4 Methodology

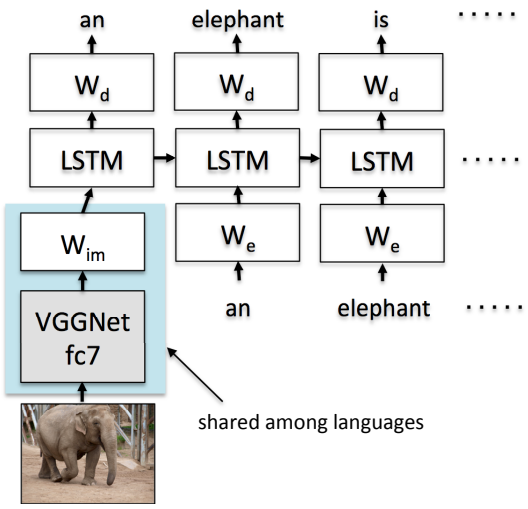


Figure 3: Model Overview

4.1 Model Overview

Figure 3 shows an overview of our model. Following the approach of Vinyals et al. (Vinyals et al., 2015), we used a discriminative model that maximizes the probability of the correct description given the image. Our model is formulated as

$$\theta^* = \arg\max_{\theta} \sum_{(I,S)} \sum_{t=0}^{N} \log p(S_t|I, S_0, ..., S_{t-1}; \theta), \quad (1)$$

where the first summation is over pairs of an image I and its correct transcription S. For the second summation, the sum is over all words S_t in S, and N is the length of S. θ represents the model parameters. Note that the second summation represents the probability of the sentence with respect to the joint probability of its words.

We modeled $p(S_t|I, S_0, ..., S_{t-1}; \theta)$ by using a recurrent neural network (RNN). To model the sequences in the RNN, we let a fixed length hidden state or memory h_t express the variable number of words to be conditioned up to $t-1$. The h_t

is updated after obtaining a new input x_t using a non-linear function f, so that $h_{t+1} = f(h_t, x_t)$. Since an LSTM network has state-of-the art performance in sequence modeling such as machine translation, we use one for f, which we explain in the next section.

A combination of LSTM and CNN are used to model $p(S_t|I, S_0, ..., S_{t-1}; \theta)$.

$$x_{-1} = W_{im}CNN(I) \quad (2)$$

$$x_t = W_e S_t, t \in \{0...N-1\} \quad (3)$$

$$p_{t+1} = Softmax(W_d LSTM(x_t)), \quad t \in \{0...N-1\} \quad (4)$$

where W_{im} is an image feature encoding matrix, W_e is a word embedding matrix, and W_d is a word decoding matrix.

4.2 LSTM-based Language Model

An LSTM is an RNN that addresses the vanishing and exploding gradients problem and that handles longer dependencies well. An LSTM has a memory cell and various gates to control the input, the output, and the memory behaviors. We use an LSTM with input gate i_t, input modulation gate g_t, output gate o_t, and forgetting gate f_t. The number of hidden units h_t is 256. At each time step t, the LSTM state c_t, h_t is as follows:

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \quad (5)$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \quad (6)$$

$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \quad (7)$$

$$g_t = \phi(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \quad (8)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \quad (9)$$

$$h_t = o_t \odot \phi(c_t), \quad (10)$$

where $\sigma(x) = (1 + e^{-x})^{-1}$ is a sigmoid function, $\phi(x) = (e^x - e^{-x})/(e^x + e^{-x})$ is a hyperbolic tangent function, and $\odot$ denotes the element-wise product of two vectors. W and b are parameters to be learned. From the values of the hidden units h_t, the probability distribution of words is calculated as

$$p_{t+1} = Softmax(W_d h_t). \quad (11)$$

We use a simple greedy search to generate captions as a sequence of words, and, at each time step t, the predicted word is obtained using $S_t = \arg\max_S p_t$.

4.3 Image Feature Extraction with Deep Convolutional Neural Network

The image recognition performance of deep convolutional neural network models has rapidly advanced in recent years, and they are now widely used for various image recognition tasks. We used a 16-layer VGGNet (Simonyan and Zisserman, 2014), which was a top performer at the ImageNet Large Scale Visual Recognition Challenge in 2014. A 16-layer VGGNet is composed of 13 convolutional layers having small 3x3 filter kernels and 3 fully connected layers. An image feature is extracted as a 4096-dimensional vector of the VGGNet's fc7 layer, which is the second fully connected layer from the output layer. VGGNet was pretrained using the ILSVRC2014 subset of the ImageNet dataset, and its weights were not updated through training.

4.4 Dataset Split

Because our caption dataset is annotated for only 26,500 images of the MS COCO training set, we reorganized the dataset split for our experiments. Training and validation set images of the MS COCO dataset were mixed and split into four blocks, and these blocks were assigned to training, validation, and testing as shown in Table 1. All blocks were used for the English caption dataset. Blocks B, C, and D were used for the Japanese caption dataset.

block	no. of images	split	language
A	96,787	train	En
B	22,500	train	En, Ja
C	2,000	val	En, Ja
D	2,000	test	En, Ja
total	123,287		

Table 1: Dataset Split

4.5 Training

The models were trained using minibatch stochastic gradient descent, and the gradients were computed by backpropagation through time. Parameter optimization was done using the RMSprop algorithm (Tieleman and Hinton, 2012) with an initial learning rate of 0.001, a decay rate of 0.999, and ϵ of 1.0^{-8}. Each image minibatch contained 100 image features, and the corresponding caption minibatch contained one sampled caption per image. To evaluate the effectiveness of Japanese

image caption generation, we used three learning schemes.

Monolingual learning This was the baseline method. The model had only one LSTM for Japanese caption generation, and only the Japanese caption corpus was used for training.

Alternate learning In this scheme, a model had two LSTMs, one for English and one for Japanese. The training batches for captions contained either English or Japanese, and the batches were fed into the model alternating between English and Japanese.

Transfer learning A model with one LSTM was trained completely for the English dataset. The trained LSTM was then removed, and another LSTM was added for Japanese caption generation. W_{im} was shared between the English and Japanese training.

These models were implemented using the Chainer neural network framework (Tokui et al., 2015). We consulted NeuralTalk (Karpathy, 2014), an open source implemenation of neural network based image caption generation system, for training parameters and dataset preprocessing. Training took about one day using NVIDIA TITAN X/Tesla M40 GPUs.

5 Evaluation

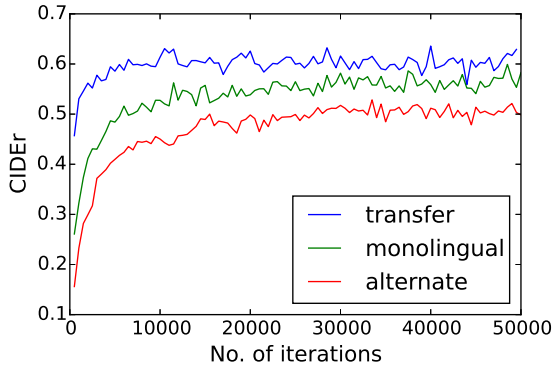


Figure 4: Learning Curve Represented by CIDEr Score

5.1 Evaluation Metrics

We used six standard metrics for evaluating the quality of the generated Japanese sentences: BLEU-1, BLEU-2, BLEU-3, BLEU-4 (Papineni et al., 2002), ROUGE-L (Lin, 2004), and CIDEr-D (Vedantam et al., 2014). We used the COCO caption evaluation tool (Chen et al., 2015b) to compute the metrics. BLEU (Papineni et al., 2002) was originally designed for automatic machine translation. By counting n-gram co-occurrences, it rates the quality of a translated sentence given several reference sentences. To apply BLEU, we considered that generating image captions is the same as translating images into sentences. ROUGE (Lin, 2004) is an evaluation metric designed by adapting BLEU to evaluate automatic text summarization algorithms. ROUGE is based on the longest common subsequences instead of n-grams. CIDEr (Vedantam et al., 2014) is a metric developed specifically for evaluating image captions. It measures consensus in image captions by performing a term-frequency inverse document frequency (TF-IDF) weighting for each n-gram. We used a robust variant of CIDEr called CIDEr-D. For all evaluation metrics, higher scores are better. In addition to these metrics, MS COCO caption evaluation (Chen et al., 2015b) uses METEOR (Lavie, 2014), another metric for evaluating automatic machine translation. Although METEOR is a good metric, it uses an English thesaurus. It was not used in our study due to the lack of a thesaurus for the Japanese language.

The CIDEr and METEOR metrics perform well in terms of correlation with human judgment (Bernardi et al., 2016). Although BLEU is unable to sufficiently discriminate between judgments, we report the BLEU figures as well since their use in literature is widespread. In the next section, we focus our analysis on CIDEr.

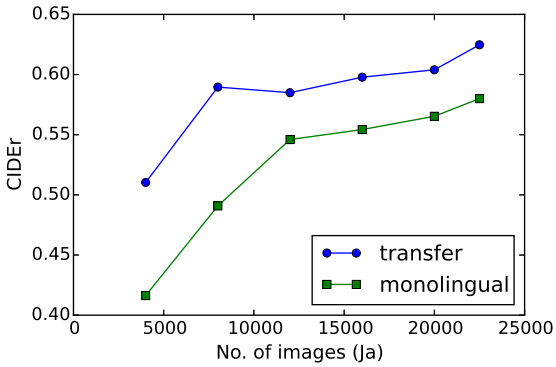


Figure 5: CIDEr Score vs. Japanese Data Set Size

5.2 Results

Table 2 shows the evaluation metrics for various settings of cross-lingual transfer learning. All values were calculated for Japanese captions gener-

	no. of images		metrics					
	En	Ja	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	CIDEr-D
monolingual	0	22,500	0.715	0.573	0.468	0.379	0.616	0.580
alternate	119,287	22,500	0.709	0.565	0.460	0.370	0.611	0.568
transfer	119,287	22,500	**0.717**	**0.574**	**0.469**	**0.380**	**0.619**	**0.625**

Table 2: Evaluation Metrics

ated for test set images. Our proposed model is labeled "transfer." As you can see, it outperformed the other two models for every metric. In particular, the CIDEr-D score was about 4% higher than that for the monolingual baseline. The performance of a model trained using the English and Japanese corpora alternately is shown on the line label "alternate." Surprisingly, this model had lower performance than the baseline model.

In Figure 4, we plot the learning curves represented by the CIDEr score for the Japanese captions generated for the validation set images. Transfer learning from English to Japanese converged faster than learning from the Japanese dataset or learning by training from both languages alternately. Figure 5 shows the relationship between the CIDEr score and the Japanese dataset size (number of images). The models pretrained using English captions (blue line) outperformed the ones trained using only Japanese captions for all training dataset sizes. As can be seen by comparing the case of 4,000 images with that of 20,000 images, the improvement due to cross-lingual transfer was larger when the Japanese dataset was smaller. These results show that pretraining the model with all available English captions is roughly equivalent to training the model with captions for 10,000 additional images in Japanese. This, in our case, nearly halves the cost of building the corpus.

Examples of machine-generated captions along with the crowd-written ground truth captions (English translations) are shown in Figure 6.

6 Discussion

Despite our initial belief, training by alternating English and Japanese input batch data for learning both languages did not work well for either language. As Japanese is a morphologically rich language and word ordering is subject-object-verb, it is one of most distant languages from English. We suspect that the alternating batch training interfered with learning the syntax of either language. Moreover, when we tried character-based models for both languages, the performance was significantly lower. This was not surprising because one word in English is roughly two characters in Japanese, and presumably differences in the language unit should affect performance. Perhaps not surprisingly, cross-lingual transfer was more effective when the resources in the target language are poor. Convergence was faster with the same amount of data in the target language when pretraining in the source language was done ahead of time. These two findings ease the burden of developing a large corpus in a resource poor language.

7 Conclusion

We have created an image caption dataset for the Japanese language by collecting 131,740 captions for 26,500 images using the Yahoo! Crowdsourcing service in Japan. We showed that pretraining a neural image caption model with the English portion of the corpus improves the performance of a Japanese caption generation model subsequently trained using Japanese data. Pretraining the model using the English captions of 119,287 images was roughly equivalent to training the model using the captions of 10,000 additional images in Japanese. This, in our case, nearly halves the cost of building a corpus. Since this performance gain is obtained without modifying the original monolingual image caption generator, the proposed model can serve as a strong baseline for future research in this area. We hope that our dataset and proposed method kick start studies on cross-lingual image caption generation and that many others follow our lead.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representation (ICLR)*.

Raffaella Bernardi, Ruket Cakici, Desmond Elliott, Aykut Erdem, Erkut Erdem, Nazli Ikizler-Cinbis, Frank Keller, Adrian Muscat, and Barbara Plank.

Figure 6: Image Caption Generation Examples

2016. Automatic description generation from images: A survey of models, datasets, and evaluation measures. *arXiv preprint arXiv:1601.03896*.

Qiang Chen, Wenjie Li, Yu Lei, Xule Liu, and Yanxiang He. 2015a. Learning to adapt credible knowledge in cross-lingual sentiment analysis. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 419–429, Beijing, China, July. Association for Computational Linguistics.

Xinlei Chen, Tsung-Yi Lin Hao Fang, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollr, and C. Lawrence Zitnick. 2015b. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference in Machine Learning (ICML)*.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 15–29, Berlin, Heidelberg. Springer-Verlag.

Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. 2015. A survey of current datasets for vision and language research. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 207–213, Lisbon, Portugal, September. Association for Computational Linguistics.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 585–590, Lisbon, Portugal, September. Association for Computational Linguistics.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1234–1244, Beijing, China, July. Association for Computational Linguistics.

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

Andrej Karpathy. 2014. Neuraltalk. `https://github.com/karpathy/neuraltalk`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Michael Denkowski Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. *ACL 2014*, page 376.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, 2014. *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, chapter Microsoft COCO: Common Objects in Context, pages 740–755. Springer International Publishing, Cham.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text summarization branches out: Proceedings of the ACL-04 workshop*, 8.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, pages 139–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June.

K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

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

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*.

Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. 2014. Cider: Consensus-based image description evaluation. *arXiv preprint arXiv:1411.5726*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *arXiv preprint arXiv:1502.03044*.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78.

Zhi-Hua Zhou, Min-Ling Zhang, Sheng-Jun Huang, and Yu-Feng Li. 2012. Multi-instance multi-label learning. *Artificial Intelligence*, 176(1):2291–2320.

Ayah Zirikly and Masato Hagiwara. 2015. Cross-lingual transfer of named entity recognizers without parallel corpora. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 390–396, Beijing, China, July. Association for Computational Linguistics.

C.L. Zitnick, D. Parikh, and L. Vanderwende. 2013. Learning the visual interpretation of sentences. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1681–1688, Dec.

Learning Concept Taxonomies from Multi-modal Data

Hao Zhang[1], **Zhiting Hu**[1], **Yuntian Deng**[1], **Mrinmaya Sachan**[1],
Zhicheng Yan[2], **Eric P. Xing**[1]
[1]Carnegie Mellon University, [2]UIUC
{hao,zhitingh,yuntiand,mrinmays,epxing}@cs.cmu.edu

Abstract

We study the problem of automatically building hypernym taxonomies from textual and visual data. Previous works in taxonomy induction generally ignore the increasingly prominent visual data, which encode important perceptual semantics. Instead, we propose a probabilistic model for taxonomy induction by jointly leveraging text and images. To avoid hand-crafted feature engineering, we design end-to-end features based on distributed representations of images and words. The model is discriminatively trained given a small set of existing ontologies and is capable of building full taxonomies from scratch for a collection of unseen conceptual label items with associated images. We evaluate our model and features on the WordNet hierarchies, where our system outperforms previous approaches by a large gap.

1 Introduction

Human knowledge is naturally organized as semantic hierarchies. For example, in WordNet (Miller, 1995), specific concepts are categorized and assigned to more general ones, leading to a semantic hierarchical structure (a.k.a taxonomy). A variety of NLP tasks, such as question answering (Harabagiu et al., 2003), document clustering (Hotho et al., 2002) and text generation (Biran and McKeown, 2013) can benefit from the conceptual relationship present in these hierarchies.

Traditional methods of manually constructing taxonomies by experts (e.g. WordNet) and interest communities (e.g. Wikipedia) are either knowledge or time intensive, and the results have limited coverage. Therefore, automatic induction of taxonomies is drawing increasing attention in both

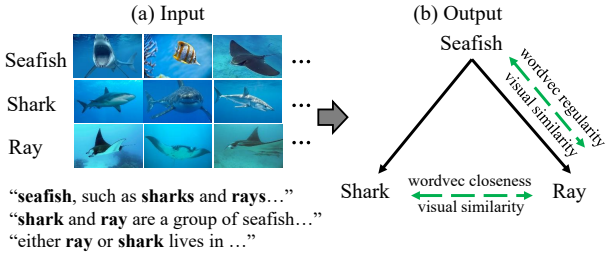


Figure 1: An overview of our system. (a) Input: a collection of label items, represented by text and images; (b) Output: we build a taxonomy from scratch by extracting features based on distributed representations of text and images.

NLP and computer vision. On one hand, a number of methods have been developed to build hierarchies based on lexical patterns in text (Yang and Callan, 2009; Snow et al., 2006; Kozareva and Hovy, 2010; Navigli et al., 2011; Fu et al., 2014; Bansal et al., 2014; Tuan et al., 2015). These works generally ignore the rich visual data which encode important perceptual semantics (Bruni et al., 2014) and have proven to be complementary to linguistic information and helpful for many tasks (Silberer and Lapata, 2014; Kiela and Bottou, 2014; Zhang et al., 2015; Chen et al., 2013). On the other hand, researchers have built visual hierarchies by utilizing only visual features (Griffin and Perona, 2008; Yan et al., 2015; Sivic et al., 2008). The resulting hierarchies are limited in interpretability and usability for knowledge transfer.

Hence, we propose to combine both visual and textual knowledge to automatically build taxonomies. We induce *is-a* taxonomies by supervised learning from existing entity ontologies where each concept category (entity) is associated with images, either from existing dataset (e.g. ImageNet (Deng et al., 2009)) or retrieved from the web using search engines, as illustrated in Fig 1. Such a scenario is realistic and can be extended to a variety of tasks; for example, in knowledge base

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1791–1801,
Berlin, Germany, August 7-12, 2016.

construction (Chen et al., 2013), text and image collections are readily available but label relations among categories are to be uncovered. In large-scale object recognition, automatically learning relations between labels can be quite useful (Deng et al., 2014; Zhao et al., 2011).

Both textual and visual information provide important cues for taxonomy induction. Fig 1 illustrates this via an example. The parent category *seafish* and its two child categories *shark* and *ray* are closely related as: (1) there is a hypernym-hyponym (*is-a*) relation between the words "seafish" and "shark"/"ray" through text descriptions like "...seafish, such as shark and ray...", "...shark and ray are a group of seafish..."; (2) images of the close neighbors, e.g., *shark* and *ray* are usually visually similar and images of the child, e.g. *shark/ray* are similar to a subset of images of *seafish*. To effectively capture these patterns, in contrast to previous works that rely on various hand-crafted features (Chen et al., 2013; Bansal et al., 2014), we extract features by leveraging the *distributed representations* that embed images (Simonyan and Zisserman, 2014) and words (Mikolov et al., 2013) as compact vectors, based on which the semantic closeness is directly measured in vector space. Further, we develop a probabilistic framework that integrates the rich multi-modal features to induce "is-a" relations between categories, encouraging *local semantic consistency* that each category should be visually and textually close to its parent and siblings.

In summary, this paper has the following contributions: (1) We propose a novel probabilistic Bayesian model (Section 3) for taxonomy induction by jointly leveraging textual and visual data. The model is discriminatively trained and can be directly applied to build a taxonomy from scratch for a collection of semantic labels. (2) We design novel features (Section 4) based on general-purpose distributed representations of text and images to capture both textual and visual relations between labels. (3) We evaluate our model and features on the ImageNet hierarchies with two different taxonomy induction tasks (Section 5). We achieve superior performance on both tasks and improve the F_1 score by 2x in the *taxonomy construction* task, compared to previous approaches. Extensive comparisons demonstrate the effectiveness of integrating visual features with language features for taxonomy induction. We also provide qualitative analysis on our features, the learned model, and the taxonomies induced to provide further insights (Section 5.3).

2 Related Work

Many approaches have been recently developed that build hierarchies purely by identifying either lexical patterns or statistical features in text corpora (Yang and Callan, 2009; Snow et al., 2006; Kozareva and Hovy, 2010; Navigli et al., 2011; Zhu et al., 2013; Fu et al., 2014; Bansal et al., 2014; Tuan et al., 2014; Tuan et al., 2015; Kiela et al., 2015). The approaches in Yang and Callan (2009) and Snow et al. (2006) assume a starting incomplete hierarchy and try to extend it by inserting new terms. Kozareva and Hovy (2010) and Navigli et al. (2011) first find leaf nodes and then use lexical patterns to find intermediate terms and all the attested hypernymy links between them. In (Tuan et al., 2014), syntactic contextual similarity is exploited to construct the taxonomy, while Tuan et al. (2015) go one step further to consider trustiness and collective synonym/contrastive evidence. Different from them, our model is discriminatively trained with multi-modal data. The works of Fu et al. (2014) and Bansal et al. (2014) use similar language-based features as ours. Specifically, in (Fu et al., 2014), linguistic regularities between pretrained word vectors (Mikolov et al., 2013) are modeled as projection mappings. The trained projection matrix is then used to induce pairwise hypernym-hyponym relations between words. Our features are partially motivated by Fu et al. (2014), but we jointly leverage both textual and visual information. In Kiela et al. (2015), both textual and visual evidences are exploited to detect pairwise lexical entailments. Our work is significantly different as our model is optimized over the whole taxonomy space rather than considering only word pairs separately. In (Bansal et al., 2014), a structural learning model is developed to induce a globally optimal hierarchy. Compared with this work, we exploit much richer features from both text and images, and leverage distributed representations instead of hand-crafted features.

Several approaches (Griffin and Perona, 2008; Bart et al., 2008; Marszałek and Schmid, 2008) have also been proposed to construct visual hierarchies from image collections. In (Bart et al., 2008), a nonparametric Bayesian model is developed to group images based on low-level features.

In (Griffin and Perona, 2008) and (Marszałek and Schmid, 2008), a visual taxonomy is built to accelerate image categorization. In (Chen et al., 2013), only binary object-object relations are extracted using co-detection matrices. Our work differs from all of these as we integrate textual with visual information to construct taxonomies.

Also of note are several works that integrate text and images as evidence for knowledge base autocompletion (Bordes et al., 2011) and zero-shot recognition (Gan et al., 2015; Gan et al., ; Socher et al., 2013). Our work is different because our task is to accurately construct multi-level hyponym-hypernym hierarchies from a set of (seen or unseen) categories.

3 Taxonomy Induction Model

Our model is motivated by the key observation that in a semantically meaningful taxonomy, a category tends to be closely related to its children as well as its siblings. For instance, there exists a hypernym-hyponym relation between the name of category *shark* and that of its parent *seafish*. Besides, images of *shark* tend to be visually similar to those of *ray*, both of which are seafishes. Our model is thus designed to encourage such local semantic consistency; and by jointly considering all categories in the inference, a globally optimal structure is achieved. A key advantage of the model is that we incorporate both visual and textual features induced from distributed representations of images and text (Section 4). These features capture the rich underlying semantics and facilitate taxonomy induction. We further distinguish the relative importance of visual and textual features that could vary in different layers of a taxonomy. Intuitively, visual features would be increasingly indicative in the deeper layers, as sub-categories under the same category of specific objects tend to be visually similar. In contrast, textual features would be more important when inducing hierarchical relations between the categories of general concepts (i.e. in the near-root layers) where visual characteristics are not necessarily similar.

3.1 The Problem

Assume a set of N categories $\boldsymbol{x} = \{x_1, x_2, \dots, x_N\}$, where each category x_n consists of a text term t_n as its name, as well as a set of images $\boldsymbol{i}_n = \{i_1, i_2, \dots\}$. Our goal is to construct a taxonomy tree T over these categories[1], such that categories of specific object types (e.g. shark) are grouped and assigned to those of general concepts (e.g. seafish). As the categories in $\boldsymbol{x}$ may be from multiple disjoint taxonomy trees, we add a *pseudo* category x_0 as the hyper-root so that the optimal taxonomy is ensured to be a single tree. Let $z_n \in \{1, \dots, N\}$ be the index of the parent of category x_n, i.e. x_{z_n} is the hypernymic category of x_n. Thus the problem of inducing a taxonomy structure is equivalent to inferring the conditional distribution $p(\boldsymbol{z}|\boldsymbol{x})$ over the set of (latent) indices $\boldsymbol{z} = \{z_1, \dots, z_n\}$, based on the images and text.

3.2 Model

We formulate the distribution $p(\boldsymbol{z}|\boldsymbol{x})$ through a model which leverages rich multi-modal features. Specifically, let $\boldsymbol{c}_n$ be the set of child nodes of category x_n in a taxonomy encoded by $\boldsymbol{z}$. Our model is defined as

$$p_w(\boldsymbol{z}, \boldsymbol{\pi}|\boldsymbol{x}, \boldsymbol{\alpha}) \propto p(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_{n=1}^{N} \prod_{x_{n'} \in \boldsymbol{c}_n} \pi_n g_w(x_n, x_{n'}, \boldsymbol{c}_n \backslash x_{n'})$$

where $g_w(x_n, x_{n'}, \boldsymbol{c}_n \backslash x_{n'})$, defined as (1)

$$g_w(x_n, x_{n'}, \boldsymbol{c}_n \backslash x_{n'}) = \exp\{\boldsymbol{w}_{d(x_{n'})}^{\top} \boldsymbol{f}_{n,n',\boldsymbol{c}_n \backslash x_{n'}}\},$$

measures the semantic consistency between category $x_{n'}$, its parent x_n as well as its siblings indexed by $\boldsymbol{c}_n \backslash x_{n'}$. The function $g_w(\cdot)$ is loglinear with respect to $\boldsymbol{f}_{n,n',\boldsymbol{c}_n \backslash x_{n'}}$, which is the feature vector defined over the set of relevant categories $(x_n, x_{n'}, \boldsymbol{c}_n \backslash x_{n'})$, with $\boldsymbol{c}_n \backslash x_{n'}$ being the set of child categories excluding $x_{n'}$ (Section 4). The simple exponential formulation can effectively encourage close relations among nearby categories in the induced taxonomy. The function has combination weights $\boldsymbol{w} = \{\boldsymbol{w}_1, \dots, \boldsymbol{w}_L\}$, where L is the maximum depth of the taxonomy, to capture the importance of different features, and the function $d(x_{n'})$ to return the depth of $x_{n'}$ in the current taxonomy. Each layer l $(1 \le l \le L)$ of the taxonomy has a specific $\boldsymbol{w}_l$ thereby allowing varying weights of the same features in different layers. The parameters are learned in a *supervised* manner. In eq 1, we also introduce a weight π_n for each node x_n, in order to capture the varying popularity of different categories (in terms of being a parent category). For example, some categories like

[1] We assume T to be a tree. Most existing taxonomies are modeled as trees (Bansal et al., 2014), since a tree helps simplify the construction and ensures that the learned taxonomy is interpretable. With minor modifications, our model also works on non-tree structures.

plant can have a large number of sub-categories, while others such as *stone* have less. We model $\boldsymbol{\pi}$ as a multinomial distribution with Dirichlet prior $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$ to encode any prior knowledge of the category popularity[2]; and the conjugacy allows us to marginalize out $\boldsymbol{\pi}$ analytically to get

$$p_w(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{\alpha}) \propto \int p(\boldsymbol{\pi}|\boldsymbol{\alpha}) \prod_{n=1}^{N} \prod_{x_{n'} \in \boldsymbol{c}_n} \pi_n g_w(x_n, x_{n'}, \boldsymbol{c}_n \backslash x_{n'}) d\boldsymbol{\pi}$$
$$\propto \prod_n \Gamma(q_n + \alpha_n) \prod_{x_{n'} \in \boldsymbol{c}_n} g_w(x_n, x_{n'}, \boldsymbol{c}_n \backslash x_{n'}) \tag{2}$$

where q_n is the number of children of category x_n.

Next, we describe our approach to infer the expectation for each z_n, and based on that select a particular taxonomy structure for the category nodes $\boldsymbol{x}$. As $\boldsymbol{z}$ is constrained to be a tree (i.e. cycle without loops), we include with eq 2, an indicator factor $\mathbf{1}(\boldsymbol{z})$ that takes 1 if $\boldsymbol{z}$ corresponds a tree and 0 otherwise. We modify the inference algorithm appropriately to incorporate this constraint.

Inference. Exact inference is computationally intractable due to the normalization constant of eq 2. We therefore use Gibbs Sampling, a procedure for approximate inference. Here we present the sampling formula for each z_n directly, and defer the details to the supplementary material. The sampling procedure is highly efficient because the normalization term and the factors that are irrelevant to z_n are cancelled out. The formula is

$$p(z_n = m|\boldsymbol{z} \backslash z_n, \cdot) \propto \mathbf{1}(z_n = m, \boldsymbol{z} \backslash z_n) \cdot \left(q_m^{-n} + \alpha_m\right) \cdot \frac{\prod_{x_{n'} \in \boldsymbol{c}_m \cup \{x_n\}} g_w(x_m, x_{n'}, \boldsymbol{c}_m \cup \{x_n\})}{\prod_{x_{n'} \in \boldsymbol{c}_m \backslash x_n} g_w(x_m, x_{n'}, \boldsymbol{c}_m \backslash x_n)}, \tag{3}$$

where q_m is the number of children of category m; the superscript $-n$ denotes the number excluding x_n. Examining the validity of the taxonomy structure (i.e. the tree indicator) in each sampling step can be computationally prohibitive. To handle this, we restrict the candidate value of z_n in eq 3, ensuring that the new z_n is always a tree. Specifically, given a tree T, we define a *structure operation* as the procedure of detaching one node x_n in T from its parent and appending it to another node x_m which is not a descendant of x_n.

Proposition 1. *(1) Applying a structure operation on a tree T will result in a structure that is still a tree. (2) Any tree structure over the node set $\boldsymbol{x}$ that has the same root node with tree T can be achieved by applying structure operation on T a finite number of times.*

[2] $\boldsymbol{\alpha}$ could be estimated using training data.

The proof is straightforward and we omit it due to space limitations. We also add a pseudo node x_0 as the fixed root of the taxonomy. Hence by initializing a tree-structured state rooted at x_0 and restricting each updating step as a structure operation, our sampling procedure is able to explore the whole valid tree space.

Output taxonomy selection. To apply the model to discover the underlying taxonomy from a given set of categories, we first obtain the marginals of $\boldsymbol{z}$ by averaging over the samples generated through eq 3, then output the optimal taxonomy $\boldsymbol{z}^*$ by finding the maximum spanning tree (MST) using the Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Bansal et al., 2014).

Training. We need to learn the model parameters $\boldsymbol{w}_l$ of each layer l, which capture the relative importance of different features. The model is trained using the EM algorithm. Let $\ell(x_n)$ be the depth (layer) of category x_n; and $\tilde{\boldsymbol{z}}$ (siblings $\tilde{\boldsymbol{c}}_n$) denote the gold structure in training data. Our training algorithm updates $\boldsymbol{w}$ through maximum likelihood estimation, wherein the gradient of $\boldsymbol{w}_l$ is (see the supplementary materials for details):

$$\delta \boldsymbol{w}_l = \sum_{n:\ell(x_n)=l} \{\boldsymbol{f}(x_{\tilde{z}_n}, x_n, \tilde{\boldsymbol{c}}_n \backslash x_n) - \mathbb{E}_p[\boldsymbol{f}(x_{z_n}, x_n, \boldsymbol{c}_n \backslash x_n)]\},$$

which is the net difference between gold feature vectors and expected feature vectors as per the model. The expectation is approximated by collecting samples using the sampler described above and averaging them.

4 Features

In this section, we describe the feature vector $\boldsymbol{f}$ used in our model, and defer more details in the supplementary material. Compared to previous taxonomy induction works which rely purely on linguistic information, we exploit both perceptual and textual features to capture the rich spectrum of semantics encoded in images and text. Moreover, we leverage the *distributed representations* of images and words to construct compact and effective features. Specifically, each image i is represented as an embedding vector $\boldsymbol{v}_i \in \mathbb{R}^a$ extracted by deep convolutional neural networks. Such image representation has been successfully applied in various vision tasks. On the other hand, the category name t is represented by its word embedding $\boldsymbol{v}_t \in \mathbb{R}^b$, a low-dimensional dense vector induced by the Skip-gram model (Mikolov et

al., 2013) which is widely used in diverse NLP applications too. Then we design $\boldsymbol{f}(x_n, x_{n'}, \boldsymbol{c}_n \backslash x_{n'})$ based on the above image and text representations. The feature vector $\boldsymbol{f}$ is used to measure the local semantic consistency between category $x_{n'}$ and its parent category x_n as well as its siblings $\boldsymbol{c}_n \backslash x_{n'}$.

4.1 Image Features

Sibling similarity. As mentioned above, close neighbors in a taxonomy tend to be visually similar, indicating that the embedding of images of sibling categories should be close to each other in the vector space $\mathbb{R}^a$. For a category x_n and its image set $\boldsymbol{i}_n$, we fit a Gaussian distribution $\mathcal{N}(\overline{\boldsymbol{v}}_{\boldsymbol{i}_n}, \Sigma_n)$ to the image vectors, where $\overline{\boldsymbol{v}}_{\boldsymbol{i}_n} \in \mathbb{R}^a$ is the mean vector and $\Sigma_n \in \mathbb{R}^{a \times a}$ is the covariance matrix. For a sibling category x_m of x_n, we define the visual similarity between x_n and x_m as

$$vissim(x_n, x_m) = [\mathcal{N}(\overline{\boldsymbol{v}}_{\boldsymbol{i}_m}; \overline{\boldsymbol{v}}_{\boldsymbol{i}_n}, \Sigma_n) + \mathcal{N}(\overline{\boldsymbol{v}}_{\boldsymbol{i}_n}; \overline{\boldsymbol{v}}_{\boldsymbol{i}_m}, \Sigma_m)]/2$$

which is the average probability of the mean image vector of one category under the Gaussian distribution of the other. This takes into account not only the distance between the mean images, but also the closeness of the images of each category. Accordingly, we compute the visual similarity between $x_{n'}$ and the set $\boldsymbol{c}_n \backslash x_{n'}$ by averaging:

$$vissim(x_{n'}, \boldsymbol{c}_n \backslash x_{n'}) = \frac{\sum_{x_m \in \boldsymbol{c}_n \backslash x_{n'}} vissim(x_{n'}, x_m)}{|\boldsymbol{c}_n| - 1}.$$

We then bin the values of $vissim(x_{n'}, \boldsymbol{c}_n \backslash x_{n'})$ and represent it as an one-hot vector, which constitutes $\boldsymbol{f}$ as a component named as *siblings image-image relation feature* (denoted as *S-V1*[3]).

Parent prediction. Similar to feature S-V1, we also create the similarity feature between the image vectors of the parent and child, to measure their visual similarity. However, the parent node is usually a more general concept than the child, and it usually consists of images that are not necessarily similar to its child. Intuitively, by narrowing the set of images to those that are most similar to its child improves the feature. Therefore, different from S-V1, when estimating the Gaussian distribution of the parent node, we only use the top K images with highest probabilities under the Gaussian distribution of the child node. We empirically show in section 5.3 that choosing an appropriate K consistently boosts the performance. We name this feature as *parent-child image-image relation feature* (denoted as *PC-V1*).

[3]S: sibling, PC: parent-child, V: visual, T: textual.

Further, inspired by the linguistic regularities of word embedding, i.e. the hypernym-hyponym relationship between words can be approximated by a linear projection operator between word vectors (Mikolov et al., 2013; Fu et al., 2014), we design a similar strategy to (Fu et al., 2014) between images and words so that the parent can be "predicted" given the image embedding of its child category and the projection matrix. Specifically, let $(x_n, x_{n'})$ be a parent-child pair in the training data, we learn a projection matrix $\boldsymbol{\Phi}$ which minimizes the distance between $\boldsymbol{\Phi}\overline{\boldsymbol{v}}_{\boldsymbol{i}_{n'}}$ (i.e. the projected mean image vector $\overline{\boldsymbol{v}}_{\boldsymbol{i}_{n'}}$ of the child) and $\boldsymbol{v}_{t_n}$ (i.e. the word embedding of the parent):

$$\boldsymbol{\Phi}^* = \underset{\boldsymbol{\Phi}}{\operatorname{argmin}} \frac{1}{N} \sum_n \|\boldsymbol{\Phi}\overline{\boldsymbol{v}}_{\boldsymbol{i}_{n'}} - \boldsymbol{v}_{t_n}\|_2^2 + \lambda \|\boldsymbol{\Phi}\|_1,$$

where N is the number of parent-child pairs in the training data. Once the projection matrix has been learned, the similarity between a child node $x_{n'}$ and its parent x_n is computed as $\|\boldsymbol{\Phi}\overline{\boldsymbol{v}}_{\boldsymbol{i}_{n'}} - \boldsymbol{v}_{t_n}\|$, and we also create an one-hot vector by binning the feature value. We call this feature as *parent-child image-word relation feature* (*PC-V2*).

4.2 Word Features

We briefly introduce the text features employed. More details about the text feature extraction could be found in the supplementary material.

Word embedding features.d PC-V1, We induce features using word vectors to measure both sibling-sibling and parent-child closeness in text domain (Fu et al., 2014). One exception is that, as each category has only one word, the sibling similarity is computed as the cosine distance between two word vectors (instead of mean vectors). This will produce another two parts of features, *parent-child word-word relation feature* (*PC-T1*) and *siblings word-word relation feature* (*S-T1*).

Word surface features. In addition to the embedding-based features, we further leverage lexical features based on the surface forms of child/parent category names. Specifically, we employ the *Capitalization*, *Ends with*, *Contains*, *Suffix match*, *LCS* and *Length different* features, which are commonly used in previous works in taxonomy induction (Yang and Callan, 2009; Bansal et al., 2014).

5 Experiments

We first disclose our implementation details in section 5.1 and the supplementary material for bet-

ter reproducibility. We then compare our model with previous state-of-the-art methods (Fu et al., 2014; Bansal et al., 2014) with two taxonomy induction tasks. Finally, we provide analysis on the weights and taxonomies induced.

5.1 Implementation Details

Dataset. We conduct our experiments on the ImageNet2011 dataset (Deng et al., 2009), which provides a large collection of category items (synsets), with associated images and a label hierarchy (sampled from WordNet) over them. The original ImageNet taxonomy is preprocessed, resulting in a tree structure with 28231 nodes.

Word embedding training. We train word embedding for synsets by replacing each word/phrase in a synset with a unique token and then using Google's word2vec tool (Mikolov et al., 2013). We combine three public available corpora together, including the latest Wikipedia dump (Wikipedia, 2014), the One Billion Word Language Modeling Benchmark (Chelba et al., 2013) and the UMBC webbase corpus (Han et al., 2013), resulting in a corpus with total 6 billion tokens. The dimension of the embedding is set to 200.

Image processing. we employ the ILSVRC12 pre-trained convolutional neural networks (Simonyan and Zisserman, 2014) to embed each image into the vector space. Then, for each category x_n with images, we estimate a multivariate Gaussian parameterized by $\mathcal{N}_{x_n} = (\mu_{x_n}, \Sigma_{x_n})$, and constrain Σ_{x_n} to be diagonal to prevent overfitting. For categories with very few images, we only estimate a mean vector μ_{x_n}. For nodes that do not have images, we ignore the visual feature.

Training configuration. The feature vector is a concatenation of 6 parts, as detailed in section 4. All pairwise distances are precomputed and stored in memory to accelerate Gibbs sampling. The initial learning rate for gradient descent in the M step is set to 0.1, and is decreased by a fraction of 10 every 100 EM iterations.

5.2 Evaluation

5.2.1 Experimental Settings

We evaluate our model on three subtrees sampled from the ImageNet taxonomy. To collect the subtrees, we start from a given root (e.g. consumer goods) and traverse the full taxonomy using BFS, and collect all descendant nodes within a depth h (number of nodes in the longest path). We vary h to get a series of subtrees with increasing heights $h \in \{4, 5, 6, 7\}$ and various scales (maximally 1326 nodes) in different domains. The statistics of the evaluation sets are provided in Table 1. To avoid ambiguity, all nodes used in ILSVRC 2012 are removed as the CNN feature extractor is trained on them.

Trees	Tree A	Tree B	Tree C
Synset ID	12638	19919	23733
Name	consumer goods	animal	food, nutrient
$h = 4$	187	207	572
$h = 5$	362	415	890
$h = 6$	493	800	1166
$h = 7$	524	1386	1326

Table 1: Statistics of our evaluation set. The bottom 4 rows give the number of nodes within each height $h \in \{4, 5, 6, 7\}$. The scale of the threes range from small to large, and there is no overlapping among them.

We design two different tasks to evaluate our model. (1) In the *hierarchy completion* task, we randomly remove some nodes from a tree and use the remaining hierarchy for training. In the test phase, we infer the parent of each removed node and compare it with groundtruth. This task is designed to figure out whether our model can successfully induce hierarchical relations after learning from within-domain parent-child pairs. (2) Different from the previous one, the *hierarchy construction* task is designed to test the generalization ability of our model, i.e. whether our model can learn statistical patterns from one hierarchy and transfer the knowledge to build a taxonomy for another collection of out-of-domain labels. Specifically, we select two trees as the training set to learn $\boldsymbol{w}$. In the test phase, the model is required to build the full taxonomy from scratch for the third tree.

We use *Ancestor* F_1 as our evaluation metric (Kozareva and Hovy, 2010; Navigli et al., 2011; Bansal et al., 2014). Specifically, we measure $F_1 = 2PR/(P+R)$ values of predicted "is-a" relations where the precision (P) and recall (R) are:

$$P = \frac{|\text{isa}_{\text{predicted}} \cap \text{isa}_{\text{gold}}|}{|\text{isa}_{\text{predicted}}|}, R = \frac{|\text{isa}_{\text{predicted}} \cap \text{isa}_{\text{gold}}|}{|\text{isa}_{\text{gold}}|}.$$

We compare our method to two previously state-of-the-art models by Fu et al. (2014) and Bansal et al. (2014), which are closest to ours.

Method	$h = 4$	$h = 5$	$h = 6$	$h = 7$
Hierarchy Completion				
Fu2014	0.66	0.42	0.26	0.21
Ours (L)	0.70	0.49	0.45	0.37
Ours (LV)	**0.73**	**0.51**	**0.50**	**0.42**
Hierarchy Construction				
Fu2014	0.53	0.33	0.28	0.18
Bansal2014	0.67	0.53	0.43	0.37
Ours (L)	0.58	0.41	0.36	0.30
Ours (LB)	0.68	0.55	0.45	0.40
Ours (LV)	0.66	0.52	0.42	0.34
Ours (LVB - E)	0.68	0.55	0.44	0.39
Ours (LVB)	**0.70**	**0.57**	**0.49**	**0.43**

Table 2: Comparisons among different variants of our model, Fu et al. (2014) and Bansal et al. (2014) on two tasks. The ancestor-F_1 scores are reported.

5.2.2 Results

Hierarchy completion. In the *hierarchy completion* task, we split each tree into 70% nodes for training and 30% for test, and experiment with different h. We compare the following three systems: (1) *Fu2014*[4] (Fu et al., 2014); (2) *Ours (L)*: Our model with only language features enabled (i.e. surface features, parent-child word-word relation feature and siblings word-word relation feature); (3) *Ours (LV)*: Our model with both language features and visual features [5]. The average performance on three trees are reported at Table 2. We observe that the performance gradually drops when h increases, as more nodes are inserted when the tree grows higher, leading to a more complex and difficult taxonomy to be accurately constructed. Overall, our model outperforms Fu2014 in terms of the F_1 score, even without visual features. In the most difficult case with $h = 7$, our model still holds an F_1 score of 0.42 ($2\times$ of Fu2014), demonstrating the superiority of our model.

Hierarchy construction. The hierarchy construction task is much more difficult than hierarchy completion task because we need to build a taxonomy from scratch given only a hyper-root. For this task, we use a leave-one-out strategy, i.e. we train our model on every two trees and test on the third, and report the average performance in Table 2. We compare the following methods: (1) *Fu2014*, (2) *Ours (L)*, and (3) *Ours (LV)*, as described above; (4) *Bansal2014*: The model by Bansal et al. (2014) retrained using our dataset; (5) *Ours (LB)*: By excluding visual features, but including other language features from Bansal et al. (2014); (6) *Ours (LVB)*: Our full model further enhanced with all semantic features from Bansal et al. (2014); (7) *Ours (LVB - E)*: By excluding word embedding-based language features from *Ours (LVB)*.

As shown, on the hierarchy construction task, our model with only language features still outperforms Fu2014 with a large gap (0.30 compared to 0.18 when $h = 7$), which uses similar embedding-based features. The potential reasons are two-fold. First, we take into account not only parent-child relations but also siblings. Second, their method is designed to induce only pairwise relations. To build the full taxonomy, they first identify all possible pairwise relations using a simple thresholding strategy and then eliminate conflicted relations to obtain a legitimate tree hierarchy. In contrast, our model is optimized over the full space of all legitimate taxonomies by taking the *structure operation* in account during Gibbs sampling.

When comparing to Bansal2014, our model with only word embedding-based features underperforms theirs. However, when introducing visual features, our performance is comparable (p-value = 0.058).Furthermore, if we discard visual features but add semantic features from Bansal et al. (2014), we achieve a slight improvement of 0.02 over Bansal2014 (p-value = 0.016), which is largely attributed to the incorporation of word embedding-based features that encode high-level linguistic regularity. Finally, if we enhance our full model with all semantic features from Bansal et al. (2014), our model outperforms theirs by a gap of 0.04 (p-value < 0.01), which justifies our intuition that perceptual semantics underneath visual contents are quite helpful.

5.3 Qualitative Analysis

In this section, we conduct qualitative studies to investigate *how* and *when* the visual information helps the taxonomy induction task.

Contributions of visual features. To evaluate the contribution of each part of the visual features to the final performance, we train our model jointly with textual features and different combinations of visual features, and report the ancestor-F_1 scores. As shown in Table 3. When incorporating the feature S-V1, the performance is substantially boosted by a large gap at all heights, show-

[4]We tried different parameter settings for the number of clusters C and the identification threshold δ, and reported the best performance we achieved.

[5]In the comparisons to (Fu et al., 2014) and (Bansal et al., 2014), we simply set $K = \infty$, *i.e.* we use all available images of the parent category to estimate the PC-V1 feature.

S-V1	PC-V1	PC-V2	h = 4	h = 5	h = 6	h = 7
			0.58	0.41	0.36	0.30
✓			0.63	0.48	0.40	0.32
	✓		0.61	0.44	0.38	0.31
		✓	0.60	0.42	0.37	0.31
✓	✓		0.65	**0.52**	0.41	0.33
✓	✓	✓	**0.66**	**0.52**	**0.42**	**0.34**

Table 3: The performance when different combinations of visual features are enabled.

ing that visual similarity between sibling nodes is a strong evidence for taxonomy induction. It is intuitively plausible, as it is highly likely that two specific categories share a common (and more general) parent category if similar visual contents are observed between them. Further, adding the PC-V1 feature gains us a better improvement than adding PC-V2, but both minor than S-V1.

Compared to that of siblings, the visual similarity between parents and children does not strongly holds all the time. For example, images of *Terrestrial animal* are only partially similar to those of *Feline*, because the former one contains the later one as a subset. Our feature captures this type of "contain" relation between parents and children by considering only the top-K images from the parent category that have highest probabilities under the Gaussian distribution of the child category. To see this, we vary K while keep all other settings, and plot the F_1 scores in Fig 2. We observe a trend that when we gradually increase K, the performance goes up until reaching some maximal; It then slightly drops (or oscillates) even when more images are available, which confirms with our feature design that only top images should be considered in parent-child visual similarity.

Overall, the three visual features complement each other, and achieve the highest performance when combined.

Visual representations. To investigate how the image representations affect the final performance, we compare the ancestor-F1 score when different pre-trained CNNs are used for visual feature extraction. Specifically, we employ both the CNN-128 model (128 dimensional feature with 15.6% top-5 error on ILSVRC12) and the VGG-16 model (4096 dimensional feature with 7.5% top-5 error) by Simonyan and Zisserman (2014), but only observe a slight improvement of 0.01 on the ancestor-F1 score for the later one.

Relevance of textual and visual features v.s. depth of tree. Compared to Bansal et al. (2014),

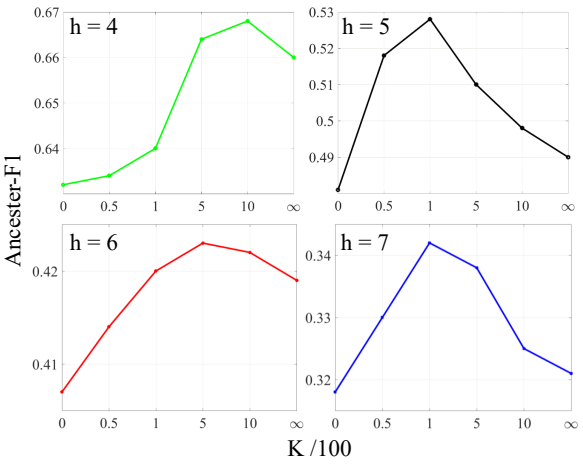


Figure 2: The Ancestor-F_1 scores changes over K (number of images used in the PC-V1 feature) at different heights. The values in the x-axis are $K/100$; $K = \infty$ means all images are used.

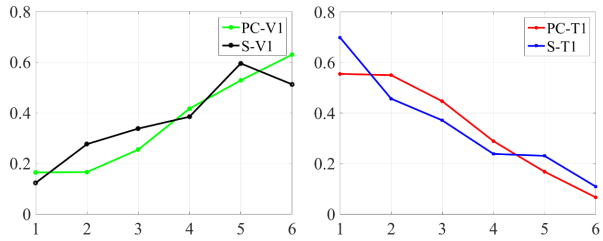


Figure 3: Normalized weights of each feature v.s. the layer depth.

a major difference of our model is that different layers of the taxonomy correspond to different weights $\boldsymbol{w}_l$, while in (Bansal et al., 2014) all layers share the same weights. Intuitively, introducing layer-wise $\boldsymbol{w}$ not only extends the model capacity, but also differentiates the importance of each feature at different layers. For example, the images of two specific categories, such as *shark* and *ray*, are very likely to be visually similar. However, when the taxonomy goes from bottom to up (specific to general), the visual similarity is gradually undermined — images of *fish* and *terrestrial animal* are not necessarily similar any more. Hence, it is necessary to privatize the weights $\boldsymbol{w}$ for different layers to capture such variations, i.e. the visual features become more and more evident from shallow to deep layers, while the textual counterparts, which capture more abstract concepts, relatively grow more indicative oppositely from specific to general.

To visualize the variations across layers, for each feature component, we fetch its correspond-

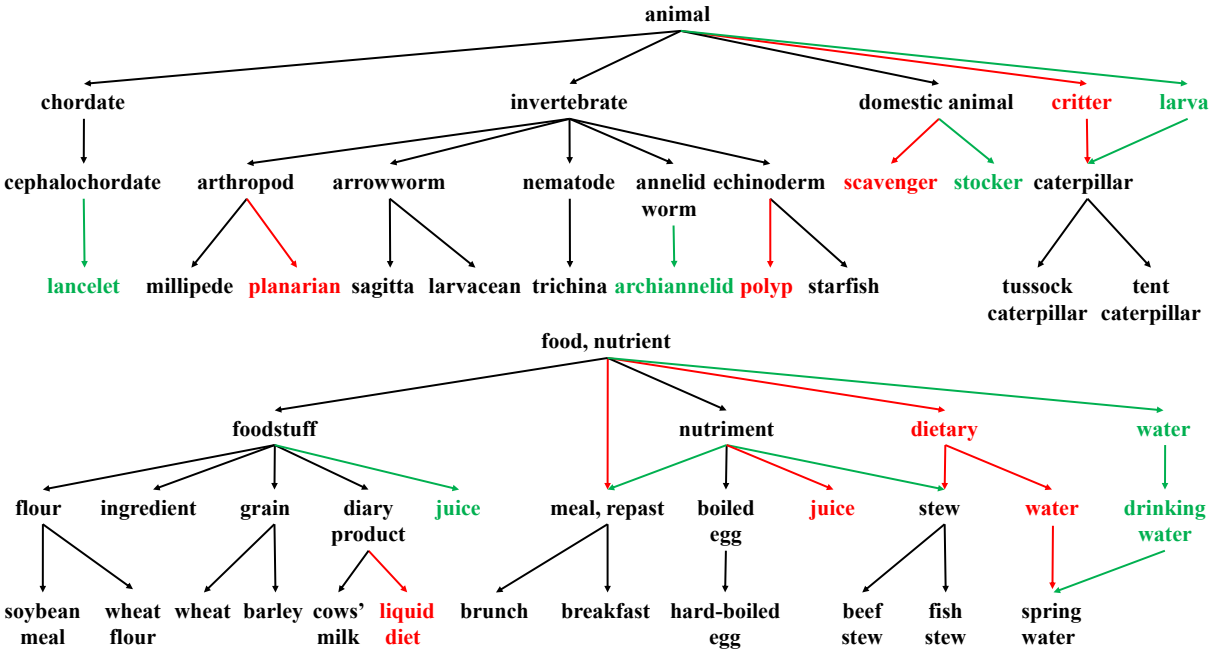


Figure 4: Excerpts of the prediction taxonomies, compared to the groundturth. Edges marked as red and green are false predictions and unpredicted groundtruth links, respectively.

ing block in $\boldsymbol{w}$ as V. Then, we average $|V|$ and observe how its values change with the layer depth h. For example, for the parent-child word-word relation feature, we first fetch its corresponding weights V from $\boldsymbol{w}$ as a 20×6 matrix, where 20 is the feature dimension and 6 is the number of layers. We then average its absolute values[6] in column and get a vector v with length 6. After ℓ_2 normalization, the magnitude of each entry in v directly reflects the relative importance of the feature as an evidence for taxonomy induction. Fig 3(b) plots how their magnitudes change with h for every feature component averaged on three train/test splits. It is noticeable that for both word-word relations (S-T1, PC-T1), their corresponding weights slightly decrease as h increases. On the contrary, the image-image relation features (S-V1, PC-V1) grows relatively more prominent. The results verify our conjecture that when the category hierarchy goes deeper into more specific classes, the visual similarity becomes relatively more indicative as an evidence for taxonomy induction.

Visualizing results. Finally, we visualize some excerpts of our predicted taxonomies, as compared to the groundtruth in Fig 4.

[6]We take the absolute value because we only care about the relevance of the feature as an evidence for taxonomy induction, but note that the weight can either encourage (positive) or discourage (negative) connections of two nodes.

6 Conclusion

In this paper, we study the problem of automatically inducing semantically meaningful concept taxonomies from multi-modal data. We propose a probabilistic Bayesian model which leverages distributed representations for images and words. We compare our model and features to previous ones on two different tasks using the ImageNet hierarchies, and demonstrate superior performance of our model, and the effectiveness of exploiting visual contents for taxonomy induction. We further conduct qualitative studies and distinguish the relative importance of visual and textual features in constructing various parts of a taxonomy.

Acknowledgements

We would like to thank anonymous reviewers for their valuable feedback. We would also like to thank Mohit Bansal for helpful suggestions. We thank NVIDIA for GPU donations. The work is supported by NSF Big Data IIS1447676.

References

Mohit Bansal, David Burkett, Gerard de Melo, and Dan Klein. 2014. Structured learning for taxonomy induction with belief propagation.

Evgeniy Bart, Ian Porteous, Pietro Perona, and Max

Welling. 2008. Unsupervised learning of visual taxonomies. In *CVPR*.

Or Biran and Kathleen McKeown. 2013. Classifying taxonomic relations between pairs of wikipedia articles.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*, number EPFL-CONF-192344.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*.

Xinlei Chen, Abhinav Shrivastava, and Abhinav Gupta. 2013. Neil: Extracting visual knowledge from web data. In *CVPR*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. 2014. Large-scale object classification using label relation graphs. In *ECCV*.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning semantic hierarchies via word embeddings. In *ACL*.

Chuang Gan, Yi Yang, Linchao Zhu, Deli Zhao, and Yueting Zhuang. Recognizing an action using its name: A knowledge-based approach. *International Journal of Computer Vision*, pages 1–17.

Chuang Gan, Ming Lin, Yi Yang, Yueting Zhuang, and Alexander G Hauptmann. 2015. Exploring semantic inter-class relationships (SIR) for zero-shot action recognition. In *AAAI*.

Gregory Griffin and Pietro Perona. 2008. Learning and using taxonomies for fast visual categorization. In *CVPR*.

Lushan Han, Abhay Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. Umbc ebiquity-core: Semantic textual similarity systems. *Atlanta, Georgia, USA*.

Sanda M Harabagiu, Steven J Maiorano, and Marius A Pasca. 2003. Open-domain textual question answering techniques. *Natural Language Engineering*.

Andreas Hotho, Alexander Maedche, and Steffen Staab. 2002. Ontology-based text document clustering.

Douwe Kiela and Léon Bottou. 2014. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *EMNLP*.

Douwe Kiela, Laura Rimell, Ivan Vulic, and Stephen Clark. 2015. Exploiting image generality for lexical entailment detection. In *ACL*.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *EMNLP*.

Marcin Marszałek and Cordelia Schmid. 2008. Constructing category hierarchies for visual recognition. In *ECCV*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*.

Roberto Navigli, Paola Velardi, and Stefano Faralli. 2011. A graph-based algorithm for inducing lexical taxonomies from scratch. In *IJCAI*.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *ACL*.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Josef Sivic, Bryan C Russell, Andrew Zisserman, William T Freeman, and Alexei A Efros. 2008. Unsupervised discovery of visual object class hierarchies. In *CVPR*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL*.

Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. 2013. Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

Luu Anh Tuan, Jung-jae Kim, and Ng See Kiong. 2014. Taxonomy construction using syntactic contextual evidence. In *EMNLP*.

Luu Anh Tuan, Jung-jae Kim, and Ng See Kiong. 2015. Incorporating trustiness and collective synonym/contrastive evidence into taxonomy construction.

Wikipedia. 2014. `https://dumps.wikimedia.org/enwiki/20141208/`.

Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. 2015. Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition. In *ICCV*.

Hui Yang and Jamie Callan. 2009. A metric-based framework for automatic taxonomy induction. In *ACL-IJCNLP*.

Hao Zhang, Gunhee Kim, and Eric P. Xing. 2015. Dynamic topic modeling for monitoring market competition from online text and image data. In *KDD*.

Bin Zhao, Fei Li, and Eric P Xing. 2011. Large-scale category structure aware image categorization. In *NIPS*.

Xingwei Zhu, Zhao-Yan Ming, Xiaoyan Zhu, and Tat-Seng Chua. 2013. Topic hierarchy construction for the organization of multi-source user generated contents. In *SIGIR*.

Generating Natural Questions About an Image

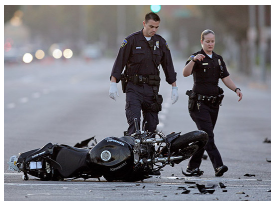
Nasrin Mostafazadeh[1], **Ishan Misra**[2], **Jacob Devlin**[3], **Margaret Mitchell**[3]
Xiaodong He[3], **Lucy Vanderwende**[3]
1 University of Rochester, 2 Carnegie Mellon University,
3 Microsoft Research
nasrinm@cs.rochester.edu, lucyv@microsoft.com

Abstract

There has been an explosion of work in the vision & language community during the past few years from image captioning to video transcription, and answering questions about images. These tasks have focused on literal descriptions of the image. To move beyond the literal, we choose to explore how questions about an image are often directed at commonsense inference and the abstract events evoked by objects in the image. In this paper, we introduce the novel task of **Visual Question Generation (VQG)**, where the system is tasked with asking a natural and engaging question when shown an image. We provide three datasets which cover a variety of images from object-centric to event-centric, with considerably more abstract training data than provided to state-of-the-art captioning systems thus far. We train and test several generative and retrieval models to tackle the task of VQG. Evaluation results show that while such models ask reasonable questions for a variety of images, there is still a wide gap with human performance which motivates further work on connecting images with commonsense knowledge and pragmatics. Our proposed task offers a new challenge to the community which we hope furthers interest in exploring deeper connections between vision & language.

1 Introduction

We are witnessing a renewed interest in interdisciplinary AI research in vision & language, from descriptions of the visual input such as image captioning (Chen et al., 2015; Fang et al., 2014; Donahue et al., 2014; Chen et al., 2015) and video transcription (Rohrbach et al., 2012; Venugopalan et al., 2015), to testing computer understanding of an image through question answering (Antol et al., 2015; Malinowski and Fritz, 2014). The most established work in the vision & language community is 'image captioning', where the task is to produce a literal description of the image. It has been shown (Devlin et al., 2015; Fang et al., 2014; Donahue et al., 2014) that a reasonable language modeling paired with deep visual features trained on large enough datasets promise a good performance on image captioning, making it a less challenging task from language learning perspective. Furthermore, although this task has a great value for communities of people who are low-sighted or cannot see in all or some environments, for others, the description does not add anything to what a person has already perceived.



Natural Questions:
- Was anyone injured in the crash?
- Is the motorcyclist alive?
- What caused this accident?

Generated Caption:
- A man standing next to a motorcycle.

Figure 1: Example image along with its natural questions and automatically generated caption.

The popularity of the image sharing applications in social media and user engagement around images is evidence that commenting on pictures is a very natural task. A person might respond to an image with a short comment such as 'cool', 'nice pic' or ask a question. Imagine someone has shared the image in Figure 1. What is the very first question that comes to mind? Your question is most probably very similar to the questions listed next to the image, expressing concern about the motorcyclist (who is not even present in the image). As you can tell, natural questions are not

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1802–1813,
Berlin, Germany, August 7-12, 2016.

about what is seen, the policemen or the motorcycle, but rather about what is inferred given these objects, e.g., an accident or injury. As such, questions are often about abstract concepts, i.e., events or states, in contrast to the concrete terms[1] used in image captioning. It is clear that the corresponding automatically generated caption[2] for Figure 1 presents only a literal description of objects.

To move beyond the literal description of image content, we introduce the novel task of Visual Question Generation (VQG), where given an image, the system should 'ask a natural and engaging question'. Asking a question that can be answered simply by looking at the image would be of interest to the Computer Vision community, but such questions are neither natural nor engaging for a person to answer and so are not of interest for the task of VQG.

Learning to ask questions is an important task in NLP and is more than a syntactic transformation of a declarative sentence (Vanderwende, 2008). Deciding what to ask about demonstrates understanding and as such, question generation provides an indication of machine understanding, just as some educational methods assess students' understanding by their ability to ask relevant questions[3]. Furthermore, training a system to ask a good question (not only answer a question) may imbue the system with what appears to be a cognitive ability unique to humans among other primates (Jordania, 2006). Developing the ability to ask relevant and to-the-point questions can be an essential component of any dynamic learner which seeks information. Such an ability can be an integral component of any conversational agent, either to engage the user in starting a conversation or to elicit task-specific information.

The contributions of this paper can be summarized as follows: (1) in order to enable the VQG research, we carefully created three datasets with a total of 75,000 questions, which range from object- to event-centric images, where we show that VQG covers a wide range of abstract terms including events and states (Section 3). (2) we collected 25,000 gold captions for our event-centric dataset and show that this dataset presents challenges to the state-of-the-art image captioning models (Section 3.3). (3) we perform analysis of various generative and retrieval approaches and conclude that end-to-end deep neural models outperform other approaches on our most-challenging dataset (Section 4). (4) we provide a systematic evaluation methodology for this task, where we show that the automatic metric ΔBLEU strongly correlates with human judgments (Section 5.3). The results show that while our models learn to generate promising questions, there is still a large gap to match human performance, making the generation of relevant and natural questions an interesting and promising new challenge to the community.

2 Related Work

For the task of image captioning, datasets have primarily focused on objects, e.g. Pascal VOC (Everingham et al., 2010) and Microsoft Common Objects in Context (MS COCO) (Lin et al., 2014). MS COCO, for example, includes complex everyday scenes with 91 basic objects in 328k images, each paired with 5 captions. Event detection is the focus in video processing and action detection, but these do not include a textual description of the event (Yao et al., 2011b; Andriluka et al., 2014; Chao et al., 2015; Xiong et al., 2015). The number of actions in each of these datasets is still relatively small, ranging from 40 (Yao et al., 2011a) to 600 (Chao et al., 2015) and all involve human-oriented activity (e.g. 'cooking', 'gardening', 'riding a bike'). In our work, we are focused on generating questions for static images of events, such as 'fire', 'explosion' or 'snowing', which have not yet been investigated in any of the above datasets.

Visual Question Answering is a relatively new task where the system provides an answer to a question about the image content. The most notable, Visual Question Answering (VQA) (Antol et al., 2015), is an open-ended (free-form) dataset, in which both the questions and the answers are crowd-sourced, with workers prompted to ask a visually verifiable question which will 'stump a smart robot'. Gao et al. (2015) used similar methodology to create a visual question answering dataset in Chinese. COCO-QA (CQA) (Ren et al., 2015), in contrast, does not use human-authored questions, but generates questions automatically from image captions of the MS COCO dataset by applying a set of transformation rules to generate the wh-question. The expected answers in CQA

[1] Concrete terms are the ones that can be experienced with five senses. Abstract terms refer to intangible things, such as feelings, concepts, and qualities

[2] Throughout this paper we use the state-of-the-art captioning system (Fang et al., 2014), henceforth MSR captioning system `https://www.captionbot.ai/`, to generate captions.

[3] `http://rightquestion.org/`

Figure 2: Example right and wrong questions for the task of VQG.

are by design limited to objects, numbers, colors, or locations. A more in- depth analysis of VQA and CQA datasets will be presented in Section 3.1.

In this work, we focus on questions which are interesting for a person to answer, not questions designed to evaluate computer vision. A recently published work on VQA, Visual7W (Zhu et al., 2016), establishes a grounding link on the object regions corresponding to the textual answer. This setup enables a system to answer a question with visual answers (in addition to textual answers). They collect a set of 327,939 7W multiple-choice QA pairs, where they point out that 'where', 'when' and 'why' questions often require high-level commonsense reasoning, going beyond spatial reasoning required for 'which' or 'who' questions. This is more in line with the type of questions that VQG captures, however, the majority of the questions in Visual7w are designed to be answerable by only the image, making them unnatural for asking a human. Thus, learning to generate the questions in VQA task is not a useful sub-task, as the intersection between VQG and any VQA questions is by definition minimal.

Previous work on question generation from textual input has focused on two aspects: the grammaticality (Wolfe, 1976; Mitkov and Ha, 2003; Heilman and Smith, 2010) and the content focus of question generation, i.e., "what to ask about". For the latter, several methods have been explored: (Becker et al., 2012) create fill-in-the-blank questions, (Mazidi and Nielsen, 2014) and (Lindberg et al., 2013) use manually constructed question templates, while (Labutov et al., 2015) use crowdsourcing to collect a set of templates and then rank the potentially relevant templates for the selected content. To our knowledge, neither a retrieval model nor a deep representation of textual input, presented in our work, have yet been used to generate questions.

3 Data Collection Methodology

Task Definition: Given an image, the task is to generate a natural question which can potentially engage a human in starting a conversation. Questions that are visually verifiable, i.e., that can be answered by looking at only the image, are outside the scope of this task. For instance, in Figure 2, a question about the number of horses (appearing in the VQA dataset) or the color of the field is not of interest. Although in this paper we focus on asking a question about an image in isolation, adding prior context or history of conversation is the natural next step in this project.

We collected the VQG questions by crowdsourcing the task on Amazon Mechanical Turk (AMT). We provide details on the prompt and the specific instructions for all the crowdsourcing tasks in this paper in the supplementary material. Our prompt was very successful at capturing non-literal questions, as the good question in Figure 2 demonstrates. In the following Sections, we describe our process for selecting the images to be included in the VQG dataset. We start with images from MS COCO, which enables meaningful comparison with VQA and CQA questions. Given that it is more natural for people to ask questions about event-centric images, we explore sourcing eventful images from Flickr and from querying an image search engine. Each data source is represented by 5,000 images, with 5 questions per image.

3.1 $VQG_{coco-5000}$ and $VQG_{Flickr-5000}$

As our first dataset, we collected VQG questions for a sample of images from the MS COCO dataset[4]. In order to enable comparisons with related datasets, we sampled 5,000 images of MS COCO which were also annotated by the CQA dataset (Ren et al., 2015) and by VQA (Antol et al., 2015). We name this dataset $VQG_{coco-5000}$. Table 1 shows a sample MS COCO image along with annotations in the various datasets. As the CQA questions are generated by rule application from captions, they are not always coherent. The VQA questions are written to evaluate the detailed visual understanding of a robot, so their questions are mainly visually grounded and literal. The table demonstrates how different VQG questions are from VQA and CQA questions.

In Figure 3 we provide statistics for the various annotations on that portion of the MS COCO images which are represented in the $VQG_{coco-5000}$

[4] http://mscoco.org/

Dataset	Annotations
COCO	- A man holding a box with a large chocolate covered donut.
CQA	- What is the man holding with a large chocolate-covered doughnut in it?
VQA	- Is this a large doughnut?
VQG	- Why is the donut so large? - Is that for a specific celebration? - Have you ever eaten a donut that large before? - Is that a big donut or a cake? - Where did you get that?

Table 1: Dataset annotations on the above image.

dataset. In Figure 3(a) we compare the percentage of object-mentions in each of the annotations. Object-mentions are words associated with the gold-annotated object boundary boxes[5] as provided with the MS COCO dataset. Naturally, COCO captions (green bars) have the highest percentage of these literal objects. Since object-mentions are often the answer to VQA and CQA questions, those questions naturally contain objects less frequently. Hence, we see that VQG questions include the mention of more of those literal objects. Figure 3(b) shows that COCO captions have a larger vocabulary size, which reflects their longer and more descriptive sentences. VQG shows a relatively large vocabulary size as well, indicating greater diversity in question formulation than VQA and CQA. Moreover, Figure 3(c) shows that the verb part of speech is represented with high frequency in our dataset.

Figure 3(d) depicts the percentage of abstract terms such as 'think' or 'win' in the vocabulary. Following Ferraro et al. (2015), we use a list of most common abstract terms in English (Vanderwende et al., 2015), and count all the other words except a set of function words as concrete. This figure supports our expectation that VQG covers more abstract concepts. Furthermore, Figure 3(e) shows inter-annotation textual similarity according to the BLEU metric (Papineni et al., 2002). Interestingly, VQG shows the highest inter-annotator textual similarity, which reflects on the existence of consensus among human for asking a natural question, even for object-centric images like the ones in MS COCO.

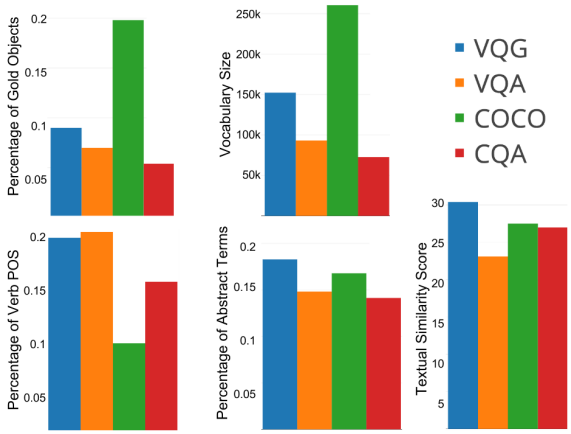


Figure 3: Comparison of various annotations on the MS COCO dataset. (a) Percentage of gold objects used in annotations. (b) Vocabulary size (c) Percentage of verb POS (d) Percentage of abstract terms (e) Inter-annotation textual similarity score.

The MS COCO dataset is limited in terms of the concepts it covers, due to its pre-specified set of object categories. Word frequency in $VQG_{coco-5000}$ dataset, as demonstrated in Figure 4, bears this out, with the words 'cat' and 'dog' the fourth and fifth most frequent words in the dataset. Not shown in the frequency graph is that words such as 'wedding', 'injured', or 'accident' are at the very bottom of frequency ranking list. This observation motivated the collection of the $VQG_{Flickr-5000}$ dataset, with images appearing as the middle photo in a story-full photo album (Huang et al., 2016) on Flickr[6]. The details about this dataset can be found in the supplementary material.

3.2 $VQG_{Bing-5000}$

To obtain a more representative visualization of specific event types, we queried a search engine[7] with 1,200 event-centric query terms which were obtained as follows: we aggregated all 'event' and 'process' hyponyms in WordNet (Miller, 1995), 1,000 most frequent TimeBank events (Pustejovsky et al., 2003) and a set of manually curated 30 stereotypical events, from which we selected the top 1,200 queries based on Project Gutenberg word frequencies. For each query, we collected the first four to five images retrieved, for a total

[5]Note that MS COCO annotates only 91 object categories.

[6]http://www.flickr.com

[7]https://datamarket.azure.com/dataset/bing/search

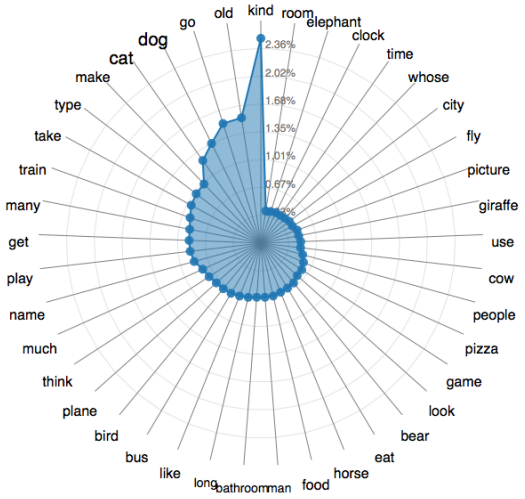


Figure 4: Frequency graph of top 40 words in $VQG_{coco-5000}$ dataset.

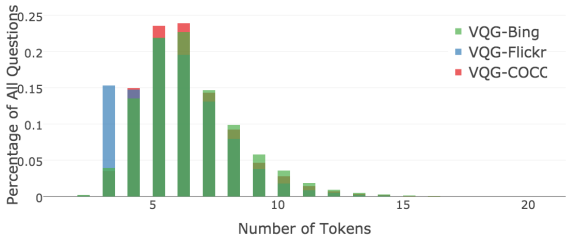


Figure 5: Average annotation length of the three VQG datasets.

of 5,000 images, having first used crowdsourcing to filter out images depicting graphics and cartoons. A similar word frequency analysis shows that the $VQG_{Bing-5000}$ dataset indeed contains more words asking about events: *happen*, *work*, *cause* appear in top 40 words, which was our aim in creating the Bing dataset.

Statistics: Our three datasets together cover a wide range of visual concepts and events, totaling 15,000 images with 75,000 questions. Figure 5 draws the histogram of number of tokens in VQG questions, where the average question length is 6 tokens. Figure 6 visualizes the n-gram distribution (with n=6) of questions in the three VQG datasets[8]. Table 2 shows the statistics of the crowdsourcing task.

3.3 $\boldsymbol{Captions_{Bing-5000}}$

The word frequencies of questions about the $\boldsymbol{VQG_{Bing-5000}}$ dataset indicate that this dataset

[8]Please refer to our web page on `http://research.microsoft.com/en-us/downloads` to get a link to a dynamic visualization and statistics of all n-gram sequences.

# all images	**15,000**
# questions per image	**5**
# all workers participated	308
Max # questions written by one worker	6,368
Average work time per worker (sec)	106.5
Median work time per worker (sec)	23.0
Average payment per question (cents)	6.0

Table 2: Statistics of crowdsourcing task, aggregating all three datasets.

is substantially different from the MS COCO dataset. Human evaluation results of a recent work (Tran et al., 2016) further confirms the significant image captioning quality degradation on out-of-domain data. To further explore this difference, we crowdsourced 5 captions for each image in the $VQG_{Bing-5000}$ dataset using the same prompt as used to source the MS COCO captions. We call this new dataset $\boldsymbol{Captions_{Bing-5000}}$. Table 3 shows the results of testing the state-of-the-art MSR captioning system on the $\boldsymbol{Captions_{Bing-5000}}$ dataset as compared to the MS COCO dataset, measured by the standard BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014) metrics. The wide gap in the results further confirms that indeed the $\boldsymbol{VQG_{Bing-5000}}$ dataset covers a new class of images; we hope the availability of this new dataset will encourage including more diverse domains for image captioning.

BLEU		**METEOR**	
Bing	*MS COCO*	*Bing*	*MS COCO*
0.101	0.291	0.151	0.247

Table 3: Image captioning results

Together with this paper we are releasing an extended set of VQG dataset to the community. We hope that the availability of this dataset will encourage the research community to tackle more end-goal oriented vision & language tasks.

4 Models

In this Section we present several generative and retrieval models for tackling the task of VQG. For all the forthcoming models we use the VGGNet (Simonyan and Zisserman, 2014) architecture for computing deep convolutional image features. We primarily use the 4096-dimensional output the last fully connected layer ($fc7$) as the input to the generative models.

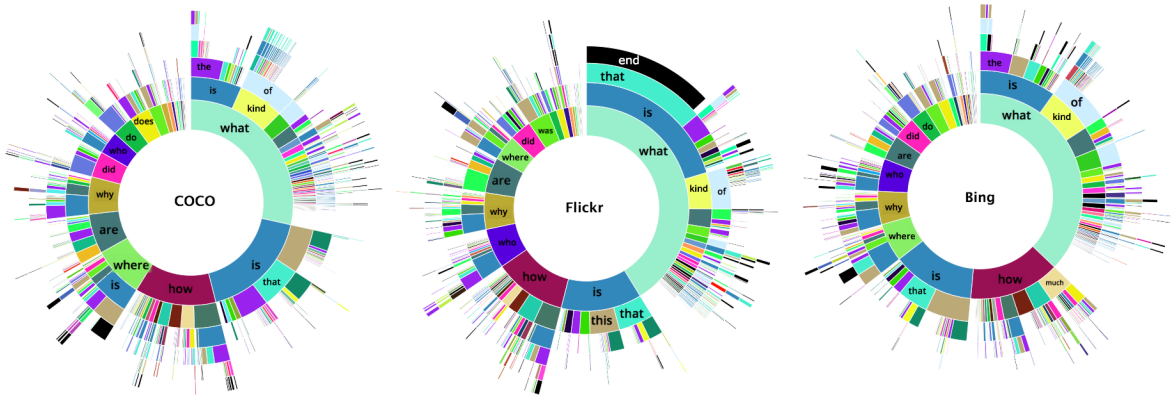


Figure 6: VQG N-gram sequences. 'End' token distinguishes natural ending with n-gram cut-off.

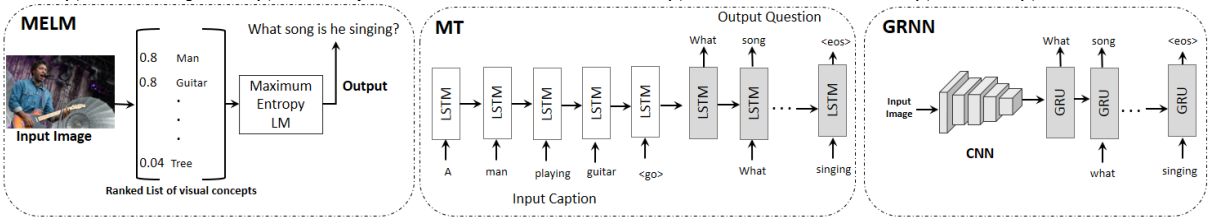


Figure 7: Three different generative models for tackling the task of VQG.

4.1 Generative Models

Figure 7 represents an overview of our three generative models. The MELM model (Fang et al., 2014) is a pipeline starting from a set of candidate word probabilities which are directly trained on images, which then goes through a maximum entropy (ME) language model. The MT model is a Sequence2Sequence translation model (Cho et al., 2014; Sutskever et al., 2014) which directly translates a description of an image into a question, where we used the MS COCO captions and $Captions_{Bing-5000}$ as the source of translation. These two models tended to generate less coherent sentences, details of which can be found in the supplementary material. We obtained the best results by using an end-to-end neural model, GRNN, as follows.

Gated Recurrent Neural Network (GRNN): This generation model is based on the state-of-the-art multimodal Recurrent Neural Network model used for image captioning (Devlin et al., 2015; Vinyals et al., 2015). First, we transform the $fc7$ vector to a 500-dimensional vector which serves as the initial recurrent state to a 500-dimensional Gated Recurrent Unit (GRU). We produce the output question one word at a time using the GRU, until we hit the end-of-sentence token. We train the GRU and the transformation matrix jointly, but we do not back-propagate the CNN due to the size of the training data. The neural network is trained using Stochastic Gradient Descent with early stopping, and decoded using a beam search of size 8. The vocabulary consists of all words seen 3 or more times in the training, which amounts to 1942 unique tokens in the full training set. Unknown words are mapped to to an `<unk>` token during training, but we do not allow the decoder to produce this token at test time.

4.2 Retrieval Methods

Retrieval models use the caption of a nearest neighbor training image to label the test image (Hodosh et al., 2013; Devlin et al., 2015; Farhadi et al., 2010; Ordonez et al., 2011). For the task of image captioning, it has been shown that up to 80% of the captions generated at test time by a near state-of-the-art generation approach (Vinyals et al., 2015) were exactly identical to the training set captions, which suggests that reusing training annotations can achieve good results. Moreover, basic nearest neighbor approaches to image captioning on the MS COCO dataset are shown to outperform generation models according to automatic metrics (Devlin et al., 2015). The performance of retrieval models of course depends on the diversity of the dataset.

We implemented several retrieval models customized for the task of VQG. As the first step, we compute K nearest neighbor images for each test image using the $fc7$ features to get a candidate pool. We obtained the most competitive results by setting K dynamically, as opposed to the earlier

				
Q.	Explosion	Hurricane	Rain Cloud	Car Accident
Human	- What caused this explosion? - Was this explosion an accident?	- What caused the damage to this city? - What happened to this place?	- Are those rain clouds? - Did it rain?	- Did the drivers of this accident live through it? - How fast were they going?
GRNN	- How much did the fire cost? - What is being burned here?	- What happened to the city? - What caused the fall?	- What kind of clouds are these? - Was there a bad storm?	- How did the car crash? - What happened to the trailer?
KNN	- What caused this fire?	- What state was this earthquake in?	- Did it rain?	- Was anybody hurt in this accident?
Caption	- A train with smoke coming from it.	- A pile of dirt.	- Some clouds in a cloudy day.	- A man standing next to a motorcycle.

Table 4: Sample generations by different systems on $VQG_{bing-5000}$, in order: Human$_{consensus}$ and Human$_{random}$, GRNN$_{bing}$ and GRNN$_{all}$, KNN+min$_{bleu-all}$, MSR captions. Q is the query-term.

works which fix K throughout the testing. We observed that candidate images beyond a certain distance made the pool noisy, hence, we establish a parameter called $max\text{-}distance$ which is an upper bound for including a neighbor image in the pool. Moreover, our experiments showed that if there exists a very similar image to the test image, the candidate pool can be ignored and that test image should become the only candidate[9]. For addressing this, we set a $min\text{-}distance$ parameter. All these parameters were tuned on the corresponding validation sets using the Smoothed-BLEU (Lin and Och, 2004) metric against the human reference questions.

Given that each image in the pool has five questions, we define the one-best question to be the question with the highest semantic similarity[10] to the other four questions. This results in a pool of K candidate questions. The following settings were used for our final retrieval models:

– **1-NN**: Set K=1, which retrieves the closest image and emits its one-best.

– **K-NN+min**: Set K=30 with $max\text{-}distance = 0.35$, and $min\text{-}distance = 0.1$. Among the 30 candidate questions (one-best of each image), find the question with the highest similarity to the rest of the pool and emit that: we compute the textual similarity according the two metrics, Smoothed-BLEU and Average-Word2Vec (gensim)[11].

Table 4 shows a few example images along with the generations of our best performing systems. For more examples please refer to the web page of the project.

5 Evaluation

While in VQG the set of possible questions is not limited, there is consensus among the natural questions (discussed in Section 3.1) which enables meaningful evaluation. Although human evaluation is the ideal form of evaluation, it is important to find an automatic metric that strongly correlates with human judgment in order to benchmark progress on the task.

5.1 Human Evaluation

The quality of the evaluation is in part determined by how the evaluation is presented. For instance,

[9] At test time, the frequency of finding a train set image with $distance \leq 0.1$ is 7.68%, 8.4% and 3.0% in COCO, Flickr and Bing datasets respectively.

[10] We use BLEU to compute textual similarity. This process eliminates outlier questions per image.

[11] Average-Word2Vec refers to the sentence-level textual similarity metric where we compute the cosine similarity between two sentences by averaging their word-level Word2Vec (Mikolov et al., 2013) vector representations. Here we use the GenSim software framework (Řehůřek and Sojka, 2010).

it is important for the human judges to see various system hypotheses at the same time in order to give a calibrated rating. We crowdsourced our human evaluation on AMT, asking three crowd workers to each rate the quality of candidate questions on a three-point semantic scale.

5.2 Automatic Evaluation

The goal of automatic evaluation is to measure the similarity of system-generated question hypotheses and the crowdsourced question references. To capture n-gram overlap and textual similarity between hypotheses and references, we use standard Machine Translation metrics, BLEU (Papineni et al., 2002) and METEOR (Denkowski and Lavie, 2014). We use BLEU with equal weights up to 4-grams and default setting of METEOR version 1.5. Additionally we use ΔBLEU (Galley et al., 2015) which is specifically tailored towards generation tasks with diverse references, such as conversations. ΔBLEU requires rating per reference, distinguishing between the quality of the references. For this purpose, we crowd-sourced three human ratings (on a scale of 1-3) per reference and used the majority rating.

The pairwise correlational analysis of human and automatic metrics is presented in Table 6, where we report on Pearson's r, Spearman's ρ and Kendall's τ correlation coefficients. As this table reveals, ΔBLEU strongly correlates with human judgment and we suggest it as the main evaluation metric for testing a VQG system. It is important to note that BLEU is also very competitive with ΔBLEU, showing strong correlations with human judgment. Hence, we recommend using BLEU for any further benchmarking and optimization purposes. BLEU can also be used as a proxy for ΔBLEU for evaluation purposes whenever rating per reference are not available.

5.3 Results

In this section, we present the human and automatic metric evaluation results of the models introduced earlier. We randomly divided each VQG-5000 dataset into train (50%), val (25%) and test (25%) sets. In order to shed some light on differences between our three datasets, we present the evaluation results separately on each dataset in Table 5. Each model (Section 4.2) is once trained on all train sets, and once trained only on its corresponding train set (represented as X in the results table). For quality control and further insight on the task, we include two human annotations among our models: 'Human$_{consensus}$' (the same as one-best) which indicates the consensus human annotation on the test image and 'Human$_{random}$' which is a randomly chosen annotation among the five human annotations.

It is quite interesting to see that among the human annotations, Human$_{consensus}$ achieves consistently higher scores than Human$_{random}$. This further verifies that there is indeed a common intuition about what is the most natural question to ask about a given image. As the results of human evaluation in Table 5 shows, GRNN$_{all}$ performs the best as compared with all the other models in 2/3 of runs. All the models achieve their best score on $VQG_{COCO-5000}$, which was expected given the less diverse set of images. Using automatic metrics, the GRNN$_X$ model outperforms other models according to all three metrics on the $VQG_{Bing-5000}$ dataset. Among retrieval models, the most competitive is K-NN+min_bleu_all, which performs the best on $VQG_{COCO-5000}$ and $VQG_{Flickr-5000}$ datasets according to BLEU and ΔBLEU score. This further confirms our effective retrieval methodology for including $min\text{-}distance$ and n-gram overlap similarity measures. Furthermore, the boost from 1-NN to K-NN models is considerable according to both human and automatic metrics. It is important to note that none of the retrieval models beat the GRNN model on the Bing dataset. This additionally shows that our Bing dataset is in fact more demanding, making it a meaningful challenge for the community.

6 Discussion

We introduced the novel task of 'Visual Question Generation', where given an image, the system is tasked with asking a natural question. We provide three distinct datasets, each covering a variety of images. The most challenging is the Bing dataset, requiring systems to generate questions with event-centric concepts such as 'cause', 'event', 'happen', etc., from the visual input. Furthermore, we show that our Bing dataset presents challenging images to the state-of-the-art captioning systems. We encourage the community to report their system results on the Bing test dataset and according to the ΔBLEU automatic metric. All the datasets will be released to the public[12].

This work focuses on developing the capabil-

[12]Please find Visual Question Generation under `http://research.microsoft.com/en-us/downloads`.

		$Human_{consensus}$	$Human_{random}$	$GRNN_{X}$	$GRNN_{all}$	$1\text{-}NN_{bleu-X}$	$1\text{-}NN_{gensim-X}$	$K\text{-}NN+min_{bleu-X}$	$K\text{-}NN+min_{gensim-X}$	$1\text{-}NN_{bleu-all}$	$1\text{-}NN_{gensim-all}$	$K\text{-}NN+min_{bleu-all}$	$K\text{-}NN+min_{gensim-all}$
						Human Evaluation							
	Bing	2.49	2.38	1.35	**1.76**	1.72	1.72	1.69	1.57	1.72	1.73	1.75	1.58
	COCO	2.49	2.38	1.66	1.94	1.81	1.82	1.88	1.64	1.82	1.82	**1.96**	1.74
	Flickr	2.34	2.26	1.24	**1.57**	1.44	1.44	1.54	1.28	1.46	1.46	1.52	1.30
						Automatic Evaluation							
BLEU	Bing	87.1	83.7	**12.3**	11.1	9.0	9.0	11.2	7.9	9.0	9.0	11.8	7.9
	COCO	86.0	83.5	13.9	14.2	11.0	11.0	19.1	11.5	10.7	10.7	**19.2**	11.2
	Flickr	84.4	83.6	9.9	9.9	7.4	7.4	10.9	5.9	7.6	7.6	**11.7**	5.8
MET.	Bing	62.2	58.8	**16.2**	15.8	14.7	14.7	15.4	14.7	14.7	14.7	15.5	14.7
	COCO	60.8	58.3	18.5	18.5	16.2	16.2	**19.7**	17,4	15.9	15.9	19.5	17.5
	Flickr	59.9	58.6	14.3	**14.9**	12.3	12.3	13.6	12.6	12.6	12.6	14.6	13.0
ΔBLEU	Bing	63.38	57.25	**11.6**	10.8	8.28	8.28	10.24	7.11	8.43	8.43	11.01	7.59
	COCO	60.81	56.79	12.45	12.46	9.85	9.85	16.14	9.96	9.78	9.78	**16.29**	9.96
	Flickr	62.37	57.34	9.36	9.55	6.47	6.47	9.49	5.37	6.73	6.73	**9.8**	5.26

Table 5: Results of evaluating various models according to different metrics. X represents training on the corresponding dataset in the row. Human score per model is computed by averaging human score across multiple images, where human score per image is the median rating across the three raters.

	METEOR	**BLEU**	**ΔBLEU**
r	**0.916** (4.8e-27)	0.915 (4.6e-27)	0.915 (5.8e-27)
ρ	0.628 (1.5e-08)	0.67 (7.0e-10)	**0.702** (5.0e-11)
τ	0.476 (1.6e-08)	0.51 (7.9e-10)	**0.557** (3.5e-11)

Table 6: Correlations of automatic metrics against human judgments, with p-values in parentheses.

ity to ask relevant and to-the-point questions, a key intelligent behavior that an AI system should demonstrate. We believe that VQG is one step towards building such a system, where an engaging question can naturally start a conversation. To continue progress on this task, it is possible to increase the size of the training data, but we also expect to develop models that will learn to generalize to unseen concepts. For instance, consider the examples of system errors in Table 7, where visual features can be enough for detecting the specific set of objects in each image, but the system cannot make sense of the combination of previously unseen concepts. Another natural future extension of this work is to include question generation within a conversational system (Sordoni et al., 2015; Li et al., 2016), where the context and conversation history affect the types of questions being asked.

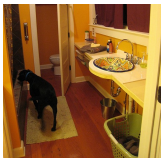
		
Human	- How long did it take to make that ice sculpture?	- Is the dog looking to take a shower?
GRNN	- How long has he been hiking?	- Is this in a hotel room?
KNN	- How deep was the snow?	- Do you enjoy the light in this bathroom?

Table 7: Examples of errors in generation. The rows are $Human_{consensus}$, $GRNN_{all}$, and $KNN+min_{bleu-all}$.

Acknowledgment

We would like to thank the anonymous reviewers for their invaluable comments. We thank Larry Zitnick and Devi Parikh for their helpful discussions regarding this work, Rebecca Hanson for her great help in data collection, Michel Galley for his guidelines on evaluation, and Bill Dolan for his valuable feedback throughout this work.

References

Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2014. 2d human pose estimation: New benchmark and state of the art analysis. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual question answering. In *International Conference on Computer Vision (ICCV)*.

Lee Becker, Sumit Basu, and Lucy Vanderwende. 2012. Mind the gap: Learning to choose gaps for question generation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 742–751, Montréal, Canada, June. Association for Computational Linguistics.

Yu-Wei Chao, Zhan Wang, Yugeng He, Jiaxuan Wang, and Jia Deng. 2015. HICO: A benchmark for recognizing human-object interactions in images. In *Proceedings of the IEEE International Conference on Computer Vision*.

Jianfu Chen, Polina Kuznetsova, David Warren, and Yejin Choi. 2015. Déjà image-captions: A corpus of expressive descriptions in repetition. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 504–514, Denver, Colorado, May–June. Association for Computational Linguistics.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 100–105, Beijing, China, July. Association for Computational Linguistics.

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. 2014. Long-term recurrent convolutional networks for visual recognition and description. *CoRR*, abs/1411.4389.

Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June.

Hao Fang, Saurabh Gupta, Forrest N. Iandola, Rupesh Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C. Platt, C. Lawrence Zitnick, and Geoffrey Zweig. 2014. From captions to visual concepts and back. *CoRR*, abs/1411.4952.

Ali Farhadi, Mohsen Hejrati, Mohammad Amin Sadeghi, Peter Young, Cyrus Rashtchian, Julia Hockenmaier, and David Forsyth. 2010. Every picture tells a story: Generating sentences from images. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 15–29, Berlin, Heidelberg. Springer-Verlag.

Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. 2015. A survey of current datasets for vision and language research. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 207–213, Lisbon, Portugal, September. Association for Computational Linguistics.

Michel Galley, Chris Brockett, Alessandro Sordoni, Yangfeng Ji, Michael Auli, Chris Quirk, Margaret Mitchell, Jianfeng Gao, and Bill Dolan. 2015. deltaBLEU: A discriminative metric for generation tasks with intrinsically diverse targets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 445–450, Beijing, China, July. Association for Computational Linguistics.

Haoyuan Gao, Junhua Mao, Jie Zhou, Zhiheng Huang, Lei Wang, and Wei Xu. 2015. Are you talking to a machine? dataset and methods for multilingual image question answering. *CoRR*, abs/1505.05612.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California, June. Association for Computational Linguistics.

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *J. Artif. Int. Res.*, 47(1):853–899, May.

Ting-Hao Huang, Francis Ferraro, Nasrin Mostafazadeh, Ishan Misra, Aishwarya Agrawal, Jacob Devlin, Ross B. Girshick, Xiaodong He, Pushmeet Kohli, Dhruv Batra, C. Lawrence Zitnick, Devi Parikh, Lucy Vanderwende, Michel Galley, and Margaret Mitchell. 2016. Visual storytelling.

In *Proceedings of NAACL 2016*. Association for Computational Linguistics.

Joseph Jordania. 2006. *Who Asked the First Question? The Origins of Human Choral Singing, Intelligence, Language and Speech*. Logos.

Igor Labutov, Sumit Basu, and Lucy Vanderwende. 2015. Deep questions without deep understanding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

David Lindberg, Fred Popowich, John Nesbit, and Phil Winne. 2013. Generating natural language questions to support learning on-line. In *Proceedings of the 14th European Workshop on Natural Language Generation*, pages 105–114, Sofia, Bulgaria, August. Association for Computational Linguistics.

Mateusz Malinowski and Mario Fritz. 2014. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems 27*, pages 1682–1690.

Karen Mazidi and Rodney D. Nielsen. 2014. Linguistic considerations in automatic question generation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 321–326, Baltimore, Maryland, June. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Ruslan Mitkov and Le An Ha. 2003. Computer-aided generation of multiple-choice tests. In Jill Burstein and Claudia Leacock, editors, *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 17–22.

Vicente Ordonez, Girish Kulkarni, and Tamara L. Berg. 2011. Im2text: Describing images using 1 million captioned photographs. In *Neural Information Processing Systems (NIPS)*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Pustejovsky, P. Hanks, R. Sauri, A. See, R. Gaizauskas, A. Setzer, D. Radev, B. Sundheim, D. Day, L. Ferro, and M. Lazo. 2003. The TIMEBANK corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656, Lancaster, March.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. `http://is.muni.cz/publication/884893/en`.

Mengye Ren, Ryan Kiros, and Richard Zemel. 2015. Question answering about images using visual semantic embeddings. In *Deep Learning Workshop, ICML 2015*.

Marcus Rohrbach, Sikandar Amin, Mykhaylo Andriluka, and Bernt Schiele. 2012. A database for fine grained activity detection of cooking activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, IEEE, June.

K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 196–205, Denver, Colorado, May–June. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Kenneth Tran, Xiaodong He, Lei Zhang, Jian Sun, Cornelia Carapcea, Chris Thrasher, Chris Buehler, and Chris Sienkiewicz. 2016. Rich image captioning in the wild. In *Proceedings of Deep Vision Workshop at CVPR 2016*. IEEE, June.

Lucy Vanderwende, Arul Menezes, and Chris Quirk. 2015. An amr parser for english, french, german, spanish and japanese and a new amr-annotated corpus. Proceedings of NAACL 2015, June.

Lucy Vanderwende. 2008. The importance of being important: Question generation. In *In Workshop on the Question Generation Shared Task and Evaluation Challenge*, Arlington, VA.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *Proceedings the 2015 Conference of the North American Chapter of the Association for Computational Linguistics – Human Language Technologies (NAACL HLT 2015)*, pages 1494–1504, Denver, Colorado, June.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Computer Vision and Pattern Recognition*.

John H Wolfe. 1976. Automatic question generation from text-an aid to independent study. In *ACM SIGCUE Outlook*, volume 10, pages 104–112. ACM.

Yuanjun Xiong, Kai Zhu, Dahua Lin, and Xiaoou Tang. 2015. Recognize complex events from static images by fusing deep channels. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June.

Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas J. Guibas, and Li Fei-Fei. 2011a. Action recognition by learning bases of action attributes and parts. In *International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November.

Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas J. Guibas, and Li Fei-Fei. 2011b. Human action recognition by learning bases of action attributes and parts. In *International Conference on Computer Vision (ICCV)*, Barcelona, Spain, November.

Yuke Zhu, Oliver Groth, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual7w: Grounded question answering in images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, IEEE.

Physical Causality of Action Verbs in Grounded Language Understanding

Qiaozi Gao[†] **Malcolm Doering**[‡*] **Shaohua Yang**[†] **Joyce Y. Chai**[†]
[†]Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
[‡]Department of Systems Innovation, Osaka University, Toyonaka, Osaka, Japan
{gaoqiaoz, yangshao, jchai}@msu.edu
doering.malcolm.robert@irl.sys.es.osaka-u.ac.jp

Abstract

Linguistics studies have shown that action verbs often denote some *Change of State (CoS)* as the result of an action. However, the causality of action verbs and its potential connection with the physical world has not been systematically explored. To address this limitation, this paper presents a study on *physical causality* of action verbs and their implied changes in the physical world. We first conducted a crowdsourcing experiment and identified eighteen categories of physical causality for action verbs. For a subset of these categories, we then defined a set of detectors that detect the corresponding change from visual perception of the physical environment. We further incorporated physical causality modeling and state detection in grounded language understanding. Our empirical studies have demonstrated the effectiveness of causality modeling in grounding language to perception.

1 Introduction

Linguistics studies have shown that action verbs often denote some *change of state (CoS)* as the result of an action, where the *change of state* often involves an attribute of the direct object of the verb (Hovav and Levin, 2010). For example, the result of "slice a pizza" is that the state of the object (pizza) changes from one big piece to several smaller pieces. This change of state can be perceived from the physical world. In Artificial Intelligence (Russell and Norvig, 2010), decades of research on planning, for example, back to the early days of the STRIPS planner (Fikes and Nilsson, 1971), have defined action schemas to capture the change of state caused by a given action. Based on action schemas, planning algorithms can be applied to find a sequence of actions to achieve a goal state (Ghallab et al., 2004). The state of the physical world is a very important notion and changing the state becomes a driving force for agents' actions. Thus, motivated by linguistic literature on action verbs and AI literature on action representations, in our view, modeling change of physical state for action verbs, in other words, *physical causality*, can better connect language to the physical world.

Although this kind of physical causality has been described in linguistic studies (Hovav and Levin, 2010), a detailed account of potential causality that could be denoted by an action verb is lacking. For example, in VerbNet (Schuler, 2005) the semantic representation for various verbs may indicate that a change of state is involved, but it does not provide the specifics associated with the verb's meaning (e.g., to what attribute of its *patient* the changes might occur).

To address this limitation, we have conducted an empirical investigation on verb semantics from a new angle of how they may change the state of the physical world. As the first step in this investigation, we selected a set of action verbs from a cooking domain and conducted a crowd-sourcing study to examine the potential types of causality associated with these verbs. Motivated by linguistics studies on typology for gradable adjectives, which also have a notion of change along a scale (Dixon and Aikhenvald, 2006), we developed a set of eighteen main categories to characterize physical causality. We then defined a set of change-of-state detectors focusing on visual perception. We further applied two approaches, a knowledge-driven approach and a learning-based approach, to incorporate causality modeling in

[*]This work was conducted at Michigan State University where the author received his MS degree.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1814–1824,
Berlin, Germany, August 7-12, 2016.

grounded language understanding. Our empirical results have demonstrated that both of these approaches achieve significantly better performance in grounding language to perception compared to previous approaches (Yang et al., 2016).

2 Related Work

The notion of *causality* or *causation* has been explored in psychology, linguistics, and computational linguistics from a wide range of perspectives. For example, different types of causal relations such as causing, enabling, and preventing (Goldvarg and Johnson-Laird, 2001; Wolff and Song, 2003) have been studied extensively as well as their linguistic expressions (Wolff, 2003; Song and Wolff, 2003; Neeleman et al., 2012) and automated extraction of causal relations from text (Blanco et al., 2008; Mulkar-Mehta et al., 2011; Radinsky et al., 2012; Riaz and Girju, 2014). Different from these previous works, this paper focuses on the physical causality of action verbs, in other words, change of state in the physical world caused by action verbs as described in (Hovav and Levin, 2010). This is motivated by recent advances in computer vision, robotics, and grounding language to perception and action.

A recent trend in computer vision has started looking into intermediate representations beyond lower-level visual features for action recognition, for example, by incorporating object affordances (Koppula et al., 2013) and causality between actions and objects (Fathi and Rehg, 2013). Fathi and Rehg (2013) have borken down detection of actions to detection of state changes from video frames. Yang and colleagues (2013; 2014) have developed an object segmentation and tracking method to detect state changes (or, in their terms, consequences of actions) for action recognition. More recently, Fire and Zhu (2015) have developed a framework to learn perceptual causal structures between actions and object statuses in videos.

In the robotics community, as robots' low-level control systems are often pre-programmed to handle (and thus execute) only primitive actions, a high-level language command will need to be translated to a sequence of primitive actions in order for the corresponding action to take place. To make such translation possible, previous works (She et al., 2014a; She et al., 2014b; Misra et al., 2015; She and Chai, 2016) explicitly model verbs with predicates describing the resulting states of actions. Their empirical evaluations have demonstrated how incorporating resulting states into verb representations can link language with underlying planning modules for robotic systems. These results have motivated a systematic investigation on modeling physical causality for action verbs.

Although recent years have seen an increasing amount of work on grounding language to perception (Yu and Siskind, 2013; Walter et al., 2013; Liu et al., 2014; Naim et al., 2015; Liu and Chai, 2015), no previous work has investigated the link between physical causality denoted by action verbs and the change of state visually perceived. Our work here intends to address this limitation and examine whether the causality denoted by action verbs can provide top-down information to guide visual processing and improve grounded language understanding.

3 Modeling Physical Causality for Action Verbs

3.1 Linguistics Background on Action Verbs

Verb semantics have been studied extensively in linguistics (Pustejovsky, 1991; Levin, 1993; Baker et al., 1998; Kingsbury and Palmer, 2002). Particularly, for action verbs (such as *run*, *throw*, *cook*), Hovav and Levin (Hovav and Levin, 2010) propose that they can be divided into two types: ***manner verbs*** that "specify as part of their meaning a manner of carrying out an action" (e.g., *nibble, rub, scribble, sweep, flutter, laugh, run, swim*), and ***result verbs*** that "specify the coming about of a result state" (e.g., *clean, cover, empty, fill, chop, cut, melt, open, enter*). Result verbs can be further classified into three categories: *Change of State* verbs, which denote a change of state for a property of the verb's object (e.g. "to warm"); *Inherently Directed Motion* verbs, which denote movement along a path in relation to a landmark object (e.g. "to arrive"); and *Incremental Theme* verbs, which denote the incremental change of volume or area of the object (e.g. "to eat") (Levin and Hovav, 2010). In this work, we mainly focus on result verbs. Unlike Hovav and Levin's definition of *Change of State* verbs, we use the term *change of state* in a more general way such that the location, volume, and area of an object are part of its state.

Previous linguistic studies have also shown that result verbs often specify movement along a scale (Hovav and Levin, 2010), i.e., they are

verbs of scalar change. A scale is "a set of points on a particular dimension (e.g. height, temperature, cost)". In the case of verbs, the dimension is an attribute of the object of the verb. For example, "John cooled the coffee" means that the temperature attribute of the object *coffee* has decreased. Kennedy and McNally give a very detailed description of scale structure and its variations (Kennedy and McNally, 2005). Interestingly, gradable adjectives also have their semantics defined in terms of a scale structure. Dixon and Aikhenvald have defined a typology for adjectives which include categories such as Dimension, Color, Physical Property, Quantification, and Position (Dixon and Aikhenvald, 2006). The connection between gradable adjectives and result verbs through scale structure motivates us to use the Dixon typology as a basis to define our categorization of causality for verbs.

In summary, previous linguistic literature has provided abundant evidence and discussion on change of state for action verbs. It has also provided extensive knowledge on potential dimensions that can be used to categorize change of state as described in this paper.

3.2 A Crowd-sourcing Study

Motivated by the above linguistic insight, we have conducted a pilot study to examine the feasibility of causality modeling using a small set of verbs which appear in the TACoS corpus (Regneri et al., 2013). This corpus is a collection of natural language descriptions of actions that occur in a set of cooking videos. This is an ideal dataset to start with since it contains mainly descriptions of physical actions. Possibly because most actions in the cooking domain are goal-directed, a majority of the verbs in TACoS denote results of actions (changes of state) which can be observed in the world.

More specifically, we chose ten verbs (*clean, rinse, wipe, cut, chop, mix, stir, add, open, shake*)) based on the criteria that they occur relatively frequently in the corpus and take a variety of different objects as their *patient*. We paired each verb with three different objects in the role of *patient*. Nouns (e.g., *cutting board, dish, counter, knife, hand, cucumber, beans, leek, eggs, water, break, bowl, etc.*) were chosen based on the criteria that they represent objects dissimilar to each other, since we hypothesize that the change of state indicated by the verb will differ depending on the object's features.

Each verb-noun pair was presented to turkers via Amazon Mechanical Turk (AMT) and they were asked to describe (by text) the changes of state that occur to the object as a result of the verb. The descriptions were collected under two conditions: (1) without showing the corresponding video clips (so turkers would have to use their imagination of the physical situation) and (2) showing the corresponding video clips. For each condition and each verb-noun pair, we collected 30 turkers' responses, which resulted in a total of 1800 natural language responses describing change of state.

3.3 Categorization of Change of State

Based on Dixon and Aikhenvald's typology for adjectives (Dixon and Aikhenvald, 2006) and turkers' responses, we identified a categorization to characterize causality, as shown in Table 1. This categorization is also driven by the expectation that these attributes can be potentially recognized from the physical world by artificial agents. The first column specifies the type of state change and the second column specifies specific attributes related to the type. The third column specifies the particular value associated with the attribute, e.g., it could be a binary categorization on whether a change happens or not (i.e., *changes*), or a direction along a scale (i.e., *increase/decrease*), or a specific value (i.e., *specific* such as "five pieces"). In total, we have identified eighteen causality categories corresponding to eighteen attributes as shown in Table 1.

An important motivation of modeling physical causality is to provide guidance for visual processing. Our hypothesis is that once a language description is given together with its corresponding visual scene, potential causality of verbs or verb-noun pairs can trigger some visual detectors associated with the scene. This can potentially improve grounded language understanding (e.g., grounding nouns to objects in the scene). Next we give a detailed account on these visual detectors and their role in grounded language understanding.

4 Visual Detectors based on Physical Causality

The changes of state associated with the eighteen attributes can be detected from the physical world

Type	Attribute	Attribute Value
Dimension	Size, length, volume	Changes, increases, decreases, specific
	Shape	Changes, specific (cylindrical, flat, etc.)
Color/Texture	Color	Appear, disappear, changes, mix, separate, specific (green, red, etc.)
	Texture	Changes, specific (slippery, frothy, etc.)
Physical Property	Weight	Increase, decrease
	Flavor, smell	Changes, intensifies, specific
	Solidity	Liquefies, solidifies, specific
	Wetness	Becomes wet(ter), dry(er)
	Visibility	Appears, disappears
	Temperature	Increases, decreases
	Containment	Becomes filled, emptied, hollow
	Surface Integrity	A hole or opening appears
Quantification	Number of pieces	Increases, one becomes many, decreases, many become one
Position	Location	Changes, enter/exit container, specific
	Occlusion	Becomes covered, uncovered
	Attachment	Becomes detached
	Presence	No longer present, becomes present
	Orientation	Changes, specific

Table 1: Categorization of physical causality.

Attribute	Rule-based Detector	Refined Rule-based Detector
Attachment / NumberOfPieces	Multiple object tracks merge into one, or one object track breaks into multiple.	Multiple tracks merge into one.
		One track breaks into multiple.
Presence / Visibility	Object track appears or disappears.	Object track appears.
		Object track disappears.
Location	Object's final location is different from the initial location.	Location shifts upwards.
		Location shifts downwards.
		Location shifts rightwards.
		Location shifts leftwards.
Size	Object's x-axis length or y-axis length is different from the initial values.	Object's x-axis length increases.
		Object's x-axis length decreases.
		Object's y-axis length increases.
		Object's y-axis length decreases.

Table 2: Causality detectors applied to *patient* of a verb.

using various sensors. In this paper, we only focus on attributes that can be detected by visual perception. More specifically, we chose the subset: *Attachment*, *NumberOfPieces*, *Presence*, *Visibility*, *Location*, *Size*. They are chosen because: 1) according to the pilot study, they are highly correlated with our selected verbs; and 2) they are relatively easy to be detected from vision.

Corresponding to these causality attributes, we defined a set of rule-based detectors as shown in Table 2. These in fact are very simple detectors, which consist of four major detectors and a refined set that distinguishes directions of state change. These visual detectors are specifically applied to the potential objects that may serve as *patient* for a verb to identify whether certain changes of state occur to these objects in the visual scene.

5 Verb Causality in Grounded Language Understanding

In this section, we demonstrate how verb causality modeling and visual detectors can be used together for grounded language understanding. As shown in Figure 1, given a video clip V of human action and a parallel sentence S describing the action, our goal is to ground different semantic roles of the verb (e.g., *get*) to objects in the video. This is similar to the grounded semantic role labeling task (Yang et al., 2016). Here, we focus on a set of four semantic roles {*agent*, *patient*, *source*, *destination*}. We also assume that we have object and hand tracking results from video data. Each object in the video is represented by a track, which is a series of bounding boxes across video frames. Thus, given a video clip and a parallel sentence, the task is to ground semantic roles of the verb $\lambda_1, \lambda_2, \ldots, \lambda_k$ to object (or hand) tracks $\gamma_1, \gamma_2, \ldots, \gamma_n$, in the video.[1] We applied two approaches to this problem.

[1]For manipulation actions, the *agent* is almost always one of the human's hands (or both hands). So we constrain the grounding of the *agent* role to hand tracks, and constrain the grounding of the other roles to object tracks.

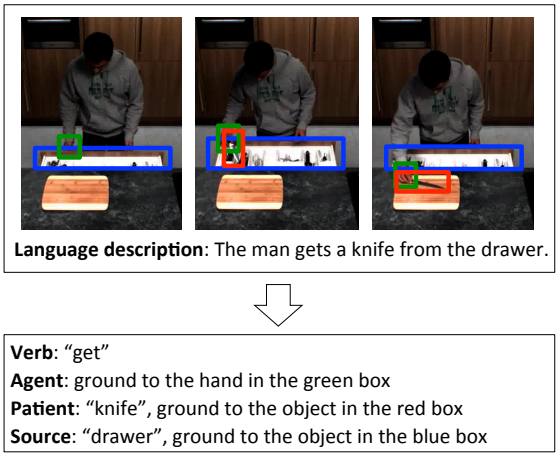


Figure 1: Grounding semantic roles of the verb *get* in the sentence: *the man gets a knife from the drawer*.

5.1 Knowledge-driven Approach

We intend to establish that the knowledge of physical causality for action verbs can be acquired directly from the crowd and such knowledge can be coupled with visual detectors for grounded language understanding.

Acquiring Knowledge. To acquire knowledge of verb causality, we collected a larger dataset of causality annotations based on sentences from the TACoS Multilevel corpus (Rohrbach et al., 2014), through crowd-sourcing on Amazon Mechanical Turk. Annotators were shown a sentence containing a verb-patient pair (e.g., "The person **chops** the **cucumber** into slices on the cutting board"). And they were asked to annotate the change of state that occurred to the *patient* as a result of the verb by choosing up to three options from the 18 causality attributes. Each sentence was annotated by three different annotators.

This dataset contains 4391 sentences, with 178 verbs, 260 nouns, and 1624 verb-noun pairs. After summarizing the annotations from three different annotators, each sentence is represented by a 18-dimension causality vector. In the vector, an element is 1 if at least two annotators labeled the corresponding causality attribute as true, 0 otherwise. For 83% of all the annotated sentences, at least one causality attribute was agreed on by at least two people.

From the causality annotation data, we can extract a *verb causality vector* $\mathbf{c}(\mathrm{v})$ for each verb by averaging all causality vectors of the sentences that contain this verb v.

Applying Knowledge. Since the collected causality knowledge was only for the *patient*, we first look at the grounding of *patient*. Given a sentence containing a verb v and its *patient*, we want to ground the *patient* to one of the object tracks in the video clip. Suppose we have the causality knowledge, i.e., $\mathbf{c}(\mathrm{v})$, for the verb. For each candidate track in the video, we can generate a causality detection vector $\mathbf{d}(\gamma_i)$, using the predefined causality detectors. A straightforward way is to ground the *patient* to the object track whose causality detection results has the best coherence with the causality knowledge of the verb. The coherence is measured by the cosine similarity between $\mathbf{c}(\mathrm{v})$ and $\mathbf{d}(\gamma_i)$.[2]

Since objects in other semantic roles often have relations with the *patient* during the action, once we have grounded the *patient*, we can use it as an anchor point to ground the other three semantic roles. To do this, we define two new detectors for grounding each role as shown in Table 3. These detectors are designed using some common sense knowledge, e.g., *source* is likely to be the initial location of the *patient*; *destination* is likely to be the final location of the *patient*; *agent* is likely to be the hand that touches the *patient*. With these new detectors, we simply ground a role to the object (or hand) track that has the largest number of positive detections from the corresponding detectors.

It is worth noting that although currently we only acquired knowledge for verbs that appear in the cooking domain, the same approach can be extended to verbs in other domains. The detectors associated with attributes are expected to remain the same. The significance of this knowledge-driven method is that, once you have the causality knowledge of a verb, it can be directly applied to any domain without additional training.

5.2 Learning-based Approach

Our second approach is based on learning from training data. A key requirement for this approach is the availability of annotated data where the arguments of a verb are already correctly grounded to the objects in the visual scene. Then we can learn the association between detected causality

[2] In the case that not every causality attribute has a corresponding detector, we need to first condense $\mathbf{c}(\mathrm{v})$ to the same dimensionality with $\mathbf{d}(\gamma_i)$.

Semantic Role	Rule-based Detector
Source	Patient track appears within its bounding box.
	Its track is overlapping with the patient track at the initial frame.
Destination	Patient track disappears within its bounding box.
	Its track is overlapping with the patient track at the final frame.
Agent	Its track is overlapping with the patient track when the patient track appears or disappears.
	Its track is overlapping with the patient track when the patient track starts moving or stops moving.

Table 3: Causality detectors for grounding *source*, *destination*, and *agent*.

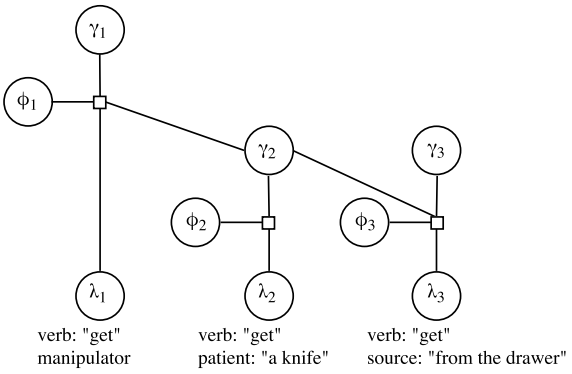


Figure 2: The CRF factor graph of the sentence: *the man gets a knife from the drawer*.

attributes and verbs. We use Conditional Random Field (CRF) to model the semantic role grounding problem. In this approach, causality detection results are used as features in the model.

An example CRF factor graph is shown in Figure 2. The structure of CRF graph is created based on the extracted semantic roles, which already abstracts away syntactic variations such as active/passive constructions. This CRF model is similar to the ones in (Tellex et al., 2011) and (Yang et al., 2016), where $\phi_1, \ldots, \phi_4$ are binary random variables, indicating whether the grounding is correct. In the learning stage, we use the following objective function:

$$p(\Phi|\lambda_1, \ldots, \lambda_k, \gamma_1, \ldots, \gamma_k, \mathrm{v}) = \frac{1}{Z} \prod_i \Psi_i(\phi_i, \lambda_i, \gamma_1, \ldots, \gamma_k, \mathrm{v}) \quad (1)$$

where Φ is the binary random vector $[\phi_1, \ldots, \phi_k]$, and v is the verb. Z is the normalization constant. Ψ_i is the potential function that takes the following log-linear form:

$$\Psi_i(\phi_i, \lambda_i, \Gamma, \mathrm{v}) = \exp\left(\sum_l w_l f_l(\phi_i, \lambda_i, \Gamma, \mathrm{v})\right) \quad (2)$$

where f_l is a feature function, w_l is feature weight to be learned, and $\Gamma = [\gamma_1, \ldots, \gamma_k]$ are the groundings. In our model, we use the following features:

1. Joint features between a track label of γ_i and a word occurrence in λ_i.

2. Joint features between each of the causality detection results and a verb v. Causality detection includes all the detectors in Table 2 and Table 3. Note that the causality detectors shown in Table 3 capture relations between groundings of different semantic roles.

During learning, gradient ascent with L2 regularization is used for parameter learning.

Compared to (Tellex et al., 2011) and (Yang et al., 2016), a key difference in our model is the incorporation of causality detectors. These previous works (Tellex et al., 2011; Yang et al., 2016) apply geometric features, for example, to capture relations, distance, and relative directions between grounding objects. These geometric features can be noisy. In our model, features based on causality detectors are motivated and informed by the underlying causality models for corresponding action verbs.

In the inference step, we want to find the most probable groundings. Given a video clip and its parallel sentence, we fix the Φ to be true, and search for groundings $\gamma_1, \ldots, \gamma_k$ that maximize the probability as in Equation 1. To reduce the search space we apply beam search to ground in the following order: *patient*, *source*, *destination*, *agent*.

5.3 Experiments and Results

We conducted our experiments using the dataset from (Yang et al., 2016). This dataset was developed from a subset of the TACoS corpus (Regneri et al., 2013). It contains a set of video clips paired with natural language descriptions related to two cooking tasks "cutting cucumber" and "cutting bread". Each task has 5 videos showing how different people perform the same task, and each of these videos was split into pairs of video clips and corresponding sentences. For each video clip, objects are annotated with bounding boxes, tracks,

	All	***take***	***put***	***get***	***cut***	***open***	***wash***	***slice***	***rinse***	***place***	***peel***	***remove***
# Instances	279	58	15	47	29	6	28	13	29	29	10	15
With Ground-truth Track Labels												
Label Matching	67.7	70.7	46.7	72.3	69.0	16.7	85.7	69.2	82.8	37.9	90.0	60.0
Yang et al., 2016	84.6	93.2	91.7	93.6	77.8	80.0	93.5	86.7	90.0	66.7	80.0	38.9
VC-Knowledge	**89.6***	**94.8**	73.3	**100***	**93.1**	**83.3**	**100**	92.3	**96.6**	58.6	90.0	73.3*
VC-Learning	**90.3***	**94.8**	**86.7**	**100***	**93.1**	**83.3**	89.3	92.3	**96.6**	**75.9**	80.0	66.7*
Without Track Labels												
Label Matching	9.0	12.1	13.3	2.1	10.3	16.7	3.6	7.7	10.3	10.3	20.0	6.7
Yang et al., 2016	24.5	11.9	8.3	17.0	50.0	10.0	29.0	40.0	40.0	0	60.0	11.1
VC-Knowledge	**60.2***	**82.8***	**60.0***	**87.2***	**58.6**	**50.0**	**39.3**	**46.2**	**41.4**	**48.3***	10.0	**40.0**
VC-Learning	**71.7***	**91.4***	33.3	**87.2***	**72.4**	**83.3***	**46.4**	**84.6***	**51.7**	**65.5***	**80.0**	**60.0***

Table 4: Grounding accuracy on *patient* role

	Overall	**Agent**	**Patient**	**Source**	**Destination**
Number of Instances	644	279	279	51	35
With Ground-truth Track Labels					
Label Matching	66.3	68.5	67.7	41.2	74.3
Yang et al., 2016	84.2	86.4	84.6	72.6	81.6
VC-Knowledge	**86.8**	**89.3**	**89.6***	**60.8**	82.9
VC-Learning	**88.2***	**88.2**	**90.3***	**76.5**	88.6
Without Track Labels					
Label Matching	33.5	66.7	9.0	7.8	2.9
Yang et al., 2016	48.2	86.1	24.5	15.7	13.2
VC-Knowledge	**69.9***	**89.6**	**60.2***	**45.1***	**25.7**
VC-Learning	**75.0***	**87.1**	**71.7***	**41.2***	**54.3***

Table 5: Grounding accuracy on four semantic roles

and labels (e.g. "cucumber, cutting board" etc). For each sentence, the semantic roles of a verb are extracted using Propbank (Kingsbury and Palmer, 2002) definitions and each of them is annotated with the ground truth groundings in terms of the object tracks in the corresponding video clip. We selected the 11 most frequent verbs (*get, take, wash, cut, rinse, slice, place, peel, put, remove, open*) and the 4 most frequent explicit semantic roles (*agent, patient, source, destination*) in this evaluation. In total, this dataset includes 977 pairs of video clips and corresponding sentences, and 1096 verb-patient occurrences.

We compare our knowledge-driven approach (*VC-Knowledge*) and learning-based approach (*VC-Learning*) with the following two baselines.

Label Matching. This method simply grounds the semantic role to the track whose label matches the word phrase. If there are multiple matching tracks, it will randomly choose one of them. If there is no matching track, it will randomly select one from all the tracks.

Yang et al., 2016. This work studies grounded semantic role labeling. The evaluation data from this work is used in this paper. It is a natural baseline for comparison.

To evaluate the learning-based approaches such as *VC-Learning* and *(Yang, et al., 2016)*, 75% of video clips with corresponding sentences were randomly sampled as the training set. The remaining 25% were used as the test set. For approaches which do not need training such as *Label Matching* and *VC-Knowledge*, we used the same test set to report their results.

The results of the *patient* role grounding for each verb are shown in Table 4. The results of grounding all four semantic roles are shown in Table 5. The scores in bold are statistically significant ($p < 0.05$) compared to the *Label Matching* method. The scores with an asterisk ($*$) are statistically significant ($p < 0.05$) compared to *(Yang et al., 2016)*.

As it can be difficult to obtain labels for the track, especially when the vision system encounters novel objects, we further conducted several experiments assuming we do not know the labels for the object tracks. In this case, only geometric information of tracked objects is available. Table 4 and Table 5 also include these results.

From the grounding results, we can see that the causality modeling has shown to be very effective in grounding semantic roles. First of all, both the knowledge-driven approach and the learning-based approach outperform the two baselines. In

	All	*take*	*put*	*get*	*cut*	*open*	*wash*	*slice*	*rinse*	*place*	*peel*	*remove*
VC-Knowledge	89.6	94.8	73.3	100	93.1	83.3	100	92.3	96.6	58.6	90.0	73.3
P-VC-Knowledge	89.9	96.6	73.3	100	96.6	66.7	100	92.3	96.6	65.5	90.0	60.0

Table 6: Grounding accuracy on *patient* role using predicted causality knowledge.

particular, our knowledge-driven approach (*VC-Knowledge*) even outperforms the trained model (*Yang et al., 2016*). Our learning-based approach (*VC-Learning*) achieves the best overall performance. In the learning-based approach, causality detection results can be seen as a set of intermediate visual features. The reason that our learning-based approach significantly outperforms the similar model in (*Yang et al., 2016*) is that the causality categorization provides a good guideline for designing intermediate visual features. These causality detectors focus on the changes of state of objects, which are more robust than the geometric features used in (*Yang et al., 2016*).

In the setting of no object recognition labels, *VC-Knowledge* and *VC-Learning* also generate significantly better grounding accuracy than the two baselines. This once again demonstrates the advantage of using causality detection results as intermediate visual features. All these results illustrate the potential of causality modeling for grounded language understanding.

The results in Table 5 also indicate that grounding *source* or *destination* is more difficult than grounding *patient* or *agent*. One reason could be that *source* and *destination* do not exhibit obvious change of state as a result of action, so their groundings usually depend on the correct grounding of other roles such as *patient*.

Since automated tracking for this TACoS dataset is notably difficult due to the complexity of the scene and the lack of depth information, our current results are based on annotated tracks. But object tracking algorithms have made significant progress in recent years (Yang et al., 2013; Milan et al., 2014). We intend to apply our algorithms with automated tracking on real scenes in the future.

6 Causality Prediction for New Verbs

While various methods can be used to acquire causality knowledge for verbs, it may be the case that during language grounding, we do not know the causality knowledge for every verb. Furthermore, manual annotation/acquisition of causality knowledge for all verbs can be time-consuming. In this section, we demonstrate that the existing causality knowledge for some seed verbs can be used to predict causality for new verbs of which we have no knowledge.

We formulate the problem as follows. Suppose we have causality knowledge for a set of seed verbs as training data. Given a new verb, whose causality knowledge is not known, our goal is to predict the causality attributes associated with this new verb. Although the causality knowledge is unknown, it is easy to compute Distributional Semantic Models (DSM) for this verb. Then our goal is to find the causality vector $\mathbf{c}'$ that maximizes

$$\arg\max_{\mathbf{c}'} p(\mathbf{c}'|\mathbf{v}), \tag{3}$$

where $\mathbf{v}$ is the DSM vector for the verb v. The usage of DSM vectors is based on our hypothesis that the textual context of a verb can reveal its possible causality information. For example, the contextual words "pieces" and "halves" may indicate the CoS attribute "NumberOfPieces" for the verb "cut".

We simplify the problem by assuming that the causality vector $\mathbf{c}'$ takes binary values, and also assuming the independence between different causality attributes. Thus, we can formulate this task as a group of binary classification problems: predicting whether a particular causality attribute is positive or negative given the DSM vector of a verb. We apply logistic regression to train a separate classifier for each attribute. Specifically, for the features of a verb, we use the Distributional Memory (*typeDM*) (Baroni and Lenci, 2010) vector. The class label indicates whether the corresponding attribute is associated with the verb.

In our experiment we chose six attributes to study: *Attachment*, *NumberOfPieces*, *Presence*, *Visibility*, *Location*, and *Size*. For each one of the eleven verbs in the grounding task, we predict its causality knowledge using classifiers trained on all other verbs (i.e., 177 verbs in training set). To evaluate the predicted causality vectors, we applied them in the knowledge-driven approach (*P-VC-Knowledge*). Grounding results were compared with the same method using the causality knowledge collected via crowd-sourcing. Ta-

ble 6 shows the grounding accuracy on the *patient* role for each verb. For most verbs, using the predicted knowledge achieves very similar performance compared to using the collected knowledge. The overall grounding accuracy of using the predicted knowledge on all four semantic roles is only 0.3% lower than using the collected knowledge. This result demonstrates that physical causality of action verbs, as part of verb semantics, can be learned through Distributional Semantics.

7 Conclusion

This paper presents, to the best of our knowledge, the first attempt that explicitly models the physical causality of action verbs. We have applied causality modeling to the task of grounding semantic roles to the environment using two approaches: a knowledge-based approach and a learning-based approach.

Our empirical evaluations have shown encouraging results for both approaches. When annotated data is available (in which semantic roles of verbs are grounded to physical objects), the learning-based approach, which learns the associations between verbs and causality detectors, achieves the best overall performance. On the other hand, the knowledge-based approach also achieves competitive performance (even better than previous learned models), without any training. The most exciting aspect about the knowledge-based approach is that causality knowledge for verbs can be acquired from humans (e.g., through crowd-sourcing) and generalized to novel verbs about which we have not yet acquired causality knowledge.

In the future, we plan to build a resource for modeling physical causality for action verbs. As object recognition and tracking are undergoing significant advancements in the computer vision field, such a resource together with causality detectors can be immediately applied for any applications that require grounded language understanding.

Acknowledgments

This work was supported in part by the National Science Foundation (IIS-1208390 and IIS-1617682) and DARPA under the SIMPLEX program through UCLA (N66001-15-C-4035). The authors would like to thank anonymous reviewers for valuable comments and suggestions.

References

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics.

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

Eduardo Blanco, Nuria Castell, and Dan I Moldovan. 2008. Causal relation extraction. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*.

Robert MW Dixon and Alexandra Y Aikhenvald. 2006. *Adjective Classes: A Cross-linguistic Typology*. Explorations in Language and Space C. Oxford University Press.

Alahoum Fathi and James M Rehg. 2013. Modeling actions through state changes. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2579–2586. IEEE.

Richard E. Fikes and Nils J. Nilsson. 1971. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 608–620, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Amy Fire and Song-Chun Zhu. 2015. Learning perceptual causality from video. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(2):23.

Malik Ghallab, Dana Nau, and Paolo Traverso. 2004. *Automated planning: theory & practice*. Elsevier.

Eugenia Goldvarg and Philip N Johnson-Laird. 2001. Naive causality: A mental model theory of causal meaning and reasoning. *Cognitive science*, 25(4):565–610.

Malka Rappaport Hovav and Beth Levin. 2010. Reflections on Manner / Result Complementarity. *Lexical Semantics, Syntax, and Event Structure*, pages 21–38.

Christopher Kennedy and Louise McNally. 2005. Scale structure and the semantic typology of gradable predicates. *Language*, 81(2)(0094263):345–381.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC2002)*.

Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. 2013. Learning human activities and object affordances from rgb-d videos. *The International Journal of Robotics Research*, 32(8):951–970.

Beth Levin and Malka Rappaport Hovav. 2010. Lexicalized scales and verbs of scalar change. In *46th Annual Meeting of the Chicago Linguistics Society*.

Beth Levin. 1993. *English verb classes and alternations: A preliminary investigation*. University of Chicago press.

Changsong Liu and Joyce Y. Chai. 2015. Learning to mediate perceptual differences in situated human-robot dialogue. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI15)*, pages 2288–2294, Austin, TX.

Changsong Liu, Lanbo She, Rui Fang, and Joyce Y. Chai. 2014. Probabilistic labeling for efficient referential grounding based on collaborative discourse. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 13–18, Baltimore, MD.

Anton Milan, Stefan Roth, and Kaspar Schindler. 2014. Continuous energy minimization for multitarget tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(1):58–72.

Dipendra Kumar Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 992–1002, Beijing, China, July. Association for Computational Linguistics.

Rutu Mulkar-Mehta, Christopher Welty, Jerry R Hoobs, and Eduard Hovy. 2011. Using granularity concepts for discovering causal relations. In *Proceedings of the FLAIRS conference*.

Iftekhar Naim, Young C. Song, Qiguang Liu, Liang Huang, Henry Kautz, Jiebo Luo, and Daniel Gildea. 2015. Discriminative unsupervised alignment of natural language instructions with corresponding video segments. In *Proceedings of NAACL HLT 2015*, pages 164–174, Denver, Colorado, May–June. Association for Computational Linguistics.

Ad Neeleman, Hans Van de Koot, et al. 2012. The linguistic expression of causation. *The Theta System: Argument Structure at the Interface*, page 20.

J Pustejovsky. 1991. The syntax of event structure. *Cognition*, 41(1-3):47–81.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of the 21st international conference on World Wide Web*, pages 909–918. ACM.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics (TACL)*, 1:25–36.

Mehwish Riaz and Roxana Girju. 2014. In-depth exploitation of noun and verb semantics to identify causation in verb-noun pairs. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDial)*, page 161.

Anna Rohrbach, Marcus Rohrbach, Wei Qiu, Annemarie Friedrich, Manfred Pinkal, and Bernt Schiele. 2014. Coherent multi-sentence video description with variable level of detail. In *Pattern Recognition*, pages 184–195. Springer.

S. Russell and P. Norvig. 2010. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Karin Kipper Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. Ph.D. thesis, University of Pennsylvania.

Lanbo She and Joyce Y. Chai. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Berlin, Germany.

Lanbo She, Yu Cheng, Joyce Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. 2014a. Teaching robots new actions through natural language instructions. In *ROMAN, 2014 IEEE*, Edinburgh, UK, August.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Chai, and Ning Xi. 2014b. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the SIGDIAL 2014 Conference*, Philadelphia, US, June.

Grace Song and Phillip Wolff. 2003. Linking perceptual properties to the linguistic expression of causation. *Language, culture and mind*, pages 237–250.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Matthew R Walter, Sachithra Hemachandra, Bianca Homberg, Stefanie Tellex, and Seth Teller. 2013. Learning semantic maps from natural language descriptions. In *Robotics: Science and Systems*.

Phillip Wolff and Grace Song. 2003. Models of causation and the semantics of causal verbs. *Cognitive Psychology*, 47(3):276–332.

Phillip Wolff. 2003. Direct causation in the linguistic coding and individuation of causal events. *Cognition*, 88(1):1–48.

Yezhou Yang, Cornelia Fermuller, and Yiannis Aloimonos. 2013. Detection of manipulation action consequences (mac). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2563–2570.

Yezhou Yang, Anupam Guha, C Fermuller, and Yiannis Aloimonos. 2014. A cognitive system for understanding human manipulation actions. *Advances in Cognitive Sysytems*, 3:67–86.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y. Chai. 2016. Grounded semantic role labeling. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, San Diego, CA.

Haonan Yu and Jeffrey Mark Siskind. 2013. Grounded language learning from video described with sentences. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 53–63.

Optimizing an Approximation of ROUGE – a Problem-Reduction Approach to Extractive Multi-Document Summarization

Maxime Peyrard and **Judith Eckle-Kohler**
Research Training Group AIPHES and UKP Lab
Computer Science Department, Technische Universität Darmstadt
www.aiphes.tu-darmstadt.de, www.ukp.tu-darmstadt.de

Abstract

This paper presents a problem-reduction approach to extractive multi-document summarization: we propose a reduction to the problem of scoring individual sentences with their ROUGE scores based on supervised learning. For the summarization, we solve an optimization problem where the ROUGE score of the selected summary sentences is maximized. To this end, we derive an approximation of the ROUGE-N score of a set of sentences, and define a principled discrete optimization problem for sentence selection. Mathematical and empirical evidence suggests that the sentence selection step is solved almost exactly, thus reducing the problem to the sentence scoring task. We perform a detailed experimental evaluation on two DUC datasets to demonstrate the validity of our approach.

1 Introduction

Multi-document summarization (MDS) is the task of constructing a summary from a topically related document collection. This paper focuses on the variant of extractive and generic MDS, which has been studied in detail for the news domain using available benchmark datasets from the Document Understanding Conference (DUC) (Over et al., 2007).

Extractive MDS can be cast as a budgeted subset selection problem (McDonald, 2007; Lin and Bilmes, 2011) where the document collection is considered as a set of sentences and the task is to select a subset of the sentences under a length constraint. State-of-the-art and recent works in extractive MDS solve this discrete optimization problem using integer linear programming (ILP) or submodular function maximization (Gillick and Favre, 2009; Mogren et al., 2015; Li et al., 2013b; Kulesza and Taskar, 2012; Hong and Nenkova, 2014). The objective function that is maximized in the optimization step varies considerably in previous work. For instance, Yih et al. (2007) maximize the number of informative words, Gillick and Favre (2009) the coverage of particular concepts, and others maximize a notion of "summary worthiness", while minimizing summary redundancy (Lin and Bilmes, 2011; Kågebäck et al., 2014).

There are also multiple approaches which maximize the evaluation metric for system summaries itself based on supervised Machine Learning (ML). System summaries are commonly evaluated using ROUGE (Lin, 2004), a recall oriented metric that measures the n-gram overlap between a system summary and a set of human-written reference summaries.

The benchmark datasets for MDS can be employed in two different ways for supervised learning of ROUGE scores: either by training a model that assigns ROUGE scores to individual textual units (e.g., sentences), or by performing structured output learning and directly maximizing the ROUGE scores of the created summaries (Nishikawa et al., 2014; Takamura and Okumura, 2010; Sipos et al., 2012). The latter approach suffers both from the limited amount of training data and from the higher complexity of the machine learning models.

In contrast, supervised learning of ROUGE scores for individual sentences can be performed with simple regression models using hundreds of sentences as training instances, taken from a single pair of documents and reference summaries. Extractive MDS can leverage the ROUGE scores of individual sentences in various ways, in particular, as part of an optimization step. In our work, we follow the previously successful approaches to

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1825–1836,
Berlin, Germany, August 7-12, 2016.

extractive MDS using discrete optimization, and make the following contributions:

We provide a theoretical justification and empirical validation for using ROUGE scores of individual sentences as an optimization objective. Assuming that ROUGE scores of individual sentences have been estimated by a supervised learner, we derive an approximation of the ROUGE-N score for a set of sentences from the ROUGE-N scores of the individual sentences in the general case of $N >= 1$.

We use our approximation to define a mathematically principled discrete optimization problem for sentence selection. We empirically evaluate our framework on two DUC datasets, demonstrating the validity of our approximation, as well as its ability to achieve competitive ROUGE scores in comparison to several strong baselines.

Most importantly, the resulting framework reduces the MDS task to the problem of scoring individual sentences with their ROUGE scores. The overall summarization task is converted to two sequential tasks: (i) scoring single sentences, and (ii) selecting summary sentences by solving an optimization problem where the ROUGE score of the selected sentences is maximized.

The optimization objective we propose almost exactly solves (ii), which we justify by providing both mathematical and empirical evidence. Hence, solving the whole problem of MDS is reduced to solving (i).

The rest of this paper is structured as follows: in Section 2, we discuss related work. Section 3 presents our subset selection framework consisting of an approximation of the ROUGE score of a set of sentences, and a mathematically principled discrete optimization problem for sentence selection. We evaluate our framework in Section 4 and discuss the results in Section 5. Section 6 concludes.

2 Related Work

Related to our approach is previous work in extractive MDS that (i) casts the summarization problem as budgeted subset selection, and (ii) employs supervised learning on MDS datasets to learn a scoring function for textual units.

Budgeted Subset Selection Extractive MDS can be formulated as the problem of selecting a subset of textual units from a document collection such that the overall score of the created summary is maximal and a given length constraint is observed. The selection of textual units for the summary relies on their individual scores, assigned by a scoring function which represents aspects of their relevance for a summary. Often, sentences are considered as textual units.

Simultaneously maximizing the relevance scores of the selected units and minimizing their pairwise redundancy given a length constraint is a global inference problem which can be solved using ILP (McDonald, 2007). Several state-of-the-art results in MDS have been obtained by using ILP to maximize the number of relevant concepts in the created summary while minimizing the pairwise similarity between the selected sentences (Gillick and Favre, 2009; Boudin et al., 2015; Woodsend and Lapata, 2012).

Another way to formulate the problem of finding the best subset of textual units is to maximize a submodular function. Maximizing submodular functions is a general technique that uses a greedy optimization algorithm with a mathematical guarantee on optimality (Nemhauser and Wolsey, 1978). Performing summarization in the framework of submodularity is natural because summaries try to maximize the coverage of relevant units while minimizing redundancy (Lin and Bilmes, 2011). However, several different coverage and redundancy functions have been proposed (Lin and Bilmes, 2011; Kågebäck et al., 2014; Yin and Pei, 2015) recently, and there is not yet a clear consensus on which coverage function to maximize.

Supervised Learning Supervised learning using datasets with reference summaries has already been employed in early work on summarization to classify sentences as summary-worthy or not (Kupiec et al., 1995; Aone et al., 1995).

Learning a scoring function for various kinds of textual units has become especially popular in the context of global optimization: scores of textual units, learned from data, are fed into an ILP problem solver to find the subset of sentences with maximal overall score. For example, Yih et al. (2007) score each word in the document cluster based on frequency and position, Li et al. (2013b) learn bigram frequency in the reference summaries, and Hong and Nenkova (2014) learn word importance from a rich set of features.

Closely related to our work are summarization approaches that include a supervised component

which assigns ROUGE scores to individual sentences. For example, Ng et al. (2012), Li et al. (2013a) and Li et al. (2015) all use a regression model to learn ROUGE-2 scores for individual sentences, but use it in different ways for the summarization. While Ng et al. (2012) use the ROUGE scores of sentences in combination with the Maximal Marginal Relevance algorithm as a baseline approach, Li et al. (2013a) use the scores to select the top-ranked sentences for sentence compression and subsequent summarization. Li et al. (2015), in contrast, use the ROUGE scores to re-rank a set of sentences that are output by an optimization step.

While learning ROUGE scores of textual units is widely used in summarization systems, the theoretical background on why this is useful has not been well studied yet. In our work, we present the mathematical and empirical justification for this common practice. In the next section, we start with the mathematical justification.

3 Content Selection Framework

3.1 Approximation of ROUGE-N

Notation: Let $S = \{s_i | i \leq m\}$ be a set of m sentences which constitute a system summary. We use $\rho_N(S)$ or simply $\rho(S)$ to denote the ROUGE-N score of S. ROUGE-N evaluates the n-gram overlap between S and a set of reference summaries (Lin, 2004). Let S^* denote the reference summary and R_N the number of n-gram tokens in S^*. R_N is a function of the summary length in words, in particular, R_1 is the target size of the summary in words. Finally, let $F_S(g)$ denote the number of times the n-gram type g occurs in S. For a single reference summary, ROUGE-N is computed as follows:

$$\rho(S) = \frac{1}{R_N} \sum_{g \in S^*} min(F_S(g), F_{S^*}(g)) \quad (1)$$

For compactness, we use the following notation for any set of sentences X:

$$C_{X,S^*}(g) = min(F_X(g), F_{S^*}(g)) \quad (2)$$

$C_{X,S^*}(g)$ can be understood as the contribution of the n-gram g.

ROUGE-N for a Pair of Sentences: Using this notation, the ROUGE-N score of a set of two sentences a and b can be written as:

$$\rho(a \cup b) = \frac{1}{R_N} \sum_{g \in S^*} min(C_{a \cup b,S^*}(g), F_{S^*}(g)) \quad (3)$$

We observe that $\rho(a \cup b)$ can be expressed as a function of the individual scores $\rho(a)$ and $\rho(b)$:

$$\rho(a \cup b) = \rho(a) + \rho(b) - \epsilon(a \cap b) \quad (4)$$

where $\epsilon(a \cap b)$ is an error correction term that discards overcounted n-grams from the sum of $\rho(a)$ and $\rho(b)$:

$$\epsilon(a \cap b) = \frac{1}{R_N} \sum_{g \in S^*} max(C_{a,S^*}(g) + C_{b,S^*}(g) - F_{S^*}(g), 0) \quad (5)$$

A proof that this error correction is correct is given in appendix A.1.

General Formulation of ROUGE-N: We can extend the previous formulation of ρ to sets of arbitrary cardinality using recursion. If $\rho(S)$ is given for a set of sentences S, and a is a sentence then:

$$\rho(S \cup a) = \rho(S) + \rho(a) - \epsilon(S \cap a) \quad (6)$$

We prove in appendix A.1 that this formula is the ROUGE-N score of $S \cup a$.

Another way to obtain ρ for an arbitrary set S is to adapt the principle of inclusion-exclusion:

$$\rho(S) = \sum_{i=1}^{m} \rho(s_i) + \sum_{k=2}^{m} (-1)^{k+1} \Big(\sum_{1 \leq i_1 \leq \cdots \leq i_k \leq m} \epsilon^{(k)}(s_{i_1} \cap \cdots \cap s_{i_k}) \Big) \quad (7)$$

This formula can be understood as adding up scores of individual sentences, but n-grams appearing in the intersection of two sentences might be overcounted. $\epsilon^{(2)}$ is used to account for these n-grams. But now, n-grams in the intersection of three sentences might be undercounted and $\epsilon^{(3)}$ is used to correct this. Each $\epsilon^{(k)}$ contributes to improving the accuracy by refining the errors made by $\epsilon^{(k-1)}$ for the n-grams appearing in the intersection of k sentences. When $k = |S|$, $\rho(S)$ is exactly the ROUGE-N of S. A rigorous proof and details about $\epsilon^{(k)}$ are provided in appendix A.2.

Approximation of ROUGE-N for a Pair of Sentences: To find a valid approximation of ρ as defined in (7), we first consider the $\rho(a \cup b)$ from equation (3) and then extend it to the general case. When maximizing ρ, scores for sentences are assumed to be given (e.g., estimated by a ML component). We still need to estimate $\epsilon(a \cap b)$, which means, according to (5), to estimate:

$$\sum_{g \in S^*} max(C_{a,S^*}(g) + C_{b,S^*}(g) - F_{S^*}(g), 0) \quad (8)$$

At inference time, neither S^* (the reference summary) nor F_{S^*} (number of occurrences of n-grams in the reference summary) is known.

At this point, we can observe that, similar as for sentence scoring, ϵ can be estimated via a supervised ML component. Such an ML model can easily be trained on the intersections of all sentence pairs in a given training dataset. Hence, we can assume that both the scores for individual sentences and the ϵ are learned empirically from data using ML. As a result, we have pushed all estimation steps into supervised ML components, which leaves the subset selection step fully principled.

However, we found in our experiments that even a simple heuristic yields a decent approximation of ϵ. The heuristic uses the frequency $freq(g)$ of an n-gram g observed in the source documents:

$$\sum_{g \in S^*} max(C_{a,S^*}(g) + C_{b,S^*}(g) - F_{S^*}(g), 0) \approx \sum_{g \in a \cap b} \mathbb{1}[freq(g) \geq \alpha] \quad (9)$$

The threshold α tells us which n-grams are likely to appear in the reference summary, and it is determined by grid-search on the training set. This is penalizing n-grams which appear twice and are likely to occur in the summary. It can be understood as a way of limiting redundancy. In practice, we used $\alpha = 0.3$. However, we experimented with various values of the hyper-parameter α and found that its value has no significant impact as long as it is fairly small (< 0.5). Higher values will ignore too many redundant n-grams and the summary will have a high redundancy.

R_N is known since it is simply the number of n-gram tokens in the summaries. We end up with the following approximation for the pairwise case:

$$\tilde{\rho}(a \cup b) = \rho(a) + \rho(b) - \tilde{\epsilon}(a \cup b), \text{ where}$$
$$\tilde{\epsilon}(a \cup b) = \frac{1}{R_N} \sum_{g \in a \cap b} \mathbb{1}[freq(g) \geq \alpha] \quad (10)$$

General Approximation of ROUGE-N: Now, we can approximate $\rho(S)$ for the general case defined by equation (7). We recall that $\rho(S)$ contains the sum of $\rho(s_i)$, the pairwise error terms $\epsilon^{(2)}(s_i \cap s_j)$, the error terms of three sentences $\epsilon^{(3)}$ and so on.

We can restrict ourselves to the individual sentences and the pairwise error corrections. Indeed, the intersection between more than two sentences is often empty, and accounting for it does not improve the accuracy significantly, but greatly increases the computational cost.

A formulation of ϵ in the case of two sentences has already been defined in (10). Thus, we have an approximation of the ROUGE-N function for any set of sentences that can be computed at inference time:

$$\tilde{\rho}(S) = \sum_{i=1}^{n} \rho(s_i) - \sum_{a,b \in S, a \neq b} \tilde{\epsilon}(a \cap b) \quad (11)$$

We empirically checked the validity of this approximation. For this, we sampled 1000 sets of sentences from source documents of DUC-2003 (sets of 2 to 5 sentences) and compared their $\tilde{\rho}$ score to the real ROUGE-N. We observe a pearson's r correlation ≥ 0.97, which validates $\tilde{\rho}$.

3.2 Discrete Optimization

$\tilde{\rho}$ from equation (11) defines a set function that scores a set of sentences. The task of summarization is now to select the set S^* with maximal $\tilde{\rho}(S^*)$ under a length constraint.

Submodularity: A submodular function is a set function obeying the diminishing returns property: $\forall S \subseteq T$ and a sentence a: $F(S \cup a) - F(S) \geq F(T \cup a) - F(T)$. Submodular functions are convenient because maximization under constraints can be done greedily with a guarantee of the optimality of the solution (Nemhauser et al., 1978).

It has been shown that ROUGE-N is submodular (Lin and Bilmes, 2011) and it is easy to verify that $\tilde{\rho}$ is submodular as well (the proof is given in the supplemental material).

We can therefore apply the greedy maximization algorithm to find a good set of sentences. This has the advantage of being straightforward and fast, however it does not necessarily find the optimal solution.

ILP: A common way to solve a discrete optimization problem is to formulate it as an ILP. It

maximizes (or minimizes) a linear objective function with some linear constraints where the variables are integers. ILP has been well studied and existing tools can efficiently retrieve the exact solution of an ILP problem.

We observe that it is possible to formulate the maximization of $\tilde{\rho}(S)$ as an ILP. Let x be the binary vector whose i-th entry indicates whether sentence i is in the summary or not, $\tilde{\rho}(s_i)$ the scores of sentences, and K the length constraint. We pre-compute the symmetric matrix $\tilde{P}$ where $\tilde{P}_{i,j} = \tilde{\epsilon}(s_i \cap s_j)$ and solve the following ILP:

$$max(\sum_{i=1}^{n} x_i * \tilde{\rho}(s_i) - d\frac{1}{R}\sum_{i \geq j} \alpha_{i,j} * \tilde{P}i, j)$$
$$\sum_{i=1}^{n} x_i * len(s_i) \leq K$$
$$\forall (i,j), \alpha_{i,j} - x_i \leq 0$$
$$\forall (i,j), \alpha_{i,j} - x_j \leq 0$$
$$\forall (i,j), x_i + x_j - \alpha_{i,j} \leq 1$$

d is a damping factor that allows to account for approximation errors. When $d = 0$, the problem becomes the maximization of "summary worthiness" under a length constraint, with "summary worthiness" being defined by $\rho(s_i)$.

In practice, we used a value $d = 0.9$ because we observed that the learner tends to slightly overestimate the ROUGE-N scores of sentences. The mathematical derivation implies $d = 1$, however we can easily adjust for shifts in average scores of sentences from the estimation step by adjusting d. Another option would be to post-process the scores after the estimation step to fix the average and let $d = 1$ in the optimization step. Indeed, if d moves away from 1, we move away from the mathematical framework of ROUGE-N maximization.

If $d \neq 0$, it seems intuitive to interpret the second term as minimizing the summary redundancy, which is in accordance to previous works.

However, in our framework, this term has a precise interpretation:

it maximizes ROUGE-N scores up to the second order of precision, and the ROUGE-N formula itself already induces a notion of "summary worthiness" and redundancy, which we can empirically infer from data via supervised ML for sentence scoring, and a simple heuristic for sentence intersections.

4 Evaluation

We perform three kinds of experiments in order to empirically evaluate our framework: first, we show that our proposed approximation is valid, then we analyze a basic supervised sentence scoring component, and finally we perform an extrinsic evaluation on end-to-end extractive MDS.

In our experiments, we use the DUC datasets from 2002 and 2003 (DUC-02 and DUC-03). We use the variants of ROUGE identified by Owczarzak et al. (2012) as strongly correlating with human evaluation methods: ROUGE-2 recall with stemming and stopwords not removed (giving the best agreement with human evaluation), and ROUGE-1 recall (as the measure with the highest ability to identify the better summary in a pair of system summaries). For DUC-03, summaries are truncated to 100 words, and to 200 words for DUC-02. [1] The truncation is done automatically by ROUGE. [2]

4.1 Framework Validity

Given that sentences receive scores close to their individual ROUGE-N, we presented a function that approximates the ROUGE-N of sets of these sentences and proposed an optimization to find the best scoring set under a length constraint.

To validate our framework empirically, we consider its upper-bound, which is obtained when our ILP/submodular optimizations use the real ROUGE-N scores of the individual sentences, calculated based on the reference summaries. We compare this upper bound to a greedy approach, which simply adds the best scoring sentences one by one to the subset until the length limit is reached, and to the real upper bound for extractive summarization which is determined by solving a maximum coverage problem for n-grams from the reference summary (as it was done by Takamura and Okumura (2010)).

Table 1 shows the results. We observe that ILP-R produces scores close to the reference, thus reducing the problem of extractive summarization to the task of sentence scoring, because the perfect scores induced near perfect extracted summaries in this framework. SBL-R seems less promising than ILP-R because it greedily maximizes a function which ILP-R exactly maximizes. Therefore, we continue our experiments in the following sec-

[1] In the official DUC-03 competitions, summaries of length 665 bytes were expected. Systems could produce different numbers of words. The variation in length has a noticeable impact on ROUGE recall scores.

[2] ROUGE-1.5.5 with the parameters: -n 2 -m -a -l 100 -x -c 95 -r 1000 -f A -p 0.5 -t 0. The length parameter becomes -l 200 for DUC-02.

tions with ILP-R only. However, SBL-R offers a nice trade-off between performance and computation cost. The greedy optimization of SBL-R is noticeably faster than ILP-R.

	DUC-02		DUC-03	
	R1	R2	R1	R2
Greedy	0.597	0.414	0.391	0.148
SBL-R	0.630	0.484	0.424	0.160
ILP-R	0.644	0.495	0.447	0.178
Upper Bound	0.648	0.497	0.452	0.181

Table 1: Upper bound of our framework compared to extractive upper bound.

In practice, the learner will not produce perfect scores. We experimentally validated that with learned scores converging to true scores, the extracted summary converges to the best extractive summary (w.r.t to ROUGE-N). To this end, we simulated approximate learners by artificially randomizing the true scores to end up with lists having various correlations with the true scores. We fed these scores to ILP-R and computed the ROUGE-1 of the generated summaries for an example topic from DUC-2003. Figure 1 displays the expected ROUGE-1 versus the performance of the artificial learner (correlation with true scores of sentences). We observe that, as the learner improves, the generated summaries approach the best ROUGE scoring summary.

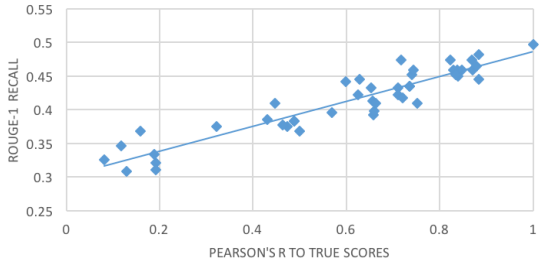


Figure 1: ROUGE-1 of summary against sentence scores correlation with true ROUGE-1 scores of sentences (d30003t from DUC-2003).

4.2 Sentence Scoring

Now we look at the supervised learning component which learns ROUGE-N scores for individual sentences. We know that we can achieve an overall summary ROUGE-N score close to the upper bound, if a learner would be able to learn the scores perfectly. For better understanding the difficulty of the task of sentence scoring, we look at the correlation of the scores produced by a basic learner and the true scores given in a reference dataset.

Model and Features From an existing summarization dataset (e.g. a DUC dataset), a training set can straightforwardly be extracted by annotating each sentence in the source documents with its ROUGE-N score. For each topic in the dataset, this yields a list of sentences and their target score.

To support the claim that learning ROUGE scores for individual sentences is easier than solving the whole summarization task, it is sufficient to choose a basic learner with simple features and little in-domain training data (models are trained on one DUC dataset and evaluated on another). Specifically, we employ a support vector regression (SVR).[3] We use only classical surface-level features to represent sentences (position, length, overlap with title) and combine them with frequency features. The latter include TF*IDF weighting of the terms (similar to Luhn (1958)), the sum of the frequency of the bi-grams in the sentence, as well as the sum of the document frequency (number of source documents in which the n-grams appear) of the terms and bi-grams in a sentence.

We trained two models, R1 and R2 on DUC-02 and DUC-03. For R1, the target score is the ROUGE-1 recall, while R2 learns ROUGE-2 recall.

Correlation Analysis We evaluated our sentence scoring models R1 and R2 by calculating the correlation of the scores produced by R1 and R2 and the true scores given in the DUC-03 data. We compare both models to the true ROUGE-1 and ROUGE-2 scores. In addition, we calculated the correlation of the TF*IDF and LexRank scores, in order to understand how well they would fit into our framework (TF*IDF and LexRank are described in section 4.3).

The results are displayed in Table 2. Even with a basic learner it is possible to learn scores that correlate well with the true ROUGE-N scores, which supports the claim that it is easier to learn scores for individual sentences than to solve the whole problem of summarization. This finding strongly supports our proposed reduction of the extractive MDS problem to the task of learning

[3] We use the implementation in scikit-learn (Pedregosa et al., 2011).

	with ROUGE-1			with ROUGE-2		
	Pearson's r	Kendall's tau	nDCG@15	Pearson's r	Kendall's tau	nDCG@15
TF*IDF	0.923	0.788	0.916	0.607	0.512	0.580
LexRank	0.210	0.120	0.534	0.286	0.178	0.379
model R1	**0.940**	**0.813**	**0.951**	0.653	0.545	0.693
model R2	0.729	0.496	0.891	**0.743**	**0.576**	**0.752**

Table 2: Correlation of different kinds of sentence scores and their true ROUGE-1 and ROUGE-2 scores.

scores for individual sentences, which correlate well with their true ROUGE-N scores.

We observe that TF*IDF correlates surprisingly well with the ROUGE-1 score, which indicates that we can expect a significant performance gain when feeding TF*IDF scores to our optimization framework. LexRank, on the other hand, orders sentences according to their centrality and does not look at individual sentences. Accordingly, we observe a low correlation with the true ROUGE-N scores, and thus LexRank may not benefit from the optimization (which we confirmed in our experiments).

Finally, we observe that there is significant room for improvement regarding ROUGE-2, as well as for Kendall's tau in ROUGE-1 where a more sophisticated learner could produce scores that correlate better with the true scores. The higher the correlation of the sentence scores assigned by a learner and the true scores, the better the summary produced by the subsequent subset selection.

4.3 End-to-End Evaluation

In our end-to-end evaluation on extractive MDS, we use the following baselines for comparison:

- **TF*IDF weighting**: This simple heuristic was introduced by Luhn (1958). Each sentence receives a score from the TF*IDF of its terms. We trained IDFs (Inverse Document Frequencies) on a background corpus [4] to improve the original algorithm.

- **LexRank**: Among other graph-based approaches to summarization (Mani and Bloedorn, 1997; Radev et al., 2000; Mihalcea, 2004), LexRank (Erkan and Radev, 2004) has become the most popular one. A similarity graph $G(V, E)$ is constructed where V is the set of sentences and an edge e_{ij} is drawn between sentences v_i and v_j if and only if the cosine similarity between them is above a given threshold. Sentences are scored according to their PageRank score in G. For our experiments, we use the implementation available in the sumy package.[5]

- **ICSI**: ICSI is a recent system that has been identified as one of the state-of-the-art systems by Hong et al. (2014). It is a global linear optimization framework that extracts a summary by solving a maximum coverage problem considering the most important concepts in the source documents. Concepts are identified as bi-grams and their importance is estimated via their frequency in the source documents. Boudin et al. (2015) released a Python implementation (ICSI sume) that we use in our experiments.

- **SFOUR**: SFOUR is a structured prediction approach that trains an end-to-end system with a large-margin method to optimize a convex relaxation of ROUGE (Sipos et al., 2012). We use the publicly available implementation. [6]

As described in the previous section, two models are trained: R1 and R2. We evaluate both of them in the end-to-end setup with and without our optimization. In the greedy version, sentences are added as long as the summary length is valid.

We apply the optimization for sentence scoring models trained on ROUGE-1 and ROUGE-2 as well. The scoring models are trained on one dataset and evaluated on the other. For the ILP optimization, the damping factor can vary and leads to different performance. We report the best results among few variations. In order to speed-up the ILP step, we propose to limit the search space by only looking at the top K sentences[7] (hence

[4] We used DBpedia long abstract: http://wiki.dbpedia.org/Downloads2015-04.

[5] https://github.com/miso-belica/sumy

[6] http://www.cs.cornell.edu/~rs/sfour/

[7] We used K=50 and observed that a range from K=25 to K=70 yields a good trade-off between computation cost and performance.

the importance of learning a correct ordering as well, like Kendall's tau). This results in a massive speed-up and can even lead to better results as it prunes parts of the noise. Finally, we perform significance testing with the t-test to compare differences between two means.[8]

	DUC-02		DUC-03	
	R1	R2	R1	R2
TFIDF	0.403	0.120	0.322	0.066
LexRank	0.446	0.158	0.354	0.077
ICSI	0.445	0.155	0.375	0.094
SFOUR	0.442	0.181	0.365	0.087
Greedy-R1	0.480	0.115	0.353	0.084
Greedy-R2	0.499	0.132	0.369	0.093
TFIDF+ILP	0.415	0.135	0.335	0.075
R1+ILP	0.509	0.187	0.378	0.101
R2+ILP	**0.516***	**0.192***	**0.379**	**0.102**

Table 3: Impact of the optimization step on sentence subset selection.

Results Table 3 shows the results. The proposed optimization significantly and systematically improves TF*IDF performance as we expected from our analysis in the previous section. This result suggests that using only a frequency signal in source documents is enough to get high scoring summaries, which supports the common belief that frequency is one of the most useful features for generic news summarization. It also aligns well with the strong performance of ICSI, which combines an ILP step with frequency information as well.

The optimization also significantly and systematically improves upon the greedy approach combined with our scoring models. Combining a SVR learner (SVR-1 and SVR-2) and our ILP-R produces results on par with ICSI and sometimes significantly better. SFOUR maximizes ROUGE in an end-to-end fashion, but is outperformed by our framework when using the same training data. The framework is able to reach a competitive performance even with a basic learner. These results again suggest that investigating better learners for sentence scoring might be promising in order to improve the quality of the summaries.

We observe that the model trained on ROUGE-2 is performing better than the model trained on ROUGE-1, although learning the ROUGE-2 scores seems to be harder than learning ROUGE-1 scores (as shown in table 2). However, errors and approximations propagate less easily in ROUGE-2, because the number of bi-grams in the intersection of two given sentences is far less. Hence we conclude that learning ROUGE-2 scores should be put into the focus of future work on improving sentence scoring.

[8]The symbol * indicates that the difference compared to the previous best baseline is significant with $p \leq 0.05$.

5 Discussion

This section discusses our contributions in a broader context.

ROUGE Our subset selection framework performs the task of *content selection*, selecting an unordered set of textual units (sentences for now) for a system summary. The re-ordering of the sentences is left to a subsequent processing step, which accounts for aspects of discourse coherence and readability.

While we justified our choice of ROUGE-1 recall and ROUGE-2 recall as optimization objectives by their strong correlation with human evaluation methods, ROUGE-N has also various drawbacks. In particular, it does not take into account the overall discourse coherence of a system summary (see the supplemental material for examples of summaries generated by our framework).

From a broader perspective, systems that have high ROUGE scores can only be as good as ROUGE is, as a proxy for summary quality. However, as long as systems are evaluated with ROUGE, a natural approach is to develop systems that maximize it.

Should novel automatic evaluation metrics be developed, our approach can still be applied, provided that the new metrics can be expressed as a function of the scores of individual sentences.

Structured Learning Compared to MDS approaches using structured learning, our problem-reduction has the important advantage that it considerably scales-up the available training data by working on sentences instead of documents/summaries pairs. Moreover, the task of sentence scoring is not dependent on arbitrary parameters such as the summary length which are inherently abstracted from the "summary worthiness" of individual textual units.

Error Propagation The first step of the framework is left to a ML component which can only produce approximate scores. Empirical results (in Figure 1 and Table 2) suggest that even with an

imperfect first step, the subsequent optimization is able to produce high scoring summaries. However, it might be insightful to study rigorously and in greater detail the propagation of errors induced by the first step.

Other Metrics This work focused on maximizing ROUGE-N recall because it is a widely acknowledged automatic evaluation metric. ROUGE-N relies on reference summaries which forces us to perform an estimation step. In our framework, we use ML to estimate the individual scores of sentences without using reference summaries.

However, Louis and Nenkova (2013) proposed several alternative evaluation metrics for system summaries which do not need reference summaries. They are based on the properties of the system summary and the source documents alone, and correlate well with human evaluation. Some of them can even reach a correlation with human evaluation similar to the ROUGE-2 recall.

An example of such a metric is the Jensen-Shannon Divergence (JSD) which is a symmetric smoothed version of the Kullback-Leibler divergence. Maximizing JSD can not be solved exactly with an ILP because it can not be factorized into individual sentences. However, applying an efficient greedy algorithm or maximizing a factorizable relaxation might produce strong results as well (for example, a simple greedy maximization of Kullback-Leibler divergence already yields good results (Haghighi and Vanderwende, 2009)).

Future Work In this work, we developed a principled subset selection framework and empirically justified it. We focused on solving the second step of the framework while keeping the machine learning component as simple as possible. Essentially, our framework performs a modularization of the task of MDS, where all characteristics of the data and feature representations are pushed into a separate machine learning module – they should not affect the subsequent optimization step which remains fixed.

The promising results we obtained for summarization with a basic learner (see Section 4.3) encourage future work on plugging in more sophisticated supervised learners in our framework. For example, we plan to incorporate lexical-semantic information in the feature representation and leverage large-scale unsupervised pre-training. This direction is particularly promising because we have shown that we can expect significant performance gains for end-to-end MDS as the sentence scoring component improves.

6 Conclusion

We proposed a problem-reduction approach to extractive MDS, which performs a reduction to the problem of scoring individual sentences with their ROUGE scores based on supervised learning. We defined a principled discrete optimization problem for sentence selection which relies on an approximation of ROUGE. We empirically checked the validity of the approach on standard datasets and observed that even with a basic learner the framework produces promising results. The code for our optimizers is available at `github.com/UKPLab/acl2016-optimizing-rouge`.

Acknowledgments

This work has been supported by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1.

References

Chinatsu Aone, Mary Ellen Okurowski, James Gorlinsky, and Bjornar Larsen. 1995. A Trainable Summarizer with Knowledge Acquired from Robust NLP Techniques. In Inderjeet Mani and Mark T. Maybury, editors, *Advances in Automatic Text Summarization*, pages 68–73. MIT Press, Cambridge, MA, USA.

Florian Boudin, Hugo Mougard, and Benot Favre. 2015. Concept-based Summarization using Integer Linear Programming: From Concept Pruning to Multiple Optimal Solutions. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1914–1918, Lisbon, Portugal.

Günes Erkan and Dragomir R. Radev. 2004. LexRank: Graph-based Lexical Centrality As Salience in Text Summarization. *Journal of Artificial Intelligence Research*, pages 457–479.

Dan Gillick and Benoit Favre. 2009. A Scalable Global Model for Summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, ILP '09, pages 10–18, Boulder, Colorado.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring Content Models for Multi-document Summarization. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 362–370.

Kai Hong and Ani Nenkova. 2014. Improving the Estimation of Word Importance for News Multi-Document Summarization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 712–721, Gothenburg, Sweden.

Kai Hong, John Conroy, benoit Favre, Alex Kulesza, Hui Lin, and Ani Nenkova. 2014. A Repository of State of the Art and Competitive Baseline Summaries for Generic News Summarization. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 1608–1616, Reykjavik, Iceland.

Mikael Kågebäck, Olof Mogren, Nina Tahmasebi, and Devdatt Dubhashi. 2014. Extractive Summarization using Continuous Vector Space Models. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)*, pages 31–39, Gothenburg, Sweden.

Alex Kulesza and Ben Taskar. 2012. Determinantal Point Processes for Machine Learning. *Foundations and Trends in Machine Learning*, 5:123–286.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A Trainable Document Summarizer. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 68–73, Seattle, Washington, USA. Association for Computing Machinery.

Chen Li, Fei Liu, Fuliang Weng, and Yang Liu. 2013a. Document Summarization via Guided Sentence Compression. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 490–500, Seattle, Washington, USA.

Chen Li, Xian Qian, and Yang Liu. 2013b. Using Supervised Bigram-based ILP for Extractive Summarization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1004–1013, Sofia, Bulgaria.

Chen Li, Yang Liu, and Lin Zhao. 2015. Improving Update Summarization via Supervised ILP and Sentence Reranking. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1317–1322, Denver, Colorado.

Hui Lin and Jeff A. Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 510–520, Portland, Oregon.

Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out at ACL*, pages 74–81, Barcelona, Spain.

Annie Louis and Ani Nenkova. 2013. Automatically Assessing Machine Summary Content Without a Gold Standard. *Computational Linguistic*, 39(2):267–300, June.

Hans Peter Luhn. 1958. The Automatic Creation of Literature Abstracts. *IBM Journal of Research Development*, 2:159–165.

Inderjeet Mani and Eric Bloedorn. 1997. Multi-document Summarization by Graph Search and Matching. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, pages 622–628, Providence, Rhode Island. AAAI Press.

Ryan McDonald. 2007. A Study of Global Inference Algorithms in Multi-document Summarization. In *Proceedings of the 29th European Conference on IR Research*, pages 557–564, Rome, Italy. Springer-Verlag.

Rada Mihalcea. 2004. Graph-based Ranking Algorithms for Sentence Extraction, Applied to Text Summarization. In *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ACLdemo '04, page 20, Barcelona, Spain.

Olof Mogren, Mikael Kågebäck, and Devdatt Dubhashi. 2015. Extractive Summarization by Aggregating Multiple Similarities. In *Recent Advances in Natural Language Processing*, pages 451–457, Hissar, Bulgaria.

George L. Nemhauser and Laurence A. Wolsey. 1978. Best Algorithms for Approximating the Maximum of a Submodular Set Function. *Mathematics of Operations Research*, 3(3):177–188.

George L. Nemhauser, Laurence A. Wolsey, and Marschall L. Fisher. 1978. An Analysis of Approximations for Maximizing Submodular Set FunctionsI. *Mathematical Programming*, 14:265–294.

Jun-Ping Ng, Praveen Bysani, Ziheng Lin, Min-Yen Kan, and Chew-Lim Tan. 2012. Exploiting Category-Specific Information for Multi-Document Summarization. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 2093–2108, Mumbai, India.

Hitoshi Nishikawa, Kazuho Arita, Katsumi Tanaka, Tsutomu Hirao, Toshiro Makino, and Yoshihiro Matsuo. 2014. Learning to Generate Coherent Summary with Discriminative Hidden Semi-Markov Model. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1648–1659.

Paul Over, Hoa Dang, and Donna Harman. 2007. DUC in Context. *Information Processing and Management*, 43(6):1506–1520.

Karolina Owczarzak, John M. Conroy, Hoa Trang Dang, and Ani Nenkova. 2012. An Assessment of the Accuracy of Automatic Evaluation in Summarization. In *Proceedings of Workshop on Evaluation Metrics and System Comparison for Automatic Summarization*, pages 1–9, Montreal, Canada.

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. 2000. Centroid-based Summarization of Multiple Documents: Sentence Extraction, Utility-based Evaluation, and User Studies. In *Proceedings of the NAACL-ANLP Workshop on Automatic Summarization*, volume 4, pages 21–30, Seattle, Washington.

Ruben Sipos, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Large-margin Learning of Submodular Summarization Models. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 224–233, Avignon, France.

Hiroya Takamura and Manabu Okumura. 2010. Learning to Generate Summary as Structured Output. In *Proceedings of the 19th ACM international Conference on Information and Knowledge Management*, pages 1437–1440. Association for Computing Machinery.

Kristian Woodsend and Mirella Lapata. 2012. Multiple Aspect Summarization Using Integer Linear Programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, (EMNLP-CoNLL)*, pages 233–243, Jeju Island, Korea.

Wen-tau Yih, Joshua Goodman, Lucy Vanderwende, and Hisami Suzuki. 2007. Multi-document Summarization by Maximizing Informative Content-words. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 1776–1782, Hyderabad, India. Morgan Kaufmann Publishers Inc.

Wenpeng Yin and Yulong Pei. 2015. Optimizing Sentence Modeling and Selection for Document Summarization. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 1383–1389, Buenos Aires, Argentina. AAAI Press.

A Supplemental Material

A.1 Recursive Expression of ROUGE-N

Let $S = \{s_i | i \leq m\}$ and $T = \{t_i | i \leq l\}$ be two sets of sentences, S^* the reference summary, and $\rho(X)$ denote the ROUGE-N score of the set of sentences X. Assuming that $\rho(S)$ and $\rho(T)$ are given, we prove the following recursive formula:

$$\rho(S \cup T) = \rho(S) + \rho(T) - \epsilon(S \cap T) \quad (12)$$

For compactness, we use the following notation as well:

$$C_{X,S^*}(g) = min(F_X(g), F_{S^*}(g)) \quad (13)$$

Proof: We have the following definitions:

$$\rho(S) = \frac{1}{R_N} \sum_{g \in S^*} \tilde{F}_{S,S^*}(g) \quad (14)$$

$$\rho(T) = \frac{1}{R_N} \sum_{g \in S^*} \tilde{F}_{T,S^*}(g) \quad (15)$$

$$\epsilon(S \cap T) = \frac{1}{R_N} \sum_{g \in S^*} max(C_{S,S^*}(g) + C_{T,S^*}(g) - F_{S^*}(g), 0) \quad (16)$$

And by definition of ROUGE, the formula of $S \cup T$:

$$\rho(S \cup T) = \frac{1}{R_N} \sum_{g \in S^*} min(F_{S \cup T}(g), F_{S^*}(g)) \quad (17)$$

In order to prove equation (12), we have to show that the following equation holds:

$$\sum_{g \in S^*} C_{S,S^*}(g) + \sum_{g \in S^*} C_{T,S^*}(g) - \sum_{g \in S^*} max(C_{S,S^*}(g) + C_{T,S^*}(g) - F_{S^*}(g), 0) = \sum_{g \in S^*} min(F_{S \cup T}(g), F_{S^*}(g)) \quad (18)$$

It is sufficient to show:

$$\forall g \in S^*, C_{S,S^*}(g) + C_{T,S^*}(g) - max(C_{S,S^*}(g) + C_{T,S^*}(g) - F_{S^*}(g), 0) = min(F_{S \cup T}(g), F_{S^*}(g)) \quad (19)$$

Let $g \in S^*$ be a n-gram. There are two possibilities:

- $F_S(g) + F_T(g) \leq F_{S^*}(g)$: g appears less times in $S \cup T$ than in the reference summary. It implies: $min(F_{S\cup T}(g), F_{S^*}(g)) = F_{S\cup T}(g) = F_S(g) + F_T(g)$. Moreover, all $F_X(g)$ are positive numbers by definition, and $F_S(g) \leq F_{S^*}(g)$ is equivalent to: $C_{S,S^*}(g) = min(F_S(g), F_{S^*}(g)) = F_S(g)$. Similarly, we have: $C_{T,S^*}(g) = min(F_T(g), F_{S^*}(g)) = F_T(g)$. Since $max(C_{S,S^*}(g) + C_{T,S^*}(g) - F_{S^*}(g), 0) = 0$, the equation (19) holds in this case.

- $F_S(g) + F_T(g) \geq F_{S^*}(g)$: g appears more frequently in $S \cup T$ than in the reference summary. It implies: $min(F_{S\cup T}(g), F_{S^*}(g)) = F_{S^*}(g)$. Here we have: $max(C_{S,S^*}(g) + C_{T,S^*}(g) - F_{S^*}(g), 0) = C_{S,S^*}(g) + C_{T,S^*}(g) - F_{S^*}(g)$, and it directly follows that equation (19) holds in this case as well.

Equation (19) has been proved, which proves (12) as well.

A.2 Expanded Expression of ROUGE-N

Let $S = \{s_i | i \leq m\}$ be a set of sentences and $\rho(S)$ its ROUGE-N score. We prove the following formula:

$$\rho(S) = \sum_{i=1}^{m} \rho(s_i) + \sum_{k=2}^{m} (-1)^{k+1} \Big(\sum_{1 \leq i_1 \leq \cdots \leq i_k \leq m} \epsilon^{(k)}(s_{i_1} \cap \cdots \cap s_{i_k}) \Big) \tag{20}$$

Proof: Let $g \in S^*$ be a n-gram in the reference summary, and $k \in [1, m]$ the number of sentences in which it appears. Specifically, $\exists \{s_{i_1}, \cdots, s_{i_k}\}, \forall s_{i_j} \in \{s_{i_1}, \ldots, s_{i_k}\}, g \in s_{i_j}$. In order to prove the formula (20), we have to find an expression for the $\epsilon^{(k)}$ that gives to g the correct contribution to the formula:

$$\frac{1}{R_N} min(F_S(g), F_{S^*}(g)) \tag{21}$$

First, we observe that g does not appear in the terms that contain the intersection of more than k sentences. Specifically, $\epsilon^{(t)}$ is not affected by g if $t \geq k$. However, g is affected by all the $\epsilon^{(t)}$ for which $t \leq k$.

Given that g appears in the sentences $\{s_{i_1}, \ldots, s_{i_j}\}$, we can determine the score attributed to g by the previous $\epsilon^{(t)}$ ($t \leq k$):

$$S^{(k-1)}(g) = \sum_{s \in \{s_{i_1}, \ldots, s_{i_k}\}} \rho(s) + \sum_{l=2}^{k} (-1)^{(l+1)} \sum_{1 \leq i_1 \leq \cdots \leq i_l \leq k} \epsilon^{(l)}(s_{i_1} \cap \cdots \cap s_{i_l})) \tag{22}$$

Now, g receives the correct contribution to the overall scores if $\epsilon^{(k)}$ is defined as follows:

$$\epsilon^{(k)}(s_{i_1} \cap \cdots \cap s_{i_j}) = \frac{1}{R} \sum_{g \in s_{i_1} \cap \cdots \cap s_{i_j}} min(C_{\{s_{i_1}, \ldots, s_{i_k}\}}(g), F_{S^*}(g)) - S^{(k-1)}(g) \tag{23}$$

Indeed, with this expression for $\epsilon^{(k)}$, the score of g is:

$$S^{(k-1)}(g) + \frac{1}{R_N} min(C_{\{s_{i_1}, \ldots, s_{i_k}\}}(g), F_{S^*}(g)) - S^{(k-1)}(g) \tag{24}$$

Which can be simplified to:

$$\frac{1}{R_N} min(C_{\{s_{i_1}, \ldots, s_{i_k}\}}(g), F_{S^*}(g)) \tag{25}$$

Since g appears only in the sentences $\{s_{i_1}, \ldots, s_{i_k}\}$, $\tilde{F}_{\{s_{i_1}, \ldots, s_{i_k}\}}(g) = F_S(g)$ and it follows that:

$$\frac{1}{R_N} min(C_{\{s_{i_1}, \ldots, s_{i_k}\}}(g), F_{S^*}(g)) = \frac{1}{R_N} min(F_S(g), F_{S^*}(g)) \tag{26}$$

This proves equation (20) because we observe that g will not be affected by any other terms. Every $\epsilon^{(t)}$ for $t \leq k$ including g is counted by $S^{(k-1)}$, and no other terms from $\epsilon^{(k)}$ will affect g because all the other terms $\epsilon^{(k)}$ should contain at least one sentence that is not in $\{s_{i_1}, \ldots, s_{i_k}\}$ and g would not belong to this intersection by definition.

Finally, it has been proved in the appendix A.1 that for $k = 2$, $\epsilon^{(2)}$ has a reduced form:

$$\epsilon^{(2)}(s_a \cap s_b) = \frac{1}{R_N} \sum_{g \in S^*} max(C_{s_a,S^*}(g) + C_{s_b,S^*}(g) - F_{S^*}(g), 0) \tag{27}$$

In the paper, we ignore the terms for $k \geq 2$, therefore we do not search for a reduced form for these terms.

Phrase Structure Annotation and Parsing for Learner English

Ryo Nagata
Konan University
8-9-1 Okamoto, Higashinada
Kobe, Hyogo 658-8501, Japan
nagata-acl @hyogo-u.ac.jp.

Keisuke Sakaguchi
Johns Hopkins University
3400 North Charles Street
Baltimore, MD, 21218, USA
keisuke@cs.jhu.edu

Abstract

There has been almost no work on phrase structure annotation and parsing specially designed for learner English despite the fact that they are useful for representing the structural characteristics of learner English. To address this problem, in this paper, we first propose a phrase structure annotation scheme for learner English and annotate two different learner corpora using it. Second, we show their usefulness, reporting on (a) inter-annotator agreement rate, (b) characteristic CFG rules in the corpora, and (c) parsing performance on them. In addition, we explore methods to improve phrase structure parsing for learner English (achieving an F-measure of 0.878). Finally, we release the full annotation guidelines, the annotated data, and the improved parser model for learner English to the public.

1 Introduction

Learner corpora have been essential for NLP tasks related to learner language such as grammatical error correction. They are normally annotated with linguistic properties. In the beginning, attention was mainly focused on grammatical error annotation (Izumi et al., 2004; Díaz-Negrillo et al., 2009; Dale and Kilgarriff, 2011; Ng et al., 2013). Recently, it has been expanded to grammatical annotation — first, Part-Of-Speech (POS) tagging (Díaz-Negrillo et al., 2009; Nagata et al., 2011) and then syntactic annotation (Kepser et al., 2004; Dickinson and Ragheb, 2009; Ragheb and Dickinson, 2012; Ragheb and Dickinson, 2013); syntactic annotation for learner corpora is now intensively studied. Among a variety of studies, a series of work by Ragheb and Dickinson (Dickinson and Ragheb, 2009; Ragheb and Dickinson, 2012; Ragheb and Dickinson, 2013) is important in that they proposed a dependency annotation scheme, theoretically and empirically evaluated it, and revealed its theoretical problems, which gives a good starting point to those who wish to develop a new annotation scheme for learner corpora. Researchers including Foster (2004) and Ott and Ziai (2010) have even started using dependency-annotated learner corpora to develop dependency parsers for learner language.

Although research on syntactic analysis for learner corpora has been making great progress as noted above, it is not yet complete. There are at least three limitations in the previous work: (i) as far as we are aware, there has been almost no work on phrase structure annotation specially designed for learner corpora; (ii) there are no publicly available learner corpora annotated with syntax; (iii) phrase structure parsing performance on learner English has not yet been reported.

The first limitation is that there exists no phrase structure annotation scheme specially designed for learner English. As related work, Foster (2007a; 2007b) and Foster and Andersen (2009) propose a method for creating a pseudo-learner corpus by artificially generating errors in a native corpus with phrase structures. However, the resulting corpus does not capture various error patterns in learner English.

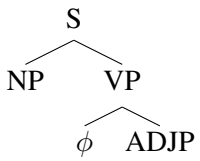
Concerning the second limitation, a corpus greatly increases in value when it is available to the public as has been seen in other domains. Nevertheless, whether dependency or phrase structure, there seems to be no publicly available learner corpora annotated with syntax.

The above two limitations cause the third one that phrase structure parsing performance on leaner English has not yet been reported. For this reason, Cahill (2015) demonstrates how ac-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1837–1847,
Berlin, Germany, August 7-12, 2016.

curately an existing parser performs on a pseudo-learner corpus (section 23 of WSJ with errors artificially generated by Foster and Andersen (2009)'s method). Cahill et al. (2014) show the performance of a phrase structure parser augmented by self-training on students' essays, many of which are presumably written by native speakers of English. Tetreault et al. (2010) partially show phrase structure parsing performance concerning preposition usage in learner English, concluding that it is effective in extracting features for preposition error correction. We need to reveal full parsing performance to be able to confirm that this is true for other syntactic categories and whether or not we should use phrase structure parsing to facilitate related tasks such as grammatical error correction and automated essay scoring.

Here, we emphasize that phrase structure annotation has at least two advantages over dependency annotation[1]. First of all, it can directly encode information about word order. This is particularly important because learner corpora often contain errors in word order. For example, phrase structure parsing will reveal in which phrases errors in word order tend to occur as we will partly do in Sect. 3. Second of all, phrase structure rather abstractly represents syntactic information in terms of phrase-to-phrase relations. This means that the characteristics of learner English are represented by means of phrase-to-phrase relations (e.g., context free grammar (CFG) rules) or even as trees. Take as an example, one of the characteristic trees we found in the corpora we have created:



As we will discuss in Sect. 3, this tree suggests the mother tongue interference that the copula is not necessary in adjective predicates in certain languages. It would be linguistically interesting to reveal what CFG rules we need to add to, or subtract from, the native CFG rule set to be able to generate learner English. This is our primary motivation for this work although our other motivations include developing a parser for learner English.

In view of this background, we address the above problems in this paper. Our contributions are three-fold. First, we present a phrase structure annotation scheme for dealing with learner English consistently and reliably. For this, we propose five principles which can be applied to creating a novel annotation scheme for learner corpora. Second, we evaluate the usefulness of the annotation scheme by annotating learner corpora using it. To be precise, we report on inter-annotator agreement rate and characteristic CFG rules in the corpora, and take the first step to revealing phrase structure parsing performance on learner English. In addition, we explore methods to improve phrase structure parsing for learner English. Finally, we release the full annotation guidelines, the annotated corpora, and the improved parser model to the public.

The rest of this paper is structured as follows. Sect. 2 describes the annotation scheme. Sect. 3 explores the annotated learner corpora. Sect. 4 evaluates parsing performance using it.

2 Phrase Structure Annotation Scheme

2.1 General Principles

The annotation scheme is designed to consistently retrieve the structure in the target text that is closest to the writer's intention. The following are the five principles we created to achieve it:

(P1) Consistency-first principle
(P2) Minimal rule set principle
(P3) Locally superficially-oriented principle
(P4) Minimum edit distance principle
(P5) Intuition principle

(P1) states that the most important thing in our annotation scheme is consistency. It is a trade-off between quality and quantity of information; detailed rules that are too complicated make annotation unmanageable yet they may bring out valuable information in learner corpora. Corpus annotation will be useless if it is inconsistent and unreliable no matter how precisely the rules can describe linguistic phenomena. Therefore, this principle favors consistency over completeness. Once we annotate a corpus consistently, we consider adding further detailed information to it.

(P2) also has to do with consistency. The smaller the number of rules is, the easier it becomes to practice the rules. Considering this, if we have several candidates for describing a new linguistic phenomenon particular to learner English, we will choose the one that minimizes the number of modifications to the existing rule set. Note that

[1]We are not arguing that phrase structure annotation is better than dependency annotation; they both have their own advantages, and thus both should be explored.

this applies to the entire rule set; an addition of a rule may change the existing rule set.

(P3) is used to determine the tag of a given token or phrase. As several researchers (Díaz-Negrillo et al., 2009; Dickinson and Ragheb, 2009; Nagata et al., 2011; Ragheb and Dickinson, 2012) point out, there are two ways of performing annotation, according to either superficial (morphological) or contextual (distributional) evidence. For example, in the sentence **My university life is enjoy.*, the word *enjoy* can be interpreted as a verb according to its morphological form or as an adjective (enjoyable) or a noun (enjoyment) according to its context. As the principle itself construes, our annotation scheme favors superficial evidence over distributional. This is because the interpretation of superficial evidence has much less ambiguity and (P3) can determine the tag of a given token by itself as seen in the above example. Distributional information is also partly encoded in our annotation scheme as we discuss in Subsect. 2.2.

(P4) regulates how to reconstruct a correct form of a given sentence containing errors, which helps to determine its phrase structure. The problem is that often one can think of several candidates as possible corrections, which can become a source of inconsistency. (P4) gives a clear solution to this problem. It selects the one that minimizes the edit distance from the original sentence. Note that the edit distances for deletion, addition, and replacement are one, one, and two (deletion and addition), respectively in our definition.

For the cases to which these four principles do not apply, the fifth and final principle (P5) allows annotators to use their intuition. It should be noted, however, that the five principles apply in the above order to avoid unnecessary inconsistency.

2.2 Annotation Rules

Our annotation scheme is based on the POS-tagging and shallow-parsing annotation guidelines for learner English (Nagata et al., 2011), which in turn are based on the Penn Treebank II-style bracketing guidelines (Bies et al., 1995) (which will be referred to as PTB-II, hereafter). This naturally leads us to adopt the PTB-II tag set in ours; an exception is that we exclude the function tags and null elements from our present annotation scheme for annotation efficiency[2]. Accordingly, we revise the above guidelines to be able to describe phrase structures characteristic of learner English.

The difficulties in syntactic annotation of learner English mainly lie in the fact that grammatical errors appear in learner English. Grammatical errors are often classified into three types as in Izumi et al. (2004): omission, insertion, and replacement type errors. In addition, we include other common error types (word order errors and fragments) in the error types to be able to describe learners' characteristics more precisely. The following discuss how to deal with these five error types based on the five principles.

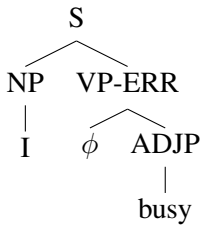
2.2.1 Omission Type Errors

This type of error is an error where a necessary word is missing. For example, some kind of determiner is missing in the sentence **I am student.*

The existing annotation rules in PTB-II can handle most omission type errors. For instance, the PTB-II rule set would parse the above example as "(S (NP I) (VP am (NP student).))." Note that syntactic tags for irrelevant parts are omitted in this example (and hereafter).

A missing head word may be more problematic. Take as an example the sentence **I busy.* where a verb is missing. The omission prevents the rule S $\rightarrow$ NP VP from applying to it. If we created a new rule for every head-omission with no limitation, it would undesirably increase the number of rules, which violates (P2).

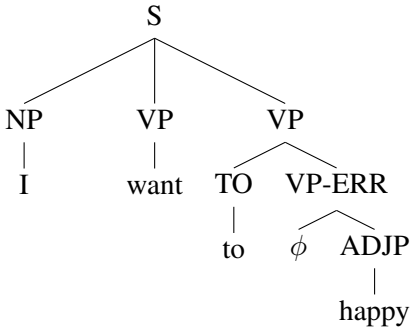
To handle head-omissions, we propose a function tag *-ERR*. It denotes that a head is missing in the phrase in question. The function tag makes it possible to apply the PTB-II rule set to sentences containing head-omissions as in:



We need to reconstruct a correct form of a given sentence to determine whether or not a head word is missing. We use Principle (P4) for solving the problem as discussed in Sect. 2.1. For instance, the sentence **I want to happy.* can be corrected as either *I want to be happy.* (edit distance is one; an addition of a word) or *I want happiness.* (three; two deletions and an addition). Following (P4), we select the first correction that minimizes the edit

[2]We will most likely include them in a future version.

distance, resulting in:



2.2.2 Insertion Type Errors

An insertion type error is an error where an extra word is used incorrectly. For example, the word *about* is an extra word in the sentence **She discussed about it.*

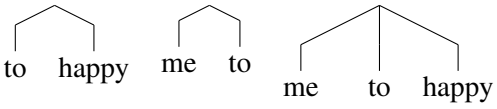
Insertion type errors are more problematic than omission type errors. It is not trivial how to annotate an erroneous extra word. On the one hand, one can argue that the extra word *about* is a preposition from its morphological form. On the other hand, one can also argue that it is not, because the verb *discuss* takes no preposition. As with this example, insertion type errors involve an ambiguity between superficial and distributional categories.

Principles (P2) and (P3) together solve the ambiguity. According to (P3), one should always stick to the superficial evidence. For example, the extra word *about* should be tagged as a preposition. After this, PTB-II applies to the rest of the sentence, which satisfies (P2). As a result, one would obtain the parse "(S (NP She) (VP discussed (PP (IN about) (NP it))))."

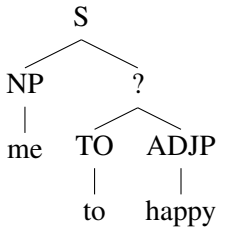
Insertion type errors pose a more vital problem in some cases. Take as an example the sentence **It makes me to happy.* where the word *to* is erroneous. As before, one can rather straightforwardly tag it as a preposition, giving the POS sequence:

**It/PRP makes/VBZ me/PRP to/TO happy/JJ ./.*

However, none of the PTB-II rules applies to the POS sequence *TO JJ* to make a phrase. This means that we have to create a new rule for such cases. There are at most three possibilities of grouping the words in question to make a phrase:



Intuitively, the first one seems to be the most acceptable. To be precise, the second one assumes a postposition, contrary to the English preposition system. The third one assumes a whole new rule generating a phrase from a personal pronoun, a preposition, and an adjective into a phrase. Thus, they cause significant modifications to PTB-II, which violates (P2). In contrast, a preposition normally constitutes a prepositional phrase with another phrase (although not normally with an adjective phrase). Moreover, the first grouping would produce for the rest of the words the perfect phrase structure corresponding to the correct sentence without the preposition *to*:



which satisfies (P2) unlike the second and third ones. Accordingly, we select the first one.

All we have to do now is to name the phrase *to happy*. There is an ambiguity between PP and ADJP, both of which can introduce the parent S. The fact that a preposition constitutes a prepositional phrase with another phrase leads us to select PP for the phrase. Furthermore, the tag of a phrase is normally determined by the POS of one of the immediate constituents, if any, that is entitled to be a head (i.e., the headedness). Considering this, we select PP in this case, which would give the parse to the entire sentence as follows:"(S (NP It) (VP makes (S (NP me) (PP (TO to) (ADJP happy))))).)"

In summary, for insertion errors to which PTB-II do not apply, we determine their phrase structures as follows: (i) intuitively group words into a phrase, minimizing the number of new rules added (it is often helpful to examine whether an existing rule is partially applicable to the words in question); (ii) name the resulting phrase by the POS of one of the immediate children that is entitled to be a head.

2.2.3 Replacement Type Errors

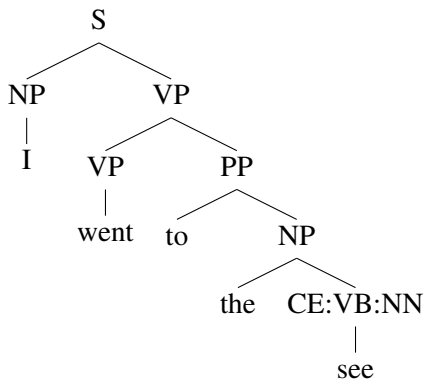
A replacement type error is an error where a word should be replaced with another word. For example, in the sentence: **I often study English conversation.*, the verb *study* should be replaced with a more appropriate verb such as *practice*.

To handle replacement type errors systematically, we introduce a concept called *POS class*, which is a grouping of POS categories defined as

Class	Members
Noun	NN, NNS, NNP, NNPS
Verb	VB, VBP, VBZ, VBD
Adjective	JJ, JJR, JJS
Adverb	RB, RBR, RBS
Participle	VBN, VBG

Table 1: POS class.

in Table 1; POS tags that are not shown in Table 1 form a POS class by itself. If the replacement in question is within the same POS class, it is annotated following Principles (P2) and (P3). Namely, the erroneous word is tagged according to its superficial form and the rest of the sentence is annotated by the original rule set, which avoids creating new rules[3]. If the replacement in question is from one POS class to another, we will need to take special care because of the ambiguity between superficial and distributional POS categories. For example, consider the sentence **I went to the see.* where the word *see* is used as a noun, which is not allowed in the standard English, and the intention of the learner is likely to be *sea* (from the surrounding context). Thus, the word *see* is ambiguous between a verb and a noun in the sentence. To avoid the ambiguity, we adopt a two layer-annotation scheme (Díaz-Negrillo et al., 2009; Nagata et al., 2011; Ragheb and Dickinson, 2012) to include both POSs. In our annotation scheme, we use a special tag (CE) for the replacement error and encode the two POSs as its attribute values as in *CE:VB:NN*. Then we can use the distributional POS tag to annotate the rest of the sentence. For example, the above example sentence would give a tree:

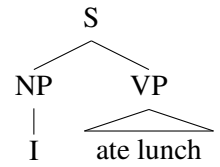


[3]This means that spelling and morphological errors are not directly coded in our annotation scheme as in *He/PRP has/VBZ a/DT books/NNS*.

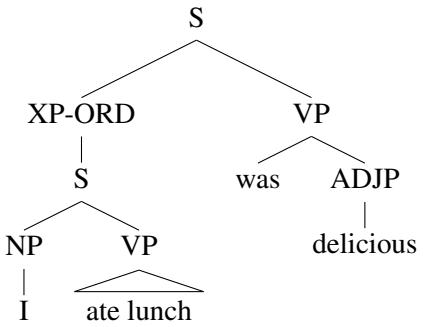
2.2.4 Errors in Word Order

Errors in word order often appear in learner English. A typical example would be the reverse of the subject-object order: **This place like my friends.* (correctly, *My friends like this place.*).

Principles (P2) and (P3) again play an important role in handling errors in word order. We first determine the POS tag of each word according to its morphological form. This is rather straightforward because errors in word order do not affect the morphological form. Then we determine the whole structure based on the resulting POS tags, following Principle (P2); if rules in PTB-II apply to the sentence in question, we parse it according to them just as in the above example sentence: "(S (NP This place) (VP like (NP my friends)).)" Even if any of the existing rules do not apply to a part of the sequence of the given POS tags, we stick to Principle (P3) as much as possible. In other words, we determine partial phrase structures according to the given POS sequence to which the existing rule set applies. Then we use the XP-ORD tag to put them together into a phrase. As an example, consider the sentence **I ate lunch was delicious.* (correctly, *The lunch I ate was delicious.*). According to the superficial forms and local contexts, the phrase *I ate lunch* would form an S:



However, the relations of the S to the rest of the constituents are not clear. Here, we use the XP-ORD tag to combine the S with the rest together:



2.2.5 Fragments

In learner corpora, sentences are sometimes incomplete. They are called fragments (e.g., missing main clause: *Because I like it.*).

Fortunately, there exists already a tag for fragments in PTB-II: FRAG. Accordingly, we use

it in our annotation scheme as well. For example, the above example would give the parse "(FRAG (SBAR Because (S (NP I (VP like (NP it))))).)" An exception is incomplete sentences which are defined as S in the bracketing guidelines for biomedical texts (Warner et al., 2012). We tag such incomplete sentences as S following the convention. For example, an adjective phrase can form an S (e.g., (S (ADVP Beautiful)!)).

2.2.6 Unknown Words and Phrases

There are cases where one cannot tell the tag of a given word. We use the UK tag for such words (e.g., *Everyone is don/UK)*.

Even if its tag is unknown, it is somehow clear in some cases that the unknown word is the head word of the phrase just as in the above example. In that case, we use the UP tag so that it satisfies the rule about the headedness of a phrase we have introduced in Subsect. 2.2.2. Based on this, the above example would give the parse "(S (NP everyone) (VP is (UP (UK don))).)"

For a phrase whose head word is unknown due to some error(s) in it, we use the XP tag instead of the UP tag. As a special case of XP, we use the XP-ORD tag to denote the information that we cannot determine the head of the phrase because of an error in word order.

3 Corpus Annotation

We selected the Konan-JIEM (KJ) learner corpus (Nagata et al., 2011) (beginning to intermediate levels) as our target data. It is manually annotated with POSs, chunks, and grammatical errors, which helps annotators to select correct tags. We also included in the target data a part of the essays in ICNALE (Ishikawa, 2011) consisting of a variety of learners (beginning to advanced levels[4]) in Asia (China, Indonesia, Japan, Korea, Taiwan, Thailand, Hong Kong, Singapore, Pakistan, Philippines). Table 2 shows the statistics on the two learner corpora.

Two professional annotators[5] participated in the annotation process. One of them first annotated the KJ data and double-checked the results. Between the first and second checks, we discussed the results with the annotator. We revised the annotation scheme based on the discussion which resulted in the present version. Then the second annotator annotated a part of the KJ data to evaluate the consistency between the two annotators. We took out 11 texts (955 tokens) as a development set. The second annotator annotated it using the revised annotation scheme where she consulted the first annotator if necessary. After this, we provided her with the differences between the results of the two annotators. Finally, the first annotator annotated the data in ICNALE while the second independently another part of the KJ data and a part of the ICNALE data (59 texts, 12,052 tokens in total), which were treated as a test set.

Table 3 shows inter-annotator agreement measured in recall, precision, F-measure, complete match rate, and chance-corrected measure (Skjærholt, 2014). We used the EVALB tool[6] with the Collins (1997)'s evaluation parameter where we regarded the annotation results of the first annotator as the gold standard set. We also used the syn-agreement tool[7] to calculate chance-corrected measure. It turns out that the agreement is very high. Even in the test set, they achieve an F-measure of 0.928 and a chance-corrected measure of 0.982. This shows that our annotation scheme enabled the annotators to consistently recognize the phrase structures in the learner corpora in which grammatical errors frequently appear. The comparison between the results of the two annotators shows the major sources of the disagreements. One of them is annotation concerning adverbial phrases. In PTB-II, an adverbial phrase between the subject NP and the main verb is allowed to be a constituent of the VP (e.g., (S (NP I) (VP (ADVP often) go))) and also of the S (e.g., (S (NP I) (ADVP often) (VP go))). Another major source is the tag FRAG (fragments); the annotators disagreed on distinguishing between FRAG and S in some cases.

The high agreement shows that the annotation scheme provides an effective way of consistently annotating learner corpora with phrase structures. However, one might argue that the annotation does not represent the characteristics of learner English well because it favors consistency (and rather simple annotation rules) over completeness.

To see if the annotation results represent the

[4]The details about the proficiency levels are available in http://language.sakura.ne.jp/icnale/about.html

[5]The annotators, whose mother tongue is Japanese, have a good command of English. They have engaged in corpus annotation including phrase structure annotation for around 20 years.

[6]http://nlp.cs.nyu.edu/evalb/

[7]https://github.com/arnsholt/syn-agreement

Corpus	# essays	# sentences	# tokens	# errors/token	# errors/sentence
KJ	233	3,260	30,517	0.15	1.4
ICNALE	134	1,930	33,913	0.08	1.4

Table 2: Statistics on annotated learner corpora.

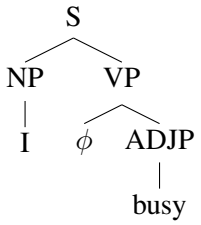
Set	R	P	F	CMR	CCM
Development	0.981	0.981	0.981	0.913	0.995
Test	0.919	0.927	0.928	0.549	0.982

Table 3: Inter-annotator agreement measured in Recall (R), Precision (P), F-measure (F), Complete Match Rate (CMR), and Chance-Corrected Measure (CCM).

characteristics of learner English, we extracted characteristic CFG rules from them. The basic idea is that we compare the CFG rules obtained from them with those from a native corpus (the Penn Treebank-II)[8]; we select as characteristic CFG rules those that often appear in the learner corpora and not in the native corpus. To formalize the extraction procedures, we denote a CFG rule and its conditional probability as $A \rightarrow B$ and $p(B|A)$, respectively. Then we define the score for $A \rightarrow B$ by $s(A \rightarrow B) = \log \frac{p_L(B|A)}{p_N(B|A)}$ where we distinguish between learner and native corpora by the subscripts L and N, respectively. We estimate $p(B|A)$ by expected likelihood estimation. Note that we remove the function tags to reduce the differences in the syntactic tags in both corpora when we calculate the score.

Table 4 shows the top 10 characteristic CFG rules sorted in descending and ascending order according to their scores, which correspond to overused and underused rules in the learner corpora, respectively. Note that Table 4 excludes rules consisting of only terminal and/or preterminal symbols to focus on the structural characteristics. Also, it excludes rules containing a Quantifier Phrase (QP; e.g., (NP (QP 100 million) dollars)), which frequently appear and is one of the characteristics in the native corpus.

In the overused column, CFG rules often contain the ϕ element. At first sight, this does not seem so surprising because ϕ never appears in the native corpus. However, the rules actually show in which syntactic environment missing heads tend to occur. For example, the CFG rule *PP* $\rightarrow$ ϕ *S* shows that prepositions tend to be missing in the prepositional phrase governing an S as in **I am good _ doing this*, which we had not realized before this investigation. More interestingly, the CFG rule *VP* $\rightarrow$ ϕ *ADJP* reveals that an adjective phrase can form a verb phrase without a verb in learner English. Looking into the annotated data shows that the copula is missing in predicative adjectives as in the tree:



This suggests the transfer of the linguistic system that the copula is not necessary or may be omitted in predicate adjectives in certain languages such as Japanese and Chinese. Similarly, the rule *VP* $\rightarrow$ ϕ *NP* shows in which environment a verb taking the object tends to be missing. Out of the 28 instances, 18 (64%) are in a subordinate clause, which implies that learners tend to omit a verb when more than one verb appear in a sentence.

The second rule *S* $\rightarrow$ *XP VP* . implies that the subject NP cannot be recognized because of a combination of grammatical errors (c.f., *S* $\rightarrow$ *NP VP* .). The corpus data show that 21% of *XP* in *S* $\rightarrow$ *XP VP* . are actually *XP-ORD* concerning an error in a relative clause just as shown in the tree in Subsect. 2.2.4. Some of the learners apparently have problems in appropriately using relative clauses in the subject position. It seems that the structure of the relative clause containing another verb before the main verb confuses them.

Most of the underused CFG rules are those that introduce rather complex structures. For exam-

[8]To confirm that the extracted characteristics are not influenced by the differences in the domains of the two corpora, we also compared the learner data with the native speaker sub-corpus in ICNALE that is in the same domain. It turned out that the extracted CFG rules, were very similar to those shown in Table 4.

Overuse	Score	Underuse	Score
PP → ϕ NP	9.0	NP → NP , NP ,	-4.6
S → XP VP .	7.2	S → NP NP	-2.7
PP → IN IN S	6.7	S → NP VP . ”	-2.6
S → XP .	6.6	ADVP → NP RBR	-2.5
VP → ϕ ADJP	6.5	S → S , NP VP .	-2.4
VP → ϕ NP	6.3	NP → NP , SBAR	-2.4
SBAR → IN NN TO S	6.1	SBAR → WHPP S	-2.3
PP → ϕ S	6.1	VP → VBD SBAR	-2.2
S → ADVP NP ADVP VP .	5.8	S → NP PRN VP .	-2.2
PP → IN TO NP	5.7	S → PP , NP VP . ”	-2.1

Table 4: Characteristic CFG rules.

ple, the eighth rule *VP → VBD SBAR* implies a structure such as *He thought that* $\cdots$. The underused CFG rules are a piece of the evidence that this population of learners of English cannot use such complex structures as fluently as native speakers do. Considering this, it will be useful feedback to provide them with the rules (transformed into interpretable forms). As in this example, phrase structure annotation should be useful not only for second language acquisition research but also for language learning assistance.

4 Parsing Performance Evaluation

We tested the following two state-of-the-art parsers on the annotated data: Stanford Statistical Natural Language Parser (ver.2.0.3) (de Marneffe et al., 2006) and Charniak-Johnson parser (Charniak and Johnson, 2005). We gave the tokenized sentences to them as their inputs. We used again the EVALB tool with the Collins (1997)'s evaluation parameter.

Table 5 shows the results. To our surprise, both parsers perform very well on the learner corpora despite the fact that it contains a number of grammatical errors and also syntactic tags that are not defined in PTB-II. Their performance is comparable to, or even better than, that on the Penn Treebank (reported in Petrov (2010)).

To achieve further improvement, we augmented the Charniak-Johnson parser with the learner data. We first retrained its parser model using the 2-21 sections of Penn Treebank Wall Street Journal (hereafter, WSJ) as training data and its 24 section as development data, following the settings shown in Charniak and Johnson (2005). We then added the learner corpora to the training data using six-fold cross validation. We split it into six parts,

Parser	R	P	F	CMR
Stanford	0.812	0.832	0.822	0.398
Charniak-Johnson	0.845	0.865	0.855	0.465

Table 5: Parsing performance on learner English.

each of which approximately consisted of 61 essays, used one sixth as test data, another one sixth as development data instead of the 24 section, and retrained the parser model using the development data and the training data consisting of the remaining four-sixths part of the learner data and the 2-21 sections of WSJ. We also conducted experiments where we copied the four sixths of the learner data n times ($1 \leq n \leq 50$) and added them to the training data to increase its weight in retraining.

Figure 1 shows the results. The simple addition of the learner data ($n = 1$) already outperforms the parser trained only on the 2-21 sections of WSJ ($n = 0$) in both recall and precision, achieving an F-measure of 0.866 and a complete match rate of 0.515. The augmented parser model particularly works well on recognizing erroneous fragments in the learner data; F-measure improved to 0.796 ($n = 1$) from 0.683 ($n = 0$) in the sentences containing fragments (i.e., FRAG) (46 out of the 111 sentences that were originally erroneously parsed made even a complete match). It was also robust against spelling errors. The performance further improves as the weight n increases (up to $F = 0.878$ when $n = 24$), which shows the effectiveness of using learner corpus data as training data.

Figure 2 shows the parsing performance of the Charniak-Johnson parser in each sub-corpus of

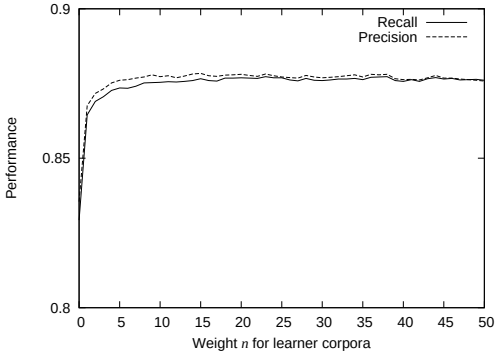


Figure 1: Relation between learner corpus size in training data and parsing performance.

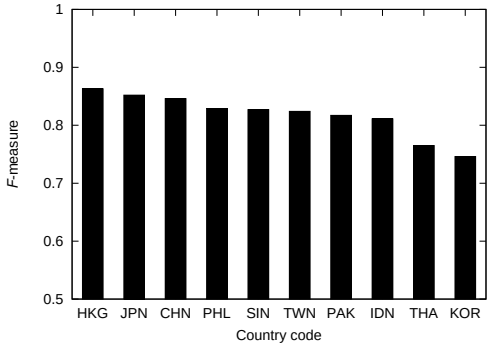


Figure 2: Parsing performance in each sub-corpus.

ICNALE (classified by country code[9]). In most of the sub-corpora, the parser achieves an F-measure of 0.800 or better. By contrast, it performs much worse on the Korean sub-corpus. The major reason for this is that it contains a number of word order errors (i.e., XP-ORD); to be precise, 27 instances compared to zero to two instances in the other sub-corpora. Similarly, FRAG is a source of parsing errors in the Thai sub-corpus. We need further investigation to determine whether the differences in parsing performance are due to the writers' mother tongue or other factors (e.g., proficiency).

We can summarize the findings as follows: (1) the state-of-the-art phrase structure parsers for native English are effective even in parsing learner English; (2) they are successfully augmented by learner corpus data; (3) the evaluation results support the previous report (Tetreault et al., 2010) that they are effective in extracting parse features for grammatical error correction (and probably for related NLP tasks such as automated essay scoring); (4) however, performance may vary depending on the writer's mother tongue and/or other factors, which we need further investigation to confirm.

5 Conclusions

This paper explored phrase structure annotation and parsing specially designed for learner English. Sect. 3 showed the usefulness of our phrase structure annotation scheme and the learner corpora annotated using it. The annotation results exhibited high consistency. They also shed light on (at least, part of) the characteristics of the learners of English. Sect. 4 further reported on the performance of the two state-of-the-art parsers on the annotated corpus, suggesting that they are accurate for providing NLP applications with phrase structures in learner English. All these findings support the effectiveness of our phrase structure annotation scheme for learner English. It would be much more difficult to conduct similar analyses and investigations without the phrase structure annotation scheme and a learner corpus annotated based on it. The annotation guidelines, the annotated data, and the parsing model for learner English created in this work are now available to the public[10].

In our future work, we will evaluate parsing performance on other learner corpora such as ICLE (Granger et al., 2009) consisting of a wide variety of learner Englishes. We will also extend phrase structure annotation, especially working on function tags.

Acknowledgments

We would like to thank Shin'ichiro Ishikawa, who created the ICNALE corpus, for providing us with the data and Arne Skærholt for the assistance to run the syn-agreement tool. We would also like to thank the three anonymous reviewers for their valuable feedback. This work was partly supported by Grant-in-Aid for Young Scientists (B) Grant Number JP26750091.

[9]Ideally, it would be better to use sub-corpora classified by their mother tongues. Unfortunately, however, only country codes are provided in ICNALE.

[10]We released the Konan-JIEM corpus with phrase structures on March 2015, which is available at `http://www.gsk.or.jp/en/catalog/gsk2015-a/`. We annotated the existing ICNALE, which was created by Dr. Ishikawa and his colleagues, with phrase structures. We released the data on Jun 2016, which is available at `http://language.sakura.ne.jp/icnale/`

References

Ann Bies, Mark Ferguson, Karen Katz, and Robert MacIntyre. 1995. Bracketing guidelines for Treebank II-style Penn treebank project.

Aoife Cahill, Binod Gyawali, and James V. Bruno. 2014. Self-training for parsing learner text. In *Proc. of 1st Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 66–73.

Aoife Cahill. 2015. Parsing learner text: to Shoehorn or not to Shoehorn. In *Proc. of 9th Linguistic Annotation Workshop*, pages 144–147.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine N-best parsing and MaxEnt discriminative reranking. In *Proc. of 43rd Annual Meeting on Association for Computational Linguistics*, pages 173–180.

Michael Collins. 1997. Three generative, lexicalised models for statistical parsing. In *Proc. of 35th Annual Meeting of the Association for Computational Linguistics*, pages 16–23.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proc. of 13th European Workshop on Natural Language Generation*, pages 242–249.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proc. of 5th International Conference on Language Resources and Evaluation*, pages 449–445.

Ana Díaz-Negrillo, Detmar Meurers, Salvador Valera, and Holger Wunsch. 2009. Towards interlanguage POS annotation for effective learner corpora in SLA and FLT. *Language Forum*, 36(1–2):139–154.

Markus Dickinson and Marwa Ragheb. 2009. Dependency annotation for learner corpora. In *Proc. of 8th Workshop on Treebanks and Linguistic Theories*, pages 59–70.

Jennifer Foster and Øistein E. Andersen. 2009. GenERRate: Generating errors for use in grammatical error detection. In *Proc. of 4th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 82–90.

Jennifer Foster. 2004. Parsing ungrammatical input: An evaluation procedure. In *Proc. of 4th International Conference on Language Resources and Evaluation*, pages 2039–2042.

Jennifer Foster. 2007a. Treebanks gone bad: generating a treebank of ungrammatical English. In *2007 Workshop on Analytics for Noisy Unstructured Data*, pages 39–46, Jan.

Jennifer Foster. 2007b. Treebanks gone bad: Parser evaluation and retraining using a treebank of ungrammatical sentences. *International Journal on Document Analysis and Recognition*, 10(3):129–145, Dec.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English v2*. Presses universitaires de Louvain, Louvain.

Shinichiro Ishikawa, 2011. *A new horizon in learner corpus studies: The aim of the ICNALE project*, pages 3–11. University of Strathclyde Publishing, Glasgow.

Emi Izumi, Toyomi Saiga, Thepchai Supnithi, Kiyotaka Uchimoto, and Hitoshi Isahara. 2004. The NICT JLE Corpus: Exploiting the language learners' speech database for research and education. *International Journal of The Computer, the Internet and Management*, 12(2):119–125.

Stephan Kepser, Ilona Steiner, and Wolfgang Sternefeld. 2004. Annotating and querying a treebank of suboptimal structures. In *Proc. of 3rd Workshop on Treebanks and Linguistic Theories*, pages 63–74.

Ryo Nagata, Edward Whittaker, and Vera Sheinman. 2011. Creating a manually error-tagged and shallow-parsed learner corpus. In *Proc. of 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1210–1219.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proc. 17th Conference on Computational Natural Language Learning: Shared Task*, pages 1–12.

Niels Ott and Ramon Ziai. 2010. Evaluating dependency parsing performance on German learner language. In *Proc. of 9th International Workshop on Treebanks and Linguistic Theories*, pages 175–186.

Slav Petrov. 2010. Products of random latent variable grammars. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27.

Marwa Ragheb and Markus Dickinson. 2012. Defining syntax for learner language annotation. In *Proc. of 24th International Conference on Computational Linguistics*, pages 965–974.

Marwa Ragheb and Markus Dickinson. 2013. Inter-annotator agreement for dependency annotation of learner language. In *Proc. of 8th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 169–179.

Arne Skjærholt. 2014. A chance-corrected measure of inter-annotator agreement for syntax. In *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics*, pages 934–944.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proc. of 48nd Annual Meeting of the Association for Computational Linguistics Short Papers*, pages 353–358.

Colin Warner, Arrick Lanfranchi, Tim O'Gorman, Amanda Howard, Kevin Gould, and Michael Regan. 2012. Bracketing biomedical text: An addendum to Penn Treebank II guidelines.

A Trainable Spaced Repetition Model for Language Learning

Burr Settles[*]
Duolingo
Pittsburgh, PA USA
burr@duolingo.com

Brendan Meeder[†]
Uber Advanced Technologies Center
Pittsburgh, PA USA
bmeeder@cs.cmu.edu

Abstract

We present *half-life regression (HLR)*, a novel model for spaced repetition practice with applications to second language acquisition. HLR combines psycholinguistic theory with modern machine learning techniques, indirectly estimating the "half-life" of a word or concept in a student's long-term memory. We use data from Duolingo — a popular online language learning application — to fit HLR models, reducing error by 45%+ compared to several baselines at predicting student recall rates. HLR model weights also shed light on which linguistic concepts are systematically challenging for second language learners. Finally, HLR was able to improve Duolingo daily student engagement by 12% in an operational user study.

1 Introduction

The *spacing effect* is the observation that people tend to remember things more effectively if they use *spaced repetition practice* (short study periods spread out over time) as opposed to *massed practice* (i.e., "cramming"). The phenomenon was first documented by Ebbinghaus (1885), using himself as a subject in several experiments to memorize verbal utterances. In one study, after a day of cramming he could accurately recite 12-syllable sequences (of gibberish, apparently). However, he could achieve comparable results with half as many practices spread out over three days.

The *lag effect* (Melton, 1970) is the related observation that people learn even better if the spacing between practices gradually increases. For example, a learning schedule might begin with review sessions a few seconds apart, then minutes, then hours, days, months, and so on, with each successive review stretching out over a longer and longer time interval.

The effects of spacing and lag are well-established in second language acquisition research (Atkinson, 1972; Bloom and Shuell, 1981; Cepeda et al., 2006; Pavlik Jr and Anderson, 2008), and benefits have also been shown for gymnastics, baseball pitching, video games, and many other skills. See Ruth (1928), Dempster (1989), and Donovan and Radosevich (1999) for thorough meta-analyses spanning several decades.

Most practical algorithms for spaced repetition are simple functions with a few hand-picked parameters. This is reasonable, since they were largely developed during the 1960s–80s, when people would have had to manage practice schedules without the aid of computers. However, the recent popularity of large-scale online learning software makes it possible to collect vast amounts of parallel student data, which can be used to empirically train richer statistical models.

In this work, we propose *half-life regression (HLR)* as a trainable spaced repetition algorithm, marrying psycholinguistically-inspired models of memory with modern machine learning techniques. We apply this model to real student learning data from Duolingo, a popular language learning app, and use it to improve its large-scale, operational, personalized learning system.

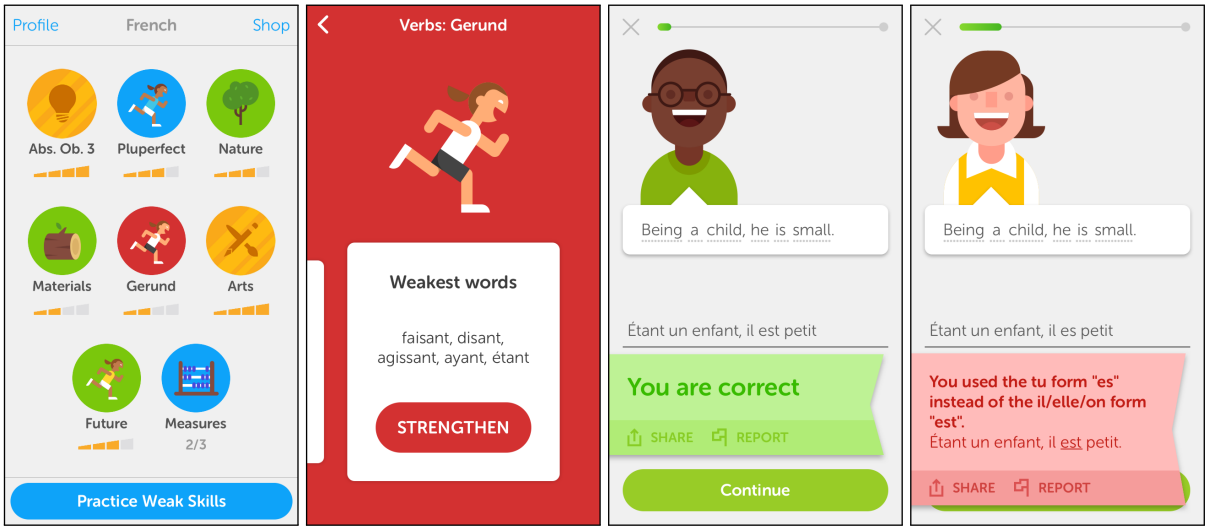
2 Duolingo

Duolingo is a free, award-winning, online language learning platform. Since launching in 2012, more than 150 million students from all over the world have enrolled in a Duolingo course, either via the website[1] or mobile apps for Android, iOS,

[*] Corresponding author.
[†] Research conducted at Duolingo.

[1] https://www.duolingo.com

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1848–1858,
Berlin, Germany, August 7-12, 2016.



(a) skill tree screen (b) skill screen (c) correct response (d) incorrect response

Figure 1: Duolingo screenshots for an English-speaking student learning French (iPhone app, 2016). (a) A course skill tree: golden skills have four bars and are "at full strength," while other skills have fewer bars and are due for practice. (b) A skill screen detail (for the *Gerund* skill), showing which words are predicted to need practice. (c,d) Grading and explanations for a translation exercise.

étant	un	enfant	il	est	petit
être.V.GER	*un*.DET.INDF.M.SG	*enfant*.N.SG	*il*.PN.M.P3.SG	*être*.V.PRES.P3.SG	*petit*.ADJ.M.SG

Figure 2: The French sentence from Figure 1(c,d) and its lexeme tags. Tags encode the root lexeme, part of speech, and morphological components (tense, gender, person, etc.) for each word in the exercise.

and Windows devices. For comparison, that is more than the total number of students in U.S. elementary and secondary schools combined. At least 80 language courses are currently available or under development[2] for the Duolingo platform. The most popular courses are for learning English, Spanish, French, and German, although there are also courses for minority languages (Irish Gaelic), and even constructed languages (Esperanto).

More than half of Duolingo students live in developing countries, where Internet access has more than tripled in the past three years (ITU and UNESCO, 2015). The majority of these students are using Duolingo to learn English, which can significantly improve their job prospects and quality of life (Pinon and Haydon, 2010).

2.1 System Overview

Duolingo uses a playfully illustrated, gamified design that combines point-reward incentives with implicit instruction (DeKeyser, 2008), mastery learning (Block et al., 1971), explanations (Fahy, 2004), and other best practices. Early research suggests that 34 hours of Duolingo is equivalent to a full semester of university-level Spanish instruction (Vesselinov and Grego, 2012).

Figure 1(a) shows an example **skill tree** for English speakers learning French. This specifies the game-like curriculum: each icon represents a **skill**, which in turn teaches a set of thematically or grammatically related words or concepts. Students tap an icon to access lessons of new material, or to practice previously-learned material. Figure 1(b) shows a screen for the French skill *Gerund*, which teaches common gerund verb forms such as *faisant* (doing) and *étant* (being). This skill, as well as several others, have already been completed by the student. However, the *Measures* skill in the bottom right of Figure 1(a) has one lesson remaining. After completing each row of skills, students "unlock" the next row of more advanced skills. This is a gamelike implementation of *mastery learning*, whereby students must reach a certain level of prerequisite knowledge before moving on to new material.

[2]https://incubator.duolingo.com

Each language course also contains a **corpus** (large database of available exercises) and a **lexeme tagger** (statistical NLP pipeline for automatically tagging and indexing the corpus; see the Appendix for details and a lexeme tag reference). Figure 1(c,d) shows an example translation exercise that might appear in the *Gerund* skill, and Figure 2 shows the lexeme tagger output for this sentence. Since this exercise is indexed with a gerund lexeme tag (*être*.V.GER in this case), it is available for lessons or practices in this skill.

The lexeme tagger also helps to provide corrective feedback. Educational researchers maintain that incorrect answers should be accompanied by *explanations*, not simply a "wrong" mark (Fahy, 2004). In Figure 1(d), the student incorrectly used the 2nd-person verb form *es* (*être*.V.PRES.P2.SG) instead of the 3rd-person *est* (*être*.V.PRES.P3.SG). If Duolingo is able to parse the student response and detect a known grammatical mistake such as this, it provides an explanation[3] in plain language. Each lesson continues until the student masters all of the **target words** being taught in the session, as estimated by a mixture model of short-term learning curves (Streeter, 2015).

2.2 Spaced Repetition and Practice

Once a lesson is completed, all the target words being taught in the lesson are added to the **student model**. This model captures what the student has learned, and estimates how well she can recall this knowledge at any given time. Spaced repetition is a key component of the student model: over time, the strength of a skill will decay in the student's long-term memory, and this model helps the student manage her practice schedule.

Duolingo uses **strength meters** to visualize the student model, as seen beneath each of the completed skill icons in Figure 1(a). These meters represent the average probability that the student can, at any moment, correctly recall a random target word from the lessons in this skill (more on this probability estimate in §3.3). At four bars, the skill is "golden" and considered fresh in the student's memory. At fewer bars, the skill has grown stale and may need practice. A student can tap the skill icon to access practice sessions and target her weakest words. For example, Figure 1(b) shows some weak words from the *Gerund* skill. Practice sessions are identical to lessons, except that the exercises are taken from those indexed with words (lexeme tags) due for practice according to student model. As time passes, strength meters continuously update and decay until the student practices.

[3]If Duolingo cannot parse the precise nature of the mistake — e.g., because of a gross typographical error — it provides a "diff" of the student's response with the closest acceptable answer in the corpus (using Levenshtein distance).

3 Spaced Repetition Models

In this section, we describe several spaced repetition algorithms that might be incorporated into our student model. We begin with two common, established methods in language learning technology, and then present our half-life regression model which is a generalization of them.

3.1 The Pimsleur Method

Pimsleur (1967) was perhaps the first to make mainstream practical use of the spacing and lag effects, with his audio-based language learning program (now a franchise by Simon & Schuster). He referred to his method as *graduated-interval recall*, whereby new vocabulary is introduced and then tested at exponentially increasing intervals, interspersed with the introduction or review of other vocabulary. However, this approach is limited since the schedule is pre-recorded and cannot adapt to the learner's actual ability. Consider an English-speaking French student who easily learns a cognate like *pantalon* (pants), but struggles to remember *manteau* (coat). With the Pimsleur method, she is forced to practice both words at the same fixed, increasing schedule.

3.2 The Leitner System

Leitner (1972) proposed a different spaced repetition algorithm intended for use with flashcards. It is more adaptive than Pimsleur's, since the spacing intervals can increase or decrease depending on student performance. Figure 3 illustrates a popular variant of this method.

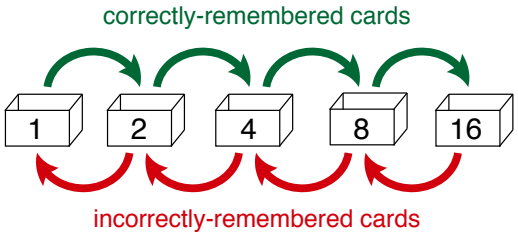


Figure 3: The Leitner System for flashcards.

The main idea is to have a few boxes that correspond to different practice intervals: 1-day, 2-day,

4-day, and so on. All cards start out in the 1-day box, and if the student can remember an item after one day, it gets "promoted" to the 2-day box. Two days later, if she remembers it again, it gets promoted to the 4-day box, etc. Conversely, if she is incorrect, the card gets "demoted" to a shorter interval box. Using this approach, the hypothetical French student from §3.1 would quickly promote *pantalon* to a less frequent practice schedule, but continue reviewing *manteau* often until she can regularly remember it.

Several electronic flashcard programs use the Leitner system to schedule practice, by organizing items into "virtual" boxes. In fact, when it first launched, Duolingo used a variant similar to Figure 3 to manage skill meter decay and practice. The present research was motivated by the need for a more accurate model, in response to student complaints that the Leitner-based skill meters did not adequately reflect what they had learned.

3.3 Half-Life Regression: A New Approach

We now describe half-life regression (HLR), starting from psychological theory and combining it with modern machine learning techniques.

Central to the theory of memory is the *Ebbinghaus model*, also known as the *forgetting curve* (Ebbinghaus, 1885). This posits that memory decays exponentially over time:

$$p = 2^{-\Delta/h} \ . \tag{1}$$

In this equation, p denotes the probability of correctly recalling an item (e.g., a word), which is a function of Δ, the *lag time* since the item was last practiced, and h, the *half-life* or measure of strength in the learner's long-term memory.

Figure 4(a) shows a forgetting curve (1) with half-life $h = 1$. Consider the following cases:

1. $\Delta = 0$. The word was just recently practiced, so $p = 2^0 = 1.0$, conforming to the idea that it is fresh in memory and should be recalled correctly regardless of half-life.

2. $\Delta = h$. The lag time is equal to the half-life, so $p = 2^{-1} = 0.5$, and the student is on the verge of being unable to remember.

3. $\Delta \gg h$. The word has not been practiced for a long time relative to its half-life, so it has probably been forgotten, e.g., $p \approx 0$.

Let $\mathbf{x}$ denote a feature vector that summarizes a student's previous exposure to a particular word, and let the parameter vector Θ contain weights that correspond to each feature variable in $\mathbf{x}$. Under the assumption that half-life should increase exponentially with each repeated exposure (a common practice in spacing and lag effect research), we let $\hat{h}_\Theta$ denote the estimated half-life, given by:

$$\hat{h}_\Theta = 2^{\Theta \cdot \mathbf{x}} \ . \tag{2}$$

In fact, the Pimsleur and Leitner algorithms can be interpreted as special cases of (2) using a few fixed, hand-picked weights. See the Appendix for the derivation of Θ for these two methods.

For our purposes, however, we want to fit Θ empirically to learning trace data, and accommodate an arbitrarily large set of interesting features (we discuss these features more in §3.4). Suppose we have a data set $\mathcal{D} = \{\langle p, \Delta, \mathbf{x} \rangle_i\}_{i=1}^{D}$ made up of student-word practice sessions. Each data instance consists of the observed recall rate p[4], lag time Δ since the word was last seen, and a feature vector $\mathbf{x}$ designed to help personalize the learning experience. Our goal is to find the best model weights Θ^* to minimize some loss function ℓ:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{D} \ell(\langle p, \Delta, \mathbf{x} \rangle_i; \Theta) \ . \tag{3}$$

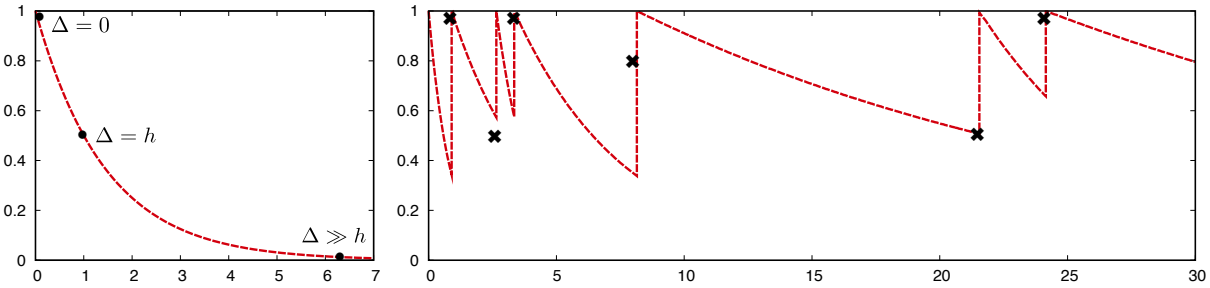
To illustrate, Figure 4(b) shows a student-word learning trace over the course of a month. Each ✖ indicates a data instance: the vertical position is the observed recall rate p for each practice session, and the horizontal distance between points is the lag time Δ between sessions. Combining (1) and (2), the model prediction $\hat{p}_\Theta = 2^{-\Delta/\hat{h}_\Theta}$ is plotted as a dashed line over time (which resets to 1.0 after each exposure, since $\Delta = 0$). The training loss function (3) aims to fit the predicted forgetting curves to observed data points for millions of student-word learning traces like this one.

We chose the L_2-regularized squared loss function, which in its basic form is given by:

$$\ell(✖; \Theta) = (p - \hat{p}_\Theta)^2 + \lambda \|\Theta\|_2^2 \ ,$$

where $✖ = \langle p, \Delta, \mathbf{x} \rangle$ is shorthand for the training data instance, and λ is a parameter to control the regularization term and help prevent overfitting.

[4] In our setting, each data instance represents a full lession or practice session, which may include multiple exercises reviewing the same word. Thus p represents the *proportion* of times a word was recalled correctly in a particular session.



(a) Ebbinghaus model ($h = 1$) (b) 30-day student-word learning trace and predicted forgetting curve

Figure 4: Forgetting curves. (a) Predicted recall rate as a function of lag time Δ and half-life $h = 1$. (b) Example student-word learning trace over 30 days: ✖ marks the observed recall rate p for each practice session, and half-life regression aims to fit model predictions $\hat{p}_\Theta$ (dashed lines) to these points.

In practice, we found it useful to optimize for the half-life h in addition to the observed recall rate p. Since we do not know the "true" half-life of a given word in the student's memory — this is a hypothetical construct — we approximate it algebraically from (1) using p and Δ. We solve for $h = \frac{-\Delta}{\log_2(p)}$ and use the final loss function:

$$\ell(\text{✖}; \Theta) = (p - \hat{p}_\Theta)^2 + \alpha(h - \hat{h}_\Theta)^2 + \lambda\|\Theta\|_2^2 \,,$$

where α is a parameter to control the relative importance of the half-life term in the overall training objective function. Since ℓ is smooth with respect to Θ, we can fit the weights to student-word learning traces using gradient descent. See the Appendix for more details on our training and optimization procedures.

3.4 Feature Sets

In this work, we focused on features that were easily instrumented and available in the production Duolingo system, without adding latency to the student's user experience. These features fall into two broad categories:

- *Interaction features*: a set of counters summarizing each student's practice history with each word (lexeme tag). These include the total number of times a student has seen the word x_n, the number of times it was correctly recalled $x_\oplus$, and the number of times incorrect $x_\ominus$. These are intended to help the model make more personalized predictions.

- *Lexeme tag features*: a large, sparse set of indicator variables, one for each lexeme tag in the system (about 20k in total). These are intended to capture the inherent difficulty of each particular word (lexeme tag).

recall rate		lag (days)	feature vector **x**			
p	($\oplus$/n)	Δ	x_n	$x_\oplus$	$x_\ominus$	$x_{\textit{être}.\text{V.GER}}$
1.0	(3/3)	0.6	3	2	1	1
0.5	(2/4)	1.7	6	5	1	1
1.0	(3/3)	0.7	10	7	3	1
0.8	(4/5)	4.7	13	10	3	1
0.5	(1/2)	13.5	18	14	4	1
1.0	(3/3)	2.6	20	15	5	1

Table 1: Example training instances. Each row corresponds to a data point in Figure 4(b) above, which is for a student learning the French word *étant* (lexeme tag *être*.V.GER).

To be more concrete, imagine that the trace in Figure 4(b) is for a student learning the French word *étant* (lexeme tag *être*.V.GER). Table 1 shows what $\langle p, \Delta, \mathbf{x} \rangle$ would look like for each session in the student's history with that word. The interaction features increase monotonically[5] over time, and $x_{\textit{être}.\text{V.GER}}$ is the only lexeme feature to "fire" for these instances (it has value 1, all other lexeme features have value 0). The model also includes a bias weight (intercept) not shown here.

4 Experiments

In this section, we compare variants of HLR with other spaced repetition algorithms in the context of Duolingo. First, we evaluate methods against historical log data, and analyze trained model weights for insight. We then describe two controlled user experiments where we deployed HLR as part of the student model in the production system.

[5]Note that in practice, we found that using the square root of interaction feature counts (e.g., $\sqrt{x_\oplus}$) yielded better results than the raw counts shown here.

Model	MAE↓	AUC↑	COR_h↑
HLR	**0.128***	0.538*	**0.201***
HLR -lex	**0.128***	0.537*	0.160*
HLR -h	0.350	0.528*	*-0.143**
HLR -lex-h	0.350	0.528*	*-0.142**
Leitner	0.235	**0.542***	*-0.098**
Pimsleur	0.445	0.510*	*-0.132**
LR	0.211	0.513*	n/a
LR -lex	0.212	0.514*	n/a
Constant $\bar{p} = 0.859$	0.175	n/a	n/a

Table 2: Evaluation results using historical log data (see text). Arrows indicate whether lower (↓) or higher (↑) scores are better. The best method for each metric is shown in bold, and statistically significant effects ($p < 0.001$) are marked with *.

4.1 Historical Log Data Evaluation

We collected two weeks of Duolingo log data, containing 12.9 million student-word lesson and practice session traces similar to Table 1 (for all students in all courses). We then compared three categories of spaced repetition algorithms:

- *Half-life regression (HLR)*, our model from §3.3. For ablation purposes, we consider four variants: with and without lexeme features (-lex), as well as with and without the half-life term in the loss function (-h).
- *Leitner and Pimsleur*, two established baselines that are special cases of HLR, using fixed weights. See the Appendix for a derivation of the model weights we used.
- *Logistic regression (LR)*, a standard machine learning[6] baseline. We evaluate two variants: with and without lexeme features (-lex).

We used the first 1 million instances of the data to tune the parameters for our training algorithm. After trying a handful of values, we settled on $\lambda = 0.1$, $\alpha = 0.01$, and learning rate $\eta = 0.001$. We used these same training parameters for HLR and LR experiments (the Leitner and Pimsleur models are fixed and do not require training).

[6]For LR models, we include the lag time x_Δ as an additional feature, since — unlike HLR — it isn't explicitly accounted for in the model. We experimented with polynomial and exponential transformations of this feature, as well, but found the raw lag time to work best.

Table 2 shows the evaluation results on the full data set of 12.9 million instances, using the first 90% for training and remaining 10% for testing. We consider several different evaluation measures for a comprehensive comparison:

- *Mean absolute error (MAE)* measures how closely predictions resemble their observed outcomes: $\frac{1}{D}\sum_{i=1}^{D}|p - \hat{p}_\Theta|_i$. Since the strength meters in Duolingo's interface are based on model predictions, we use MAE as a measure of prediction quality.
- *Area under the ROC curve (AUC)* — or the Wilcoxon rank-sum test — is a measure of ranking quality. Here, it represents the probability that a model ranks a random correctly-recalled word as more likely than a random incorrectly-recalled word. Since our model is used to prioritize words for practice, we use AUC to help evaluate these rankings.
- *Half-life correlation (COR_h)* is the Spearman rank correlation between $\hat{h}_\Theta$ and the algebraic estimate h described in §3.3. We use this as another measure of ranking quality.

For all three metrics, HLR with lexeme tag features is the best (or second best) approach, followed closely by HLR -lex (no lexeme tags). In fact, these are the only two approaches with MAE lower than a baseline constant prediction of the *average* recall rate in the training data (Table 2, bottom row). These HLR variants are also the only methods with positive COR_h, although this seems reasonable since they are the only two to directly optimize for it. While lexeme tag features made limited impact, the h term in the HLR loss function is clearly important: MAE more than doubles without it, and the -h variants are generally worse than the other baselines on at least one metric.

As stated in §3.2, Leitner was the spaced repetition algorithm used in Duolingo's production student model at the time of this study. The Leitner method did yield the highest AUC[7] values among the algorithms we tried. However, the top two HLR variants are not far behind, and they also reduce MAE compared to Leitner by least 45%.

[7]AUC of 0.5 implies random guessing (Fawcett, 2006), so the AUC values here may seem low. This is due in part to an inherently noisy prediction task, but also to a range restriction: $\bar{p} = 0.859$, so most words are recalled correctly and predictions tend to be high. Note that all reported AUC values are statistically significantly better than chance using a Wilcoxon rank sum test with continuity correction.

Lg.	Word	Lexeme Tag	θ_k
EN	camera	*camera*.N.SG	0.77
EN	ends	*end*.V.PRES.P3.SG	0.38
EN	circle	*circle*.N.SG	0.08
EN	rose	*rise*.V.PST	*-0.09*
EN	performed	*perform*.V.PP	*-0.48*
EN	writing	*write*.V.PRESP	*-0.81*
ES	liberal	*liberal*.ADJ.SG	0.83
ES	como	*comer*.V.PRES.P1.SG	0.40
ES	encuentra	*encontrar*.V.PRES.P3.SG	0.10
ES	está	*estar*.V.PRES.P3.SG	*-0.05*
ES	pensando	*pensar*.V.GER	*-0.33*
ES	quedado	*quedar*.V.PP.M.SG	*-0.73*
FR	visite	*visiter*.V.PRES.P3.SG	0.94
FR	suis	*être*.V.PRES.P1.SG	0.47
FR	trou	*trou*.N.M.SG	0.05
FR	dessous	*dessous*.ADV	*-0.06*
FR	ceci	*ceci*.PN.NT	*-0.45*
FR	fallait	*falloir*.V.IMPERF.P3.SG	*-0.91*
DE	Baby	*Baby*.N.NT.SG.ACC	0.87
DE	sprechen	*sprechen*.V.INF	0.56
DE	sehr	*sehr*.ADV	0.13
DE	den	*der*.DET.DEF.M.SG.ACC	*-0.07*
DE	Ihnen	*Sie*.PN.P3.PL.DAT.FORM	*-0.55*
DE	war	*sein*.V.IMPERF.P1.SG	*-1.10*

Table 3: Lexeme tag weights for English (EN), Spanish (ES), French (FR), and German (DE).

4.2 Model Weight Analysis

In addition to better predictions, HLR can capture the inherent difficulty of concepts that are encoded in the feature set. The "easier" concepts take on positive weights (less frequent practice resulting from longer half-lifes), while the "harder" concepts take on negative weights (more frequent practice resulting from shorter half-lifes).

Table 3 shows HLR model weights for several English, Spanish, French, and German lexeme tags. Positive weights are associated with cognates and words that are common, short, or morphologically simple to inflect; it is reasonable that these would be easier to recall correctly. Negative weights are associated with irregular forms, rare words, and grammatical constructs like past or present participles and imperfective aspect. These model weights can provide insight into the aspects of language that are more or less challenging for students of a second language.

	Daily Retention Activity		
Experiment	Any	Lesson	Practice
I. HLR (v. Leitner)	+0.3	+0.3	*-7.3**
II. HLR -lex (v. HLR)	+12.0*	+1.7*	+9.5*

Table 4: Change (%) in daily student retention for controlled user experiments. Statistically significant effects ($p < 0.001$) are marked with *.

4.3 User Experiment I

The evaluation in §4.1 suggests that HLR is a better approach than the Leitner algorithm originally used by Duolingo (cutting MAE nearly in half). To see what effect, if any, these gains have on actual student behavior, we ran controlled user experiments in the Duolingo production system.

We randomly assigned all students to one of two groups: HLR (experiment) or Leitner (control). The underlying spaced repetition algorithm determined strength meter values in the skill tree (e.g., Figure 1(a)) as well as the ranking of target words for practice sessions (e.g., Figure 1(b)), but otherwise the two conditions were identical. The experiment lasted six weeks and involved just under 1 million students.

For evaluation, we examined changes in *daily retention*: what percentage of students who engage in an activity return to do it again the following day? We used three retention metrics: any activity (including contributions to crowdsourced translations, online forum discussions, etc.), new lessons, and practice sessions.

Results are shown in the first row of Table 4. The HLR group showed a slight increase in overall activity and new lessons, but a significant decrease in practice. Prior to the experiment, many students claimed that they would practice instead of learning new material "just to keep the tree gold," but that practice sessions did not review what they thought they needed most. This drop in practice — plus positive anecdotal feedback about stength meter quality from the HLR group — led us to believe that HLR was actually better for student engagement, so we deployed it for all students.

4.4 User Experiment II

Several months later, active students pointed out that particular words or skills would decay rapidly, regardless of how often they practiced. Upon closer investigation, these complaints could be

traced to lexeme tag features with highly negative weights in the HLR model (e.g., Table 3). This implied that some feature-based overfitting had occurred, despite the L_2 regularization term in the training procedure. Duolingo was also preparing to launch several new language courses at the time, and no training data yet existed to fit lexeme tag feature weights for these new languages.

Since the top two HLR variants were virtually tied in our §4.1 experiments, we hypothesized that using interaction features alone might alleviate both student frustration and the "cold-start" problem of training a model for new languages. In a follow-up experiment, we randomly assigned all students to one of two groups: HLR -lex (experiment) and HLR (control). The experiment lasted two weeks and involved 3.3 million students.

Results are shown in the second row of Table 4. All three retention metrics were significantly higher for the HLR -lex group. The most substantial increase was for any activity, although recurring lessons and practice sessions also improved (possibly as a byproduct of the overall activity increase). Anecdotally, vocal students from the HLR -lex group who previously complained about rapid decay under the HLR model were also positive about the change.

We deployed HLR -lex for all students, and believe that its improvements are at least partially responsible for the consistent 5% month-on-month growth in active Duolingo users since the model was launched.

5 Other Related Work

Just as we drew upon the theories of Ebbinghaus to derive HLR as an empirical spaced repetition model, there has been other recent work drawing on other (but related) theories of memory.

ACT-R (Anderson et al., 2004) is a cognitive architecture whose declarative memory module[8] takes the form of a power function, in contrast to the exponential form of the Ebbinghaus model and HLR. Pavlik and Anderson (2008) used ACT-R predictions to optimize a practice schedule for second-language vocabulary, although their setting was quite different from ours. They assumed fixed intervals between practice exercises within the same laboratory session, and found that they could improve short-term learning within a session. In contrast, we were concerned with making accurate recall predictions between multiple sessions "in the wild" on longer time scales. Evidence also suggests that manipulation between sessions can have greater impact on long-term learning (Cepeda et al., 2006).

Motivated by long-term learning goals, the multiscale context model (MCM) has also been proposed (Mozer et al., 2009). MCM combines two modern theories of the spacing effect (Staddon et al., 2002; Raaijmakers, 2003), assuming that each time an item is practiced it creates an additional item-specific forgetting curve that decays at a different rate. Each of these forgetting curves is exponential in form (similar to HLR), but are combined via weighted average, which approximates a power law (similar to ACT-R). The authors were able to fit models to controlled laboratory data for second-language vocabulary and a few other memory tasks, on times scales up to several months. We were unaware of MCM at the time of our work, and it is unclear if the additional computational overhead would scale to Duolingo's production system. Nevertheless, comparing to and integrating with these ideas is a promising direction for future work.

There has also been work on more heuristic spaced repetition models, such as SuperMemo (Woźniak, 1990). Variants of this algorithm are popular alternatives to Leitner in some flashcard software, leveraging additional parameters with complex interactions to determine spacing intervals for practice. To our knowledge, these additional parameters are hand-picked as well, but one can easily imagine fitting them empirically to real student log data, as we do with HLR.

6 Conclusion

We have introduced *half-life regression (HLR)*, a novel spaced repetition algorithm with applications to second language acquisition. HLR combines a psycholinguistic model of human memory with modern machine learning techniques, and generalizes two popular algorithms used in language learning technology: Leitner and Pimsleur. We can do this by incorporating arbitrarily rich features and fitting their weights to data. This approach is significantly more accurate at predicting student recall rates than either of the previous methods, and is also better than a conventional machine learning approach like logistic regression.

[8] Declarative (specifically semantic) memory is widely regarded to govern language vocabulary (Ullman, 2005).

One result we found surprising was that lexeme tag features failed to improve predictions much, and in fact seemed to frustrate the student learning experience due to over-fitting. Instead of the sparse indicator variables used here, it may be better to decompose lexeme tags into denser and more generic features of tag *components*[9] (e.g., part of speech, tense, gender, case), and also use corpus frequency, word length, etc. This representation might be able to capture useful and interesting regularities without negative side-effects.

Finally, while we conducted a cursory analysis of model weights in §4.2, an interesting next step would be to study such weights for even deeper insight. (Note that using lexeme tag component features, as suggested above, should make this anaysis more robust since features would be less sparse.) For example, one could see whether the ranking of vocabulary and/or grammar components by feature weight is correlated with external standards such as the CEFR (Council of Europe, 2001). This and other uses of HLR hold the potential to transform data-driven curriculum design.

Data and Code

To faciliatate research in this area, we have publicly released our data set and code from §4.1: https://github.com/duolingo/halflife-regression.

Acknowledgments

Thanks to our collaborators at Duolingo, particularly Karin Tsai, Itai Hass, and André Horie for help gathering data from various parts of the system. We also thank the anonymous reviewers for suggestions that improved the final manuscript.

References

J.R. Anderson, D. Bothell, M.D. Byrne, S. Douglass, C. Libiere, and Y. Qin. 2004. An intergrated theory of mind. *Psychological Review*, 111:1036–1060.

R.C. Atkinson. 1972. Optimizing the learning of a second-language vocabulary. *Journal of Experimental Psychology*, 96(1):124–129.

J.H. Block, P.W. Airasian, B.S. Bloom, and J.B. Carroll. 1971. *Mastery Learning: Theory and Practice*. Holt, Rinehart, and Winston, New York.

[9]Engineering-wise, each lexeme tag (e.g., *être*.V.GER) is represented by an ID in the system. We used indicator variables in this work since the IDs are readily available; the overhead of retreiving all lexeme components would be inefficient in the production system. Of course, we could optimize for this if there were evidence of a significant improvement.

K.C. Bloom and T.J. Shuell. 1981. Effects of massed and distributed practice on the learning and retention of second language vocabulary. *Journal of Educational Psychology*, 74:245–248.

N.J. Cepeda, H. Pashler, E. Vul, J.T. Wixted, and D. Rohrer. 2006. Distributed practice in verbal recall tasks: A review and quantitative synthesis. *Psychological Bulletin*, 132(3):354.

Council of Europe. 2001. *Common European Framework of Reference for Languages: Learning, Teaching, Assessment*. Cambridge University Press.

R. DeKeyser. 2008. Implicit and explicit learning. In *The Handbook of Second Language Acquisition*, chapter 11, pages 313–348. John Wiley & Sons.

F.N. Dempster. 1989. Spacing effects and their implications for theory and practice. *Educational Psychology Review*, 1(4):309–330.

J.J. Donovan and D.J. Radosevich. 1999. A meta-analytic review of the distribution of practice effect: Now you see it, now you don't. *Journal of Applied Psychology*, 84(5):795–805.

J. Duchi, E. Hazan, and Y. Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

H. Ebbinghaus. 1885. *Memory: A Contribution to Experimental Psychology*. Teachers College, Columbia University, New York, NY, USA.

P.J. Fahy. 2004. Media characteristics and online learning technology. In T. Anderson and F. Elloumi, editors, *Theory and Practice of Online Learning*, pages 137–171. Athabasca University.

T. Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters*, 27:861–874.

M.L. Forcada, M. Ginestí-Rosell, J. Nordfalk, J. O'Regan, S. Ortiz-Rojas, J.A. Pérez-Ortiz, F. Sánchez-Martínez, G. Ramírez-Sánchez, and F.M. Tyers. 2011. Apertium: A free/open-source platform for rule-based machine translation. *Machine Translation*, 25(2):127–144. http://wiki.apertium.org/wiki/Main_Page.

ITU and UNESCO. 2015. The state of broadband 2015. Technical report, September.

S. Leitner. 1972. *So lernt man lernen. Angewandte Lernpsychologie – ein Weg zum Erfolg*. Verlag Herder, Freiburg im Breisgau, Germany.

A.W. Melton. 1970. The situation with respect to the spacing of repetitions and memory. *Journal of Verbal Learning and Verbal Behavior*, 9:596–606.

M.C. Mozer, H. Pashler, N. Cepeda, R.V. Lindsey, and E. Vul. 2009. Predicting the optimal spacing of study: A multiscale context model of memory. In *Advances in Neural Information Processing Systems*, volume 22, pages 1321–1329.

P.I. Pavlik Jr and J.R. Anderson. 2008. Using a model to compute the optimal schedule of practice. *Journal of Experimental Psychology: Applied*, 14(2):101—117.

P. Pimsleur. 1967. A memory schedule. *Modern Language Journal*, 51(2):73–75.

R. Pinon and J. Haydon. 2010. The benefits of the English language for individuals and societies: Quantitative indicators from Cameroon, Nigeria, Rwanda, Bangladesh and Pakistan. Technical report, Euromonitor International for the British Council.

J.G.W. Raaijmakers. 2003. Spacing and repetition effects in human memory: Application of the sam model. *Cognitive Science*, 27(3):431–452.

T.C. Ruth. 1928. Factors influencing the relative economy of massed and distributed practice in learning. *Psychological Review*, 35:19–45.

J.E.R. Staddon, I.M. Chelaru, and J.J. Higa. 2002. Habituation, memory and the brain: The dynamics of interval timing. *Behavioural Processes*, 57(2):71–88.

M. Streeter. 2015. Mixture modeling of individual learning curves. In *Proceedings of the International Conference on Educational Data Mining (EDM)*.

M.T. Ullman. 2005. A cognitive neuroscience perspective on second language acquisition: The declarative/procedural model. In C. Sanz, editor, *Mind and Context in Adult Second Language Acquisition: Methods, Theory, and Practice*, pages 141–178. Georgetown University Press.

R. Vesselinov and J. Grego. 2012. Duolingo effectiveness study. Technical report, Queens College, City University of New York.

Wikimedia Foundation. 2002. Wiktionary: A wiki-based open content dictionary, retrieved 2012–2015. https://www.wiktionary.org.

P.A. Woźniak. 1990. Optimization of learning. Master's thesis, University of Technology in Poznań.

A Appendix

A.1 Lexeme Tagger Details

We use a lexeme tagger, introduced in §2, to analyze and index the learning corpus and student responses. Since Duolingo courses teach a moderate set of words and concepts, we do not necessarily need a complete, general-purpose, multi-lingual NLP stack. Instead, for each language we use a finite state transducer (FST) to efficiently parse candidate lexeme tags[10] for each word. We then use a hidden Markov model (HMM) to determine which tag is correct in a given context.

Consider the following two Spanish sentences: '*Yo como manzanas*' ('I eat apples') and '*Corro como el viento*' ('I run like the wind'). For both sentences, the FST parses the word *como* into the lexeme tag candidates *comer*.V.PRES.P1.SG ([I] eat) and *como*.ADV.CNJ (like/as). The HMM then disambiguates between the respective tags for each sentence. Table 5 contains a reference of the abbreviations used in this paper for lexeme tags.

Abbreviation	Meaning
ACC	accusative case
ADJ	adjective
ADV	adverb
DAT	dative case
DEF	definite
DET	determiner
FORM	formal register
F	feminine
GEN	genitive case
GER	gerund
IMPERF	imperfective aspect
INDF	indefinite
INF	infinitive
M	masculine
N	noun
NT	neuter
P1/P2/P3	1st/2nd/3rd person
PL	plural
PN	pronoun
PP	past participle
PRESP	present participle
PRES	present tense
PST	past tense
SG	singular
V	verb

Table 5: Lexeme tag component abbreviations.

A.2 Pimsleur and Leitner Models

As mentioned in §3.3, the Pimsleur and Leitner algorithms are special cases of HLR using fixed, hand-picked weights. To see this, consider the original practice interval schedule used by Pimsleur (1967): 5 sec, 25 sec, 2 min, 10 min, 1 hr, 5 hr, 1 day, 5 days, 25 days, 4 months, and 2 years. If we interpret this as a sequence of $\hat{h}_\Theta$ half-lifes (i.e., students should practice when $\hat{p}_\Theta = 0.5$), we can rewrite (2) and solve for $\log_2(\hat{h}_\Theta)$ as a linear

[10] The lexeme tag set is based on a large morphology dictionary created by the Apertium project (Forcada et al., 2011), which we supplemented with entries from Wiktionary (Wikimedia Foundation, 2002) and other sources. Each Duolingo course teaches about 3,000–5,000 lexeme tags.

equation. This yields $\Theta = \{x_n : 2.4, x_b : \text{-}16.5\}$, where x_n and x_b are the number of practices and a bias weight (intercept), respectively. This model perfectly reconstructs Pimsleur's original schedule in days ($r^2 = 0.999$, $p \ll 0.001$). Analyzing the Leitner variant from Figure 3 is even simpler: this corresponds to $\Theta = \{x_\oplus : 1, x_\ominus : \text{-}1\}$, where $x_\oplus$ is the number of past correct responses (i.e., doubling the interval), and $x_\ominus$ is the number of incorrect responses (i.e., halving the interval).

A.3 Training and Optimization Details

The complete objective function given in §3.3 for half-life regression is:

$$\begin{aligned}\ell(\langle p, \Delta, \mathbf{x}\rangle; \Theta) &= (p - \hat{p}_\Theta)^2 \\ &\quad + \alpha(h - \hat{h}_\Theta)^2 + \lambda\|\Theta\|_2^2 \,.\end{aligned}$$

Substituting (1) and (2) into this equation produces the following more explicit formulation:

$$\begin{aligned}\ell(\langle p, \Delta, \mathbf{x}\rangle; \Theta) &= \left(p - 2^{-\frac{\Delta}{2^{\Theta\cdot\mathbf{x}}}}\right)^2 \\ &\quad + \alpha\left(\frac{-\Delta}{\log_2(p)} - 2^{\Theta\cdot\mathbf{x}}\right)^2 \\ &\quad + \lambda\|\Theta\|_2^2 \,.\end{aligned}$$

In general, the search for Θ^* weights to minimize ℓ cannot be solved in closed form, but since it is a smooth function, it can be optimized using gradient methods. The partial gradient of ℓ with respect to each θ_k weight is given by:

$$\begin{aligned}\frac{\partial\ell}{\partial\theta_k} &= 2(\hat{p}_\Theta - p)\ln^2(2)\hat{p}_\Theta\left(\frac{\Delta}{\hat{h}_\Theta}\right)x_k \\ &\quad + 2\alpha\left(\hat{h}_\Theta + \frac{\Delta}{\log_2(p)}\right)\ln(2)\hat{h}_\Theta x_k \\ &\quad + 2\lambda\theta_k \,.\end{aligned}$$

In order to fit Θ to a large amount of student log data, we use AdaGrad (Duchi et al., 2011), an online algorithm for stochastic gradient descent (SGD). AdaGrad is typically less sensitive to the learning rate parameter η than standard SGD, by dynamically scaling each weight update as a function of how often the corresponding feature appears in the training data:

$$\theta_k^{(+1)} := \theta_k - \eta\left[\mathrm{c}(x_k)^{-\frac{1}{2}}\right]\frac{\partial\ell}{\partial\theta_k} \,.$$

Here $\mathrm{c}(x_k)$ denotes the number of times feature x_k has had a nonzero value so far in the SGD pass through the training data. This is useful for training stability when using large, sparse feature sets (e.g., the lexeme tag features in this study). Note that to prevent computational overflow and underflow errors, we bound $\hat{p}_\Theta \in [0.0001, 0.9999]$ and $\hat{h}_\Theta \in [15 \text{ min}, 9 \text{ months}]$ in practice.

User Modeling in Language Learning with Macaronic Texts

Adithya Renduchintala and **Rebecca Knowles** and **Philipp Koehn** and **Jason Eisner**
Department of Computer Science
Johns Hopkins University
{adi.r,rknowles,phi,eisner}@jhu.edu

Abstract

Foreign language learners can acquire new vocabulary by using cognate and context clues when reading. To measure such incidental comprehension, we devise an experimental framework that involves reading mixed-language "macaronic" sentences. Using data collected via Amazon Mechanical Turk, we train a graphical model to simulate a human subject's comprehension of foreign words, based on cognate clues (edit distance to an English word), context clues (pointwise mutual information), and prior exposure. Our model does a reasonable job at predicting which words a user will be able to understand, which should facilitate the automatic construction of comprehensible text for personalized foreign language education.

1 Introduction

Second language (L2) learning requires the acquisition of vocabulary as well as knowledge of the language's constructions. One of the ways in which learners become familiar with novel vocabulary and constructions is through reading. According to Krashen's Input Hypothesis (Krashen, 1989), learners acquire language through incidental learning, which occurs when learners are exposed to comprehensible input. What constitutes "comprehensible input" for a learner varies as their knowledge of the L2 increases. For example, a student in their first month of German lessons would be hard-pressed to read German novels or even front-page news, but they might understand brief descriptions of daily routines. Comprehensible input need not be completely familiar to the learner; it could include novel vocabulary items or structures (whose meanings they can glean from context). Such input falls in the "zone of proximal development" (Vygotskiĭ, 2012), just outside of the learner's comfort zone. The related concept of "scaffolding" (Wood et al., 1976) consists of providing assistance to the learner at a level that is just sufficient for them to complete their task, which in our case is understanding a sentence.

Automatic selection or construction of comprehensible input—perhaps online and personalized—would be a useful educational technology. However, this requires modeling the student: what can an L2 learner understand in a given context? In this paper, we develop a model and train its parameters on data that we collect.

For the remainder of the paper we focus on native English speakers learning German. Our methodology is a novel solution to the problem of controlling for the learner's German skill level. We use subjects with *zero* previous knowledge of German, but we translate portions of the sentence into English. Thus, we can presume that they *do* already know the English words and *do not* already know the German words (except from seeing them in earlier trials *within* our experiment). We are interested in whether they can jointly infer the meanings of the remaining German words in the sentence, so we ask them to guess.

The resulting stimuli are sentences like "Der Polizist arrested the Bankräuber." Even a reader with no knowledge of German is likely to be able to understand this sentence reasonably well by using cognate and context clues. We refer to this as a *macaronic* sentence; so-called macaronic language is a pastiche of two or more languages (often intended for humorous effect).

Our experimental subjects are required to guess what "Polizist" and "Bankräuber" mean in this sentence. We train a featurized model to predict these guesses jointly within each sentence and thereby predict incidental comprehension on any macaronic sentence. Indeed, we hope our model design will generalize from predicting incidental comprehension on *macaronic* sentences (for our beginner subjects, who need some context words to be in English) to predicting incidental comprehension on *full German* sentences (for more ad-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1859–1869,
Berlin, Germany, August 7-12, 2016.

vanced students, who understand some of the context words *as if* they were in English). In addition, we are developing a user interface that uses macaronic sentences directly as a medium of language instruction: our companion paper (Renduchintala et al., 2016) gives an overview of that project.

We briefly review previous work, then describe our data collection setup and the data obtained. Finally, we discuss our model of learner comprehension and validate our model's predictions.

2 Previous Work

Natural language processing (NLP) has long been applied to education, but the majority of this work focuses on evaluation and assessment. Prominent recent examples include Heilman and Madnani (2012), Burstein et al. (2013) and Madnani et al. (2012). Other works fall more along the lines of intelligent and adaptive tutoring systems designed to improve learning outcomes. Most of those are outside of the area of NLP (typically focusing on math or science). An overview of NLP-based work in the education sphere can be found in Litman (2016). There has also been work specific to second language acquisition, such as Özbal et al. (2014), where the focus has been to build a system to help learners retain new vocabulary. However, much of the existing work on incidental learning is found in the education and cognitive science literature rather than NLP.

Our work is related to Labutov and Lipson (2014), which also tries to leverage incidental learning using mixed L1 and L2 language. Where their work uses surprisal to choose contexts in which to insert L2 vocabulary, we consider both context features and other factors such as cognate features (described in detail in 4.1). We collect data that gives direct evidence of the user's understanding of words (by asking them to provide English guesses) rather than indirectly (via questions about sentence validity, which runs the risk of overestimating their knowledge of a word, if, for instance, they've only learned whether it is animate or inanimate rather than the exact meaning). Furthermore, we are not only interested in whether a mixed L1 and L2 sentence is comprehensible; we are also interested in determining a distribution over the learner's belief state for each word in the sentence. We do this in an engaging, game-like setting, which provides the user with hints when the task is too difficult for them to complete.

3 Data Collection Setup

Our method of scaffolding is to replace certain foreign words and phrases with their English translations, yielding a macaronic sentence.[1] Simply presenting these to a learner would not give us feedback on the learner's belief state for each foreign word. Even assessing the learner's reading comprehension would give only weak indirect information about what was understood. Thus, we collect data where a learner explicitly guesses a foreign word's translation when seen in the macaronic context. These guesses are then treated as supervised labels to train our user model.

We used Amazon Mechanical Turk (MTurk) to collect data. Users qualified for tasks by completing a short quiz and survey about their language knowledge. Only users whose results indicated no knowledge of German and self-identified as native speakers of English were allowed to complete tasks. With German as the foreign language, we generated content by crawling a simplified-German news website, `nachrichtenleicht.de`. We chose simplified German in order to minimize translation errors and to make the task more suitable for novice learners. We translated each German sentence using the Moses Statistical Machine Translation (SMT) toolkit (Koehn et al., 2007). The SMT system was trained on the German-English Commoncrawl parallel text used in WMT 2015 (Bojar et al., 2015).

We used 200 German sentences, presenting each to 10 different users. In MTurk jargon, this yielded 2000 Human Intelligence Tasks (HITs). Each HIT required its user to participate in several rounds of guessing as the English translation was incrementally revealed. A user was paid US$0.12 per HIT, with a bonus of US$6 to any user who accumulated more than 2000 total points.

Our HIT user interface is shown in the video at `https://youtu.be/9PczEcnr4F8`.

3.1 HITs and Submissions

For each HIT, the user first sees a German sentence[2] (Figure 1). A text box is presented below each German word in the sentence, for the user

[1] Although the language distinction is indicated by italics and color, users were left to figure this out on their own.

[2] Except that we first "translate" any German words that have identical spelling in English (case-insensitive). This includes most proper names, numerals, and punctuation marks. Such translated words are displayed in English style (blue italics), and the user is not asked to guess their meanings.

Figure 1: After a user submits a set of guesses (top), the interface marks the correct guesses in green and also reveals a set of translation clues (bottom). The user now has the opportunity to guess again for the remaining German words.

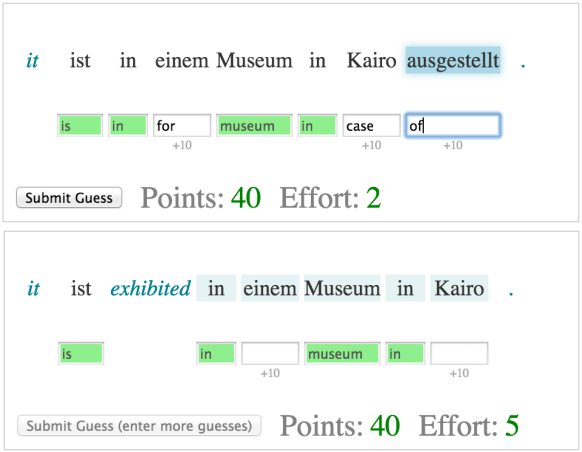


Figure 2: In this case, after the user submits a set of guesses (top), two clues are revealed (bottom): `ausgestellt` is moved into English order and then translated.

to type in their "best guess" of what each German word means. The user must fill in at least half of the text boxes before submitting this set of guesses. The resulting *submission*—i.e., the macaronic sentence together with the set of guesses—is logged in a database as a single training example, and the system displays feedback to the user about which guesses were correct.

After each submission, new *clues* are revealed (providing increased scaffolding) and the user is asked to guess again. The process continues, yielding multiple submissions, until all German words in the sentence have been translated. At this point, the entire HIT is considered completed and the user moves to a new HIT (i.e., a new sentence).

From our 2000 HITs, we obtained 9392 submissions (4.7 per HIT) from 79 distinct MTurk users.

3.2 Clues

Each update provides new clues to help the user make further guesses. There are 2 kinds of clues:

Translation Clue (Figure 1): A set of words that were originally in German are replaced with their English translations. The text boxes below these words disappear, since it is no longer necessary to guess them.

Reordering Clue (Figure 2): A German substring is moved into a more English-like position. The reordering positions are calculated using the word and phrase alignments obtained from Moses.

Each time the user submits a set of guesses, we reveal a sequence of $n = \max(1, \text{round}(N/3))$ clues, where N is the number of German words remaining in the sentence. For each clue, we sample a token that is currently in German. If the token is part of a movable phrase, we move that phrase; otherwise we translate the minimal phrase containing that token. These moves correspond exactly to clues that a user could request by clicking on the token in the macaronic reading interface of Renduchintala et al. (2016)—see that paper for details of how moves are constructed and animated. In our present experiments, the system is in control instead, and grants clues by "randomly clicking" on n tokens.

The system's probability of sampling a given token is proportional to its unigram type probability in the WMT corpus. Thus, rarer words tend to remain in German for longer, allowing the Turker to attempt more guesses for these difficult words.

3.3 Feedback

When a user submits a set of guesses, the system responds with feedback. Each guess is visibly "marked" in left-to-right order, momentarily shaded with green (for correct), yellow (for close) or red (for incorrect). Depending on whether a guess is correct, close, or wrong, users are awarded points as discussed below. Yellow and red shading then fades, to signal to the user that they may try entering a new guess. Correct guesses remain on the screen for the entire task.

3.4 Points

Adding points to the process (Figures 1–2) adds a game-like quality and lets us incentivize users by paying them for good performance (see section 3). We award 10 points for each exactly correct guess (case-insensitive). We give additional "effort points" for a guess that is close to the cor-

rect translation, as measured by cosine similarity in vector space. (We used pre-trained GLoVe word vectors (Pennington et al., 2014); when the guess or correct translation has multiple words, we take the average of the word vectors.) We deduct effort points for guesses that are careless or very poor. Our rubric for effort points is as follows:

$$e_p = \begin{cases} -1, & \text{if } \hat{e} \text{ is repeated or nonsense (red)} \\ -1, & \text{if } \mathrm{sim}(\hat{e}, e^*) < 0 \text{ (red)} \\ 0, & \text{if } 0 \leq \mathrm{sim}(\hat{e}, e^*) < 0.4 \text{ (red)} \\ 0, & \text{if } \hat{e} \text{ is blank} \\ 10 \times \mathrm{sim}(\hat{e}, e^*) & \text{otherwise (yellow)} \end{cases}$$

Here $\mathrm{sim}(\hat{e}, e^*)$ is cosine similarity between the vector embeddings of the user's guess $\hat{e}$ and our reference translation e^*. A "nonsense" guess contains a word that does not appear in the sentence bitext nor in the 20,000 most frequent word types in the GLoVe training corpus. A "repeated" guess is an incorrect guess that appears more than once in the set of guesses being submitted.

In some cases, $\hat{e}$ or e^* may itself consist of multiple words. In this case, our points and feedback are based on the best match between any word of $\hat{e}$ and any word of e^*. In alignments where multiple German words translate as a single phrase,[3] we take the phrasal translation to be the correct answer e^* for *each* of the German words.

3.5 Normalization

After collecting the data, we normalized the user guesses for further analysis. All guesses were lowercased. Multi-word guesses were crudely replaced by the longest word in the guess (breaking ties in favor of the earliest word).

The guesses included many spelling errors as well as some nonsense strings and direct copies of the input. We defined the *dictionary* to be the 100,000 most frequent word types (lowercased) from the WMT English data. If a user's guess $\hat{e}$ does not match e^* and is not in the dictionary, we replace it with

- the special symbol <COPY>, if $\hat{e}$ appears to be a copy of the German source word f (meaning that its Levenshtein distance from f is $< 0.2 \cdot \max(|\hat{e}|, |f|)$);
- else, the closest word in the dictionary[4] as measured by Levenshtein distance (breaking ties alphabetically), provided the dictionary has a word at distance ≤ 2;
- else <BLANK>, as if the user had not guessed.

4 User Model

In each submission, the user jointly guesses several English words, given spelling and context clues. One way that a *machine* could perform this task is via probabilistic inference in a factor graph—and we take this as our model of how the *human user* solves the problem.

The user observes a German sentence $\mathbf{f} = [f_1, f_2, \ldots, f_i, \ldots f_n]$. The translation of each word token f_i is E_i, which is from the user's point of view a random variable. Let Obs denote the set of indices i for which the user also observes that $E_i = e_i^*$, the aligned reference translation, because e_i^* has already been guessed correctly (green feedback) or shown as a clue. Thus, the user's posterior distribution over $\mathbf{E}$ is $P_{\boldsymbol{\theta}}(\mathbf{E} = \mathbf{e} \mid \mathbf{E}_{\text{Obs}} = \mathbf{e}^*_{\text{Obs}}, \mathbf{f}, \text{history})$, where "history" denotes the user's history of past interactions.

We assume that a user's submission $\hat{\mathbf{e}}$ is derived from this posterior distribution simply as a random sample. We try to fit the parameter vector $\boldsymbol{\theta}$ to maximize the log-probability of the submission. Note that our model is trained on the user guesses $\hat{\mathbf{e}}$, not the reference translations $\mathbf{e}^*$. That is, we seek parameters $\boldsymbol{\theta}$ that would explain why all users made their guesses.

Although we fit a single $\boldsymbol{\theta}$, this does not mean that we treat users as interchangeable (since $\boldsymbol{\theta}$ can include user-specific parameters) or unvarying (since our model conditions users' behavior on their history, which can capture some learning).

4.1 Factor Graph

We model the posterior distribution as a conditional random field (Figure 3) in which the value of E_i depends on the form of f_i as well as on the meanings e_j (which may be either observed or jointly guessed) of the context words at $j \neq i$:

$$P_{\boldsymbol{\theta}}(\mathbf{E} = \mathbf{e} \mid \mathbf{E}_{\text{Obs}} = \mathbf{e}^*_{\text{Obs}}, \mathbf{f}, \text{history}) \quad (1)$$
$$\propto \prod_{i \notin \text{Obs}} (\psi^{\text{ef}}(e_i, f_i) \cdot \prod_{j \neq i} \psi^{\text{ee}}(e_i, e_j, i-j))$$

We will define the factors ψ (the *potential functions*) in such a way that they do not "know German" but only have access to information that is available to an naive English speaker. In brief, the

[3] Our German-English alignments are constructed as in Renduchintala et al. (2016).

[4] Considering only words returned by the Pyenchant 'suggest' function (http://pythonhosted.org/pyenchant/).

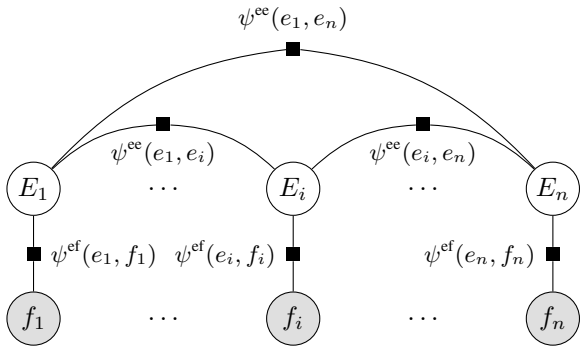


Figure 3: Model for user understanding of L2 words in sentential context. This figure shows an inference problem in which all the observed words in the sentence are in German (that is, Obs $= \emptyset$). As the user observes translations via clues or correctly-marked guesses, some of the E_i become shaded.

factor $\psi^{\text{ef}}(e_i, f_i)$ considers whether the hypothesized English word e_i "looks like" the observed German word f_i, and whether the user has previously observed during data collection that e_i is a correct or incorrect translation of f_i. Meanwhile, the factor $\psi^{\text{ee}}(e_i, e_j)$ considers whether e_i is commonly seen in the context of e_j in English text. For example, the user will elevate the probability that $E_i =$ `cake` if they are fairly certain that E_j is a related word like `eat` or `chocolate`.

The potential functions ψ are parameterized by $\boldsymbol{\theta}$, a vector of feature weights. For convenience, we define the features in such a way that we expect their weights to be positive. We rely on just 6 features at present (see section 6 for future work), although each is complex and real-valued. Thus, the weights $\boldsymbol{\theta}$ control the relative influence of these 6 different types of information on a user's guess. Our features broadly fall under the following categories: *Cognate*, *History*, and *Context*. We precomputed cognate and context features, while history features are computed on-the-fly for each training instance. All features are case-insensitive.

4.1.1 Cognate and History Features

For each German token f_i, the ψ^{ef} factor can score each possible guess e_i of its translation:

$$\psi^{\text{ef}}(e_i, f_i) = \exp(\boldsymbol{\theta}^{\text{ef}} \cdot \boldsymbol{\phi}^{\text{ef}}(e_i, f_i)) \quad (2)$$

The feature function $\boldsymbol{\phi}^{\text{ef}}$ returns a vector of 4 real numbers:

- *Orthographic Similarity*: The normalized Levenshtein distance between the 2 strings.

$$\phi^{\text{ef}}_{\text{orth}}(e_i, f_i) = 1 - \frac{\text{lev}(e_i, f_i)}{\max(|e_i|, |f_i|)} \quad (3)$$

  The weight on this feature encodes how much users pay attention to spelling.

- *Pronunciation Similarity*: This feature is similar to the previous one, except that it calculates the normalized distance between the *pronunciations* of the two words:

$$\phi^{\text{ef}}_{\text{pron}}(e_i, f_i) = \phi^{\text{ef}}_{\text{orth}}(\text{prn}(e_i), \text{prn}(f_i)) \quad (4)$$

  where the function $\text{prn}(x)$ maps a string x to its pronunciation. We obtained pronunciations for all words in the English and German vocabularies using the CMU pronunciation dictionary tool (Weide, 1998). Note that we use English pronunciation rules even for German words. This is because we are modeling a naive learner who may, in the absence of intuition about German pronunciation rules, apply English pronunciation rules to German.

- *Positive History Feature*: If a user has been rewarded in a previous HIT for guessing e_i as a translation of f_i, then they should be more likely to guess it again. We define $\boldsymbol{\phi}^{\text{ef}}_{\text{hist+}}(e_i, f_i)$ to be 1 in this case and 0 otherwise. The weight on this feature encodes whether users learn from positive feedback.

- *Negative History Feature*: If a user has already incorrectly guessed e_i as a translation of f_i in a previous submission during this HIT, then they should be less likely to guess it again. We define $\boldsymbol{\phi}^{\text{ef}}_{\text{hist-}}(e_i, f_i)$ to be -1 in this case and 0 otherwise. The weight on this feature encodes whether users remember negative feedback.[5]

4.1.2 Context Features

In the same way, the ψ^{ef} factor can score the compatibility of a guess e_i with a context word e_j, which may itself be a guess, or may be observed:

$$\psi^{\text{ee}}_{ij}(e_i, e_j) = \exp(\boldsymbol{\theta}^{\text{ee}} \cdot \boldsymbol{\phi}^{\text{ee}}(e_i, e_j, i-j)) \quad (5)$$

$\boldsymbol{\phi}^{\text{ee}}$ returns a vector of 2 real numbers:

$$\boldsymbol{\phi}^{\text{ee}}_{\text{pmi}}(e_i, e_j) = \begin{cases} \text{PMI}(e_i, e_j) \text{ if } |i-j| > 1 \\ 0 \text{ otherwise} \end{cases} \quad (6)$$

$$\boldsymbol{\phi}^{\text{ee}}_{\text{pmi1}}(e_i, e_j) = \begin{cases} \text{PMI}_1(e_i, e_j) \text{ if } |i-j| = 1 \\ 0 \text{ otherwise} \end{cases} \quad (7)$$

[5] At least in short-term memory—this feature currently omits to consider any negative feedback from previous HITs.

where the pointwise mutual information $\mathrm{PMI}(x, y)$ measures the degree to which the English words x, y tend to occur in the same English sentence, and $\mathrm{PMI}_1(x, y)$ measures how often they tend to occur in adjacent positions. These measurements are estimated from the English side of the WMT corpus, with smoothing performed as in Knowles et al. (2016).

For example, if $f_i =$ `Suppe`, the user's guess of E_i should be influenced by $f_j =$ `Brot` appearing in the same sentence, if the user suspects or observes that its translation is $E_j =$ `bread`. The PMI feature knows that `soup` and `bread` tend to appear in the same English sentences, whereas PMI_1 knows that they tend not to appear in the bigram `soup bread` or `bread soup`.

4.1.3 User-Specific Features

Apart from the basic 6-feature model, we also trained a version that includes user-specific copies of each feature (similar to the domain adaptation technique of Daumé III (2007)). For example, $\phi^{\text{ef}}_{\text{orth},32}(e_i, f_i)$ is defined to equal $\phi^{\text{ef}}_{\text{orth}}(e_i, f_i)$ for submissions by user 32, and defined to be 0 for submissions by other users.

Thus, with 79 users in our dataset, we learned 6×80 feature weights: a local weight vector for each user and a global vector of "backoff" weights. The global weight $\theta^{\text{ef}}_{\text{orth}}$ is large if users in general reward orthographic similarity, while $\theta^{\text{ef}}_{\text{orth},32}$ (which may be positive or negative) captures the degree to which user 32 rewards it more or less than is typical. The user-specific features are intended to capture individual differences in incidental comprehension.

4.2 Inference

According to our model, the probability that the user guesses $E_i = \hat{e}_i$ is given by a marginal probability from the CRF. Computing these marginals is a combinatorial optimization problem that involves reasoning jointly about the possible values of each E_i ($i \notin$ Obs), which range over the English vocabulary V^{e}.

We employ loopy belief propagation (Murphy et al., 1999) to obtain approximate marginals over the variables $\mathbf{E}$. A *tree-based* schedule for message passing was used (Dreyer and Eisner, 2009, footnote 22). We run 3 iterations with a new random root for each iteration.

We define the vocabulary V^{e} to consist of all reference translations e^*_i and normalized user guesses $\hat{e}_i$ from our entire dataset (see section 3.5), about 5K types altogether including `<BLANK>` and `<COPY>`. We define the cognate features to treat `<BLANK>` as the empty string and to treat `<COPY>` as f_i. We define the PMI of these special symbols with any e to be the mean PMI with e of all dictionary words, so that they are essentially uninformative.

4.3 Parameter Estimation

We learn our parameter vector $\boldsymbol{\theta}$ to approximately maximize the regularized log-likelihood of the users' guesses:

$$\left(\sum \log P_{\boldsymbol{\theta}}(\mathbf{E} = \hat{\mathbf{e}} \mid \mathbf{E}_{\text{Obs}} = \mathbf{e}^*_{\text{Obs}}, \mathbf{f}, \text{history})\right) - \lambda ||\boldsymbol{\theta}||^2 \quad (8)$$

where the summation is over all submissions in our dataset. The gradient of each summand reduces to a difference between observed and expected values of the feature vector $\boldsymbol{\phi} = (\boldsymbol{\phi}^{\text{ef}}, \boldsymbol{\phi}^{\text{ee}})$, summed over all factors in (1). The observed features are computed directly by setting $\mathbf{E} = \hat{\mathbf{e}}$. The expected features (which arise from the log of the normalization constant of (1)) are computed approximately by loopy belief propagation.

We trained $\boldsymbol{\theta}$ using stochastic gradient descent (SGD),[6] with a learning rate of 0.1 and regularization parameter of 0.2. The regularization parameter was tuned on our development set.

5 Experimental Results

We divided our data randomly into 5550 training instances, 1903 development instances, and 1939 test instances. Each instance was a single submission from one user, consisting of a batch of "simultaneous" guesses on a macaronic sentence.

We noted qualitatively that when a large number of English words have been revealed, particularly content words, the users tend to make better guesses. Conversely, when most context is German, we unsuprisingly see the user leave many guesses blank and make other guesses based on string similarity triggers. Such submissions are difficult to predict as different users will come up with a wide variety of guesses; our model therefore resorts to predicting similar-sounding words. For detailed examples of this see Appendix A.

[6] To speed up training, SGD was parallelized using Recht et al.'s (2011) Hogwild! algorithm. We trained for 8 epochs.

Model	Recall at k (dev)			Recall at k (test)		
	1	25	50	1	25	50
Basic	15.24	34.26	38.08	16.14	35.56	40.30
User-Adapted	15.33	34.40	38.67	16.45	35.71	40.57

Table 1: Percentage of foreign words for which the user's actual guess appears in our top-k list of predictions, for models with and without user-specific features ($k \in \{1, 25, 50\}$).

For each foreign word f_i in a submission with $i \notin$ Obs, our inference method (section 4.2) predicts a marginal probability distribution over a user's guesses $\hat{e}_i$. Table 1 shows our ability to predict user guesses.[7] Recall that this task is essentially a structured prediction task that does joint 4919-way classification of each German word. Roughly 1/3 of the time, our model's top 25 words include the user's exact guess.

However, the recall reported in Table 1 is too stringent for our educational application. We could give the model partial credit for predicting a synonym of the learner's guess $\hat{e}$. More precisely, we would like to give the model partial credit for predicting when the learner will make a poor guess of the truth e^*—even if the model does not predict the user's specific incorrect guess $\hat{e}$.

To get at this question, we use English word embeddings (as in section 3.4) as a proxy for the semantics and morphology of the words. We measure the *actual quality* of the learner's guess $\hat{e}$ as its cosine similarity to the truth, $\mathrm{sim}(\hat{e}, e^*)$. While quality of 1 is an exact match, and quality scores > 0.75 are consistently good matches, we found quality of ≈ 0.6 also reasonable. Pairs such as (`mosque`, `islamic`) and (`politics`, `government`) are examples from the collected data with quality ≈ 0.6. As quality becomes < 0.4, however, the relationship becomes tenuous, e.g., (`refugee`, `soil`).

Similarly, we measure the *predicted quality* as $\mathrm{sim}(e, e^*)$, where e is the model's 1-best prediction of the user's guess. Figure 4 plots predicted vs. actual quality (each point represents one of the learner's guesses on development data), obtaining a correlation of 0.38, which we call the "quality correlation" or QC. A clear diagonal band can be seen, corresponding to the instances where the model exactly predicts the user's guess. The cloud around the diagonal is formed by instances where the model's prediction was not identical to the user's guess but had similar quality.

[7] Throughout this section, we ignore the 5.2% of tokens on which the user did not guess (i.e., the guess was `<BLANK>` after the normalization of section 3.5). Our present model simply treats `<BLANK>` as an ordinary and very bland word (section 4.2), rather than truly attempting to predict when the user will not guess. Indeed, the model's posterior probability of `<BLANK>` in these cases is a paltry 0.0000267 on average (versus 0.0000106 when the user does guess). See section 6.

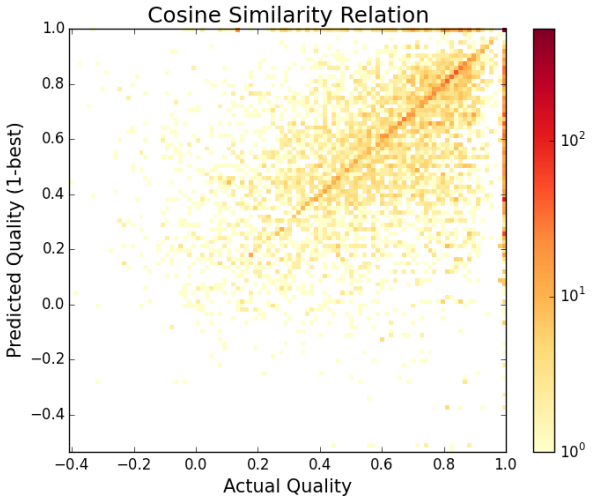


Figure 4: Actual quality $\mathrm{sim}(\hat{e}, e^*)$ of the learner's guess $\hat{e}$ on development data, versus predicted quality $\mathrm{sim}(e, e^*)$ where e is the basic model's 1-best prediction.

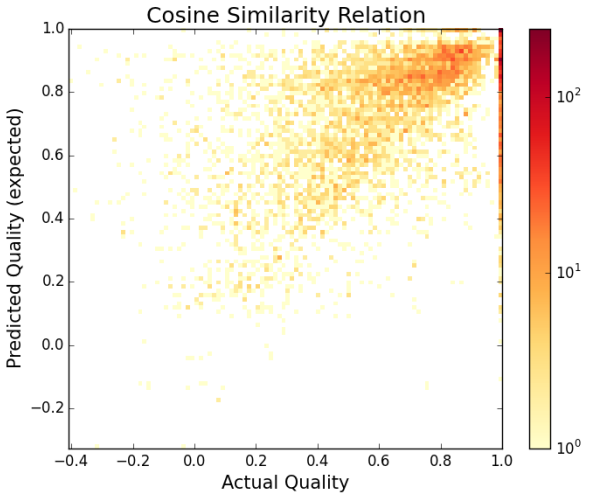


Figure 5: Actual quality $\mathrm{sim}(\hat{e}, e^*)$ of the learner's guess $\hat{e}$ on development data, versus the expectation of the predicted quality $\mathrm{sim}(e, e^*)$ where e is distributed according to the basic model's posterior.

We also consider the *expected* predicted quality, averaging over the model's predictions e of $\hat{e}$ (for all $e \in V^e$) in proportion to the probabilities that it assigns them. This allows the model to more smoothly assess whether the learner is likely to make a high-quality guess. Figure 5 shows this version, where the points tend to shift upward and the quality correlation (QC) rises to 0.53.

All QC values are given in Table 2. We used expected QC on the development set as the criterion for selecting the regularization coefficient λ and as the early stopping criterion during training.

Model	Dev		Test	
	Exp	1-Best	Exp	1-Best
Basic	0.525	0.379	0.543	0.411
User-Adapted	0.527	0.427	0.544	0.439

Table 2: Quality correlations: basic and user-adapted models.

Feature Removed	QC	
	Expected	1-Best
None	0.522	0.425
Cognate	0.516	0.366*
Context	0.510	0.366*
History	0.499*	0.259*

Table 3: Impact on quality correlation (QC) of removing features from the model. Ablated QC values marked with asterisk* differ significantly from the full-model QC values in the first row ($p < 0.05$, using the test of Preacher (2002)).

5.1 Feature Ablation

To test the usefulness of different features, we trained our model with various feature categories disabled. To speed up experimentation, we sampled 1000 instances from the training set and trained our model on those. The resulting QC values on dev data are shown in Table 3. We see that removing history-based features has the most significant impact on model performance: both QC measures drop relative to the full model. For cognate and context features, we see no significant impact on the expected QC, but a significant drop in the 1-best QC, especially for context features.

5.2 Analysis of User Adaptation

Table 2 shows that the user-specific features significantly improve the *1-best* QC of our model, although the much smaller improvement in *expected* QC is insignificant.

User adaptation allows us to discern different styles of incidental comprehension. A user-adapted model makes fine-grained predictions that could help to construct better macaronic sentences for a given user. Each user who completed at least 10 HITs has their user-specific weight vector shown as a row in Figure 6. Recall that the user-specific weights are not used in isolation, but are *added* to backoff weights shared by all users.

These user-specific weight vectors cluster into four groups. Furthermore, the average points per HIT differ by cluster (significantly between each cluster pair), reflecting the success of different strategies.[8] Users in group (a) employ a generalist

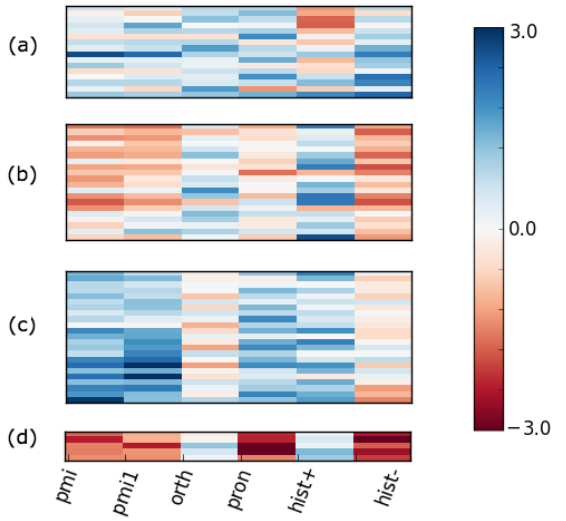


Figure 6: The user-specific weight vectors, clustered into groups. Average points per HIT for the HITs completed by each group: (a) 45, (b) 48, (c) 50 and (d) 42.

strategy for incidental comprehension. They pay typical or greater-than-typical attention to all features of the current HIT, but many of them have diminished memory for vocabulary learned during past HITs (the hist+ feature). Users in group (b) seem to use the opposite strategy, deriving their success from retaining common vocabulary across HITs (hist+) and falling back on orthography for new words. Group (c) users, who earned the most points per HIT, appear to make heavy use of context and pronunciation features *together* with hist+. We also see that pronunciation similarity seems to be a stronger feature for group (c) users, in contrast to the more superficial orthographic similarity. Group (d), which earned the fewest points per HIT, appears to be an "extreme" version of group (b): these users pay unusually little attention to any model features other than orthographic similarity and hist+. (More precisely, the model finds group (d)'s guesses harder to predict on the basis of the available features, and so gives a more uniform distribution over V^e.)

6 Future Improvements to the Model

Our model's feature set (section 4.1) could clearly be refined and extended. Indeed, in a separate paper (Knowles et al., 2016), we use a more tightly controlled experimental design to explore some simple feature variants. A cheap way to vet features would be to test whether they help on the task of modeling reference translations, which are

[8] Recall that in our data collection process, we award points for each HIT (section 3.4). While the points were designed more as a reward than as an evaluation of learner success, a higher score does reflect more guesses that were correct or close, while a lower score indicates that some words were never guessed before the system revealed them as clues.

more plentiful and less noisy than the user guesses.

For *Cognate* features, there exist many other good string similarity metrics (including trainable ones). We could also include $\boldsymbol{\phi}^{\text{ef}}$ features that consider whether e_i's part of speech, frequency, and length are plausible given f_i's burstiness, observed frequency, and length. (E.g., only short common words are plausibly translated as determiners.)

For *Context* features, we could design versions that are more sensitive to the position and status of the context word j. We speculate that the actual influence of e_j on a user's guess e_i is stronger when e_j is observed rather than itself guessed; when there are fewer intervening tokens (and particularly fewer observed ones); and when $j < i$. Orthogonally, $\boldsymbol{\phi}^{\text{ef}}(e_i, e_j)$ could go beyond PMI and windowed PMI to also consider cosine similarity, as well as variants of these metrics that are thresholded or nonlinearly transformed. Finally, we do not have to treat the context positions j as independent multiplicative influences as in equation (1) (cf. Naive Bayes): we could instead use a topic model or some form of language model to determine a conditional probability distribution over E_i given all other words in the context.

An obvious gap in our current feature set is that we have no $\boldsymbol{\phi}^{\text{e}}$ features to capture that some words $e_i \in V^{\text{e}}$ are more likely guesses *a priori*. By defining several versions of this feature, based on frequencies in corpora of different reading levels, we could learn user-specific weights modeling which users are unlikely to think of an obscure word. We should also include features that fire specifically on the reference translation e_i^* and the special symbols <BLANK> and <COPY>, as each is much more likely than the other features would suggest.

For *History* features, we could consider *negative* feedback from other HITs (not just the current HIT), as well as *positive* information provided by revealed clues (not just confirmed guesses). We could also devise non-binary versions in which more recent or more frequent feedback on a word has a stronger effect. More ambitiously, we could model generalization: after being shown that `Kind` means `child`, a learner might increase the probability that the similar word `Kinder` means `child` or something related (`children`, `childish`, ...), whether because of superficial orthographic similarity or a deeper understanding of the morphology. Similarly, a learner might gradually acquire a model of typical spelling changes in English-German cognate pairs.

A more significant extension would be to model a user's learning process. Instead of representing each user by a small vector of user-specific weights, we could recognize that the user's guessing strategy and knowledge can change over time.

A serious deficiency in our current model (not to mention our evaluation metrics!) is that we treat <BLANK> like any other word. A more attractive approach would be to learn a stochastic link from the posterior distribution to the user's guess or non-guess, instead of assuming that the user simply samples the guess from the posterior. As a simple example, we might say the user guesses $e \in V^{\text{e}}$ with probability $p(e)^\beta$—where $p(e)$ is the posterior probability and $\beta > 1$ is a learned parameter—with the remaining probability assigned to <BLANK>. This says that the user tends to avoid guessing except when there are relatively high-probability words to guess.

7 Conclusion

We have presented a methodology for collecting data and training a model to estimate a foreign language learner's understanding of L2 vocabulary in partially understood contexts. Both are novel contributions to the study of L2 acquisition.

Our current model is arguably crude, with only 6 features, yet it can already often do a reasonable job of predicting what a user might guess and whether the user's guess will be roughly correct. This opens the door to a number of future directions with applications to language acquisition using personalized content and learners' knowledge.

We plan a deeper investigation into how learners detect and combine cues for incidental comprehension. We also leave as future work the integration of this model into an adaptive system that tracks learner understanding and creates scaffolded content that falls in their zone of proximal development, keeping them engaged while stretching their understanding.

Acknowledgments

This work was supported by a seed grant from the Science of Learning Institute at Johns Hopkins University, and also by a National Science Foundation Graduate Research Fellowship (Grant No. DGE-1232825) to the second author. We thank Chadia Abras, Adam Teichert, and Sanjeev Khudanpur for helpful discussions and suggestions.

References

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. Findings of the 2015 Workshop on Statistical Machine Translation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 1–46, 2015.

Jill Burstein, Joel Tetreault, and Nitin Madnani. The e-rater automated essay scoring system. In Mark D. Shermis, editor, *Handbook of Automated Essay Evaluation: Current Applications and New Directions*, pages 55–67. Routledge, 2013.

Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of ACL*, pages 256–263, June 2007.

Markus Dreyer and Jason Eisner. Graphical models over multiple strings. In *Proceedings of EMNLP*, pages 101–110, Singapore, August 2009.

Michael Heilman and Nitin Madnani. ETS: Discriminative edit models for paraphrase scoring. In *Joint Proceedings of *SEM and SemEval*, pages 529–535, June 2012.

Rebecca Knowles, Adithya Renduchintala, Philipp Koehn, and Jason Eisner. Analyzing learner understanding of novel L2 vocabulary. 2016. To appear.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL: Short Papers*, pages 177–180, 2007.

Stephen Krashen. We acquire vocabulary and spelling by reading: Additional evidence for the input hypothesis. *The Modern Language Journal*, 73(4):440–464, 1989.

Igor Labutov and Hod Lipson. Generating code-switched text for lexical learning. In *Proceedings of ACL*, pages 562–571, 2014.

Diane Litman. Natural language processing for enhancing teaching and learning. In *Proceedings of AAAI*, 2016.

Nitin Madnani, Michael Heilman, Joel Tetreault, and Martin Chodorow. Identifying high-level organizational elements in argumentative discourse. In *Proceedings of NAACL-HLT*, pages 20–28, 2012.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of UAI*, pages 467–475, 1999.

Gözde Özbal, Daniele Pighin, and Carlo Strapparava. Automation and evaluation of the keyword method for second language learning. In *Proceedings of ACL (Volume 2: Short Papers)*, pages 352–357, 2014.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GLoVe: Global vectors for word representation. In *Proceedings of EMNLP*, volume 14, pages 1532–1543, 2014.

K. J. Preacher. Calculation for the test of the difference between two independent correlation coefficients [computer software], May 2002.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

Adithya Renduchintala, Rebecca Knowles, Philipp Koehn, and Jason Eisner. Creating interactive macaronic interfaces for language learning. In *Proceedings of ACL (System Demonstrations)*, 2016.

Lev Vygotskiĭ. *Thought and Language (Revised and Expanded Edition)*. MIT Press, 2012.

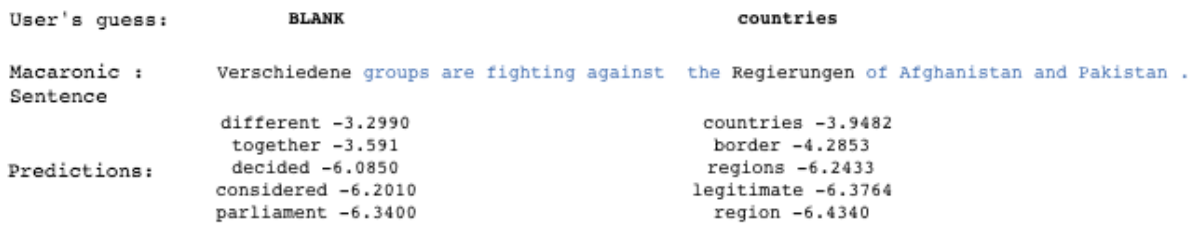
R. Weide. The CMU pronunciation dictionary, release 0.6, 1998.

David Wood, Jerome S. Bruner, and Gail Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17(2): 89–100, 1976.

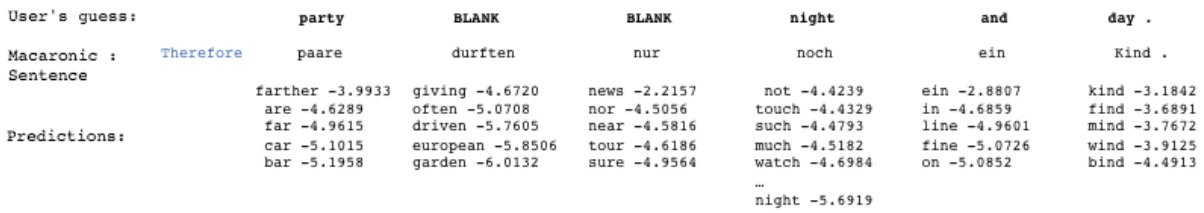
Appendices

A Example of Learner Guesses vs. Model Predictions

To give a sense of the problem difficulty, we have hand-picked and presented two training examples (submissions) along with the predictions of our basic model and their log-probabilities. In Figure 7a a large portion of the sentence has been revealed to the user in English (blue text) only 2 words are in German. The text in bold font is the user's guess. Our model expected both words to be guessed; the predictions are listed below the German words `Verschiedene` and `Regierungen`. The reference translation for the 2 words are `Various` and `governments`. In Figure 7b we see a much harder context where only one word is shown in English and this word is not particularly helpful as a contextual anchor.



(a)



(b)

Figure 7: Two examples of the system's predictions of what the user will guess on a single submission, contrasted with the user's actual guess. (The user's previous submissions on the same task instance are not shown.) In 7a, the model correctly expects that the substantial context will inform the user's guess. In 7b, the model predicts that the user will fall back on string similarity—although we can see that the user's actual guess of `and day` was likely informed by their guess of `night`, an influence that our CRF did consider. The numbers shown are log-probabilities. Both examples show the sentences in a macaronic state (after some reordering or translation has occurred). For example, the original text of the German sentence in 7b reads `Deshalb durften die Paare nur noch ein Kind bekommen .` The macaronic version has undergone some reordering, and has also erroneously dropped the verb due to an incorrect alignment.

On the Similarities Between Native, Non-native and Translated Texts

Ella Rabinovich[△⋆] **Sergiu Nisioi**[◇] **Noam Ordan**[†] **Shuly Wintner**[⋆]
[△]IBM Haifa Research Labs
[⋆]Department of Computer Science, University of Haifa
[◇]Solomon Marcus Center for Computational Linguistics, University of Bucharest
[†]The Arab College for Education, Haifa
{ellarabi,sergiu.nisioi,noam.ordan}@gmail.com, shuly@cs.haifa.ac.il

Abstract

We present a computational analysis of three language varieties: native, advanced non-native, and translation. Our goal is to investigate the similarities and differences between non-native language productions and translations, contrasting both with native language. Using a collection of computational methods we establish three main results: (1) the three types of texts are easily distinguishable; (2) non-native language and translations are closer to each other than each of them is to native language; and (3) some of these characteristics depend on the source or native language, while others do not, reflecting, perhaps, unified principles that similarly affect translations and non-native language.

1 Introduction

This paper addresses two linguistic phenomena: translation and non-native language. Our main goal is to investigate the similarities and differences between these two phenomena, and contrast them with native language. In particular, we are interested in the reasons for the differences between translations and originals, on one hand, and native and non-native language, on the other. Do they reflect "universal" principles, or are they dependent on the source/native language?

Much research in translation studies indicates that translated texts have unique characteristics. Translated texts (in any language) constitute a sublanguage of the target language, sometimes referred to as *translationese* (Gellerstam, 1986). The unique characteristics of translationese have been traditionally classified into two categories: properties that stem from *interference* of the source language (Toury, 1979), and universal traits resulting from the translation process itself, independently of the specific source and target languages (Baker, 1993; Toury, 1995). The latter so-called *translation universals* have triggered a continuous debate among translation studies researchers (Mauranen and Kujamäki, 2004; House, 2008; Becher, 2010).

Similarly, over half a century of research on second language acquisition (SLA) established the presence of *cross-linguistic influences* (CLI) in non-native utterances (Jarvis and Pavlenko, 2008). CLI is a cover term proposed by Kellerman and Sharwood-Smith (1986) to denote various phenomena that stem from language contact situations such as transfer, interference, avoidance, borrowing, etc.[1] In addition, universal traits resulting from the learning process itself have been noticed regardless of the native language, L1.[2] For example, similar developmental sequences have been observed for negation, question formation, and other sentence structures in English (Dulay and Burt, 1974; Odlin, 1989) for both Chinese and Spanish natives. Phenomena such as overgeneralization, strategies of learning (Selinker, 1972), psychological factors (Ellis, 1985), and cultural distance (Giles and Byrne, 1982) are also influential in the acquisition process.

There are clear similarities between translations and non-native language: both are affected by the simultaneous presence of (at least) two linguistic systems, which may result in a higher cognitive load (Shlesinger, 2003). The presence of the L1 may also cause similar CLI effects on the target language.

On the other hand, there are reasons to believe

[1]To avoid terminological conflicts, we shall henceforth use CLI to denote any influence of one linguistic system over another, w.r.t. both translations and non-native productions.

[2]For simplicity, we will use *L1* to refer both to the native language of a speaker and to the source language of a translated text. We use *target language* to refer to second and translation languages (English in this paper).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1870–1881,
Berlin, Germany, August 7-12, 2016.

that translationese and non-native language should differ from each other. Translations are produced by *native* speakers of the target language. Non-natives, in contrast, arguably never attain native-like abilities (Coppieters, 1987; Johnson and Newport, 1991), however this hypothesis is strongly debated in the SLA community (Birdsong, 1992; Lardiere, 2006).

Our goal in this work is to investigate three language *varieties*: the language of native speakers (N), the language of advanced, highly fluent non-native speakers (NN), and translationese (T). We use the term *constrained language* to refer to the latter two varieties. We propose a unified computational umbrella for exploring two related areas of research on bilingualism: translation studies and second language acquisition. Specifically, we put forward three main hypotheses: (1) The three language varieties have unique characteristics that make them easily distinguishable. (2) Non-native language and translations are closer to each other than either of them is to native language. (3) Some of these characteristics are dependent on the specific L1, but many are not, and may reflect unified principles that similarly affect translations and non-native language.

We test these hypotheses using several corpus-based computational methods. We use supervised and unsupervised classification (Section 4) to show that the three language varieties are easily distinguishable. In particular, we show that native and advanced non-native productions can be accurately separated. More pertinently, we demonstrate that non-native utterances and translations comprise two distinct linguistic systems.

In Section 5, we use statistical analysis to explore the unique properties of each language variety. We show that the two varieties of constrained language are much closer to each other than they are to native language: they exhibit poorer lexical richness, a tendency to use more frequent words, a different distribution of idiomatic expressions and pronouns, and excessive use of cohesive devices. This is an unexpected finding, given that both natives and translators (in contrast to non-natives) produce texts in their mother tongue.

Finally, in Section 6 we use language modeling to show that translations and non-native language exhibit similar statistical properties that clearly reflect cross-linguistic influences: experiments with distinct language families reveal salient ties between the two varieties of constrained language.

The main contribution of this work is thus theoretical: it sheds light on some fundamental questions regarding bilingualism, and we expect it to motivate and drive future research in both SLA and translation studies. Moreover, a better understanding of constrained language may also have some practical import, as we briefly mention in the following section.

2 Related work

Corpus-based investigation of translationese has been a prolific field of recent research, laying out an empirical foundation for the theoretically motivated hypotheses on the characteristics of translationese. More specifically, identification of translated texts by means of automatic classification shed light on the manifestation of translation universals and cross-linguistic influences as markers of translated texts (Baroni and Bernardini, 2006; van Halteren, 2008; Gaspari and Bernardini, 2008; Kurokawa et al., 2009; Koppel and Ordan, 2011; Ilisei and Inkpen, 2011; Volansky et al., 2015; Rabinovich and Wintner, 2015; Nisioi, 2015b), while Gaspari and Bernardini (2008) introduced a dataset for investigation of potential common traits between translations and non-native texts. Such studies prove to be important for the development of parallel corpora (Resnik and Smith, 2003), the improvement in quality of plagiarism detection (Potthast et al., 2011), language modeling, and statistical machine translation (Lembersky et al., 2012, 2013).

Computational approaches also proved beneficial for theoretical research in second language acquisition (Jarvis and Pavlenko, 2008). Numerous studies address linguistic processes attributed to SLA, including automatic detection of highly competent non-native writers (Tomokiyo and Jones, 2001; Bergsma et al., 2012), identification of the mother tongue of English learners (Koppel et al., 2005; Tetreault et al., 2013; Tsvetkov et al., 2013; Nisioi, 2015a) and typology-driven error prediction in learners' speech (Berzak et al., 2015). These studies are instrumental for language teaching and student evaluation (Smith and Swan, 2001), and can improve NLP applications such as authorship profiling (Estival et al., 2007) or grammatical error correction (Chodorow et al., 2010). Most of these studies utilize techniques that are motivated by the same abstract principles associ-

ated with L1 influences on the target language.

To the best of our knowledge, our work is the first to address both translations and non-native language under a unifying computational framework, and in particular to compare both with native language.

3 Methodology and experimental setup

3.1 Dataset

Our dataset[3] is based on the highly homogeneous corpus of the European Parliament Proceedings (Koehn, 2005). Note that the proceedings are produced as follows: (1) the utterances of the speakers are transcribed; (2) the transcriptions are sent to the speaker who may suggest minimal editing without changing the content; (3) the edited version is then translated by native speakers. Note in particular that the texts are *not* a product of simultaneous interpretation.

In this work we utilize a subset of Europarl in which each sentence is manually annotated with speaker information, including the EU state represented and the original language in which the sentence was uttered (Nisioi et al., 2016). The texts in the corpus are uniform in terms of style, respecting the European Parliament's formal standards. Translations are produced by native English speakers and all non-native utterances are selected from members not representing UK or Ireland. Europarl N consists of texts delivered by native speakers from England.

Table 1 depicts statistics of the dataset.[4] In contrast to other learner corpora such as ICLE (Granger, 2003), EFCAMDAT (Geertzen et al., 2013) or TOEFL-11 (Blanchard et al., 2013), this corpus contains translations, native, and non-native English of high proficiency speakers. Members of the European Parliament have the right to use any of the EU's 24 official languages when speaking in Parliament, and the fact that some of them prefer to use English suggests a high degree of confidence in their language skills.

3.2 Preprocessing

All datasets were split by sentence, cleaned (text lowercased, punctuation and empty lines removed) and tokenized using the Stanford tools (Manning et al., 2014). For the classification experiments we randomly shuffled the sentences within each language variety to prevent interference of other artifacts (e.g., authorship, topic) into the classification procedure. We divided the data into chunks of approximately 2,000 tokens, respecting sentence boundaries, and normalized the values of lexical features by the number of tokens in each chunk. For classification we used Platt's sequential minimal optimization algorithm (Keerthi et al., 2001; Hall et al., 2009) to train support vector machine classifiers with the default linear kernel.

sub-corpus	sentences	tokens	types
native (N)	60,182	1,589,215	28,004
non-native (NN)	29,734	783,742	18,419
translated (T)	738,597	22,309,296	71,144
total	828,513	24,682,253	117,567

Table 1: Europarl corpus statistics: native, non-native and translated texts.

In all the experiments we used (the maximal) equal amount of data from each category, thus we always randomly down-sampled the datasets in order to have a comparable number of examples in each class; specifically, 354 chunks were used for each language variety: N, NN and T.

3.3 Features

The first feature set we utilized for the classification tasks comprises *function words* (FW), probably the most popular choice ever since Mosteller and Wallace (1963) used it successfully for the Federalist Papers. Function words proved to be suitable features for multiple reasons:(1) they abstract away from contents and are therefore less biased by topic; (2) their frequency is so high that by and large they are assumed to be selected unconsciously by authors; (3) although not easily interpretable, they are assumed to reflect grammar, and therefore facilitate the study of how structures are carried over from one language to another. We used the list of approximately 400 function words provided in Koppel and Ordan (2011).

A more informative way to capture (admittedly shallow) syntax is to use *part-of-speech (POS) trigrams*. Triplets such as PP (personal pronoun) + VHZ (*have*, 3sg present) + VBN (*be*, past participle) reflect a complex tense form, represented distinctively across languages. In Europarl, for example, this triplet is highly frequent in translations

[3] The dataset is available at `http://nlp.unibuc.ro/resources.html`

[4] Appendix A provides details on the distribution of NN and T texts by various L1s.

from Finnish and Danish and much rarer in translations from Portuguese and Greek. In this work we used the top-3,000 most frequent POS trigrams in each corpus.

We also used *positional token frequency* (Grieve, 2007). The feature is defined as counts of words occupying the first, second, third, penultimate and last positions in a sentence. The motivation behind this feature is that sentences open and close differently across languages, and it should be expected that these opening and closing devices will be transferred from L1 if they do not violate the grammaticality of the target language. Positional tokens were previously used for translationese identification (Volansky et al., 2015) and for native language detection (Nisioi, 2015a).

Translations are assumed to exhibit *explicitation*: the tendency to render implicit utterances in the source text more explicit in the translation product. For example, causality, even though not always explicitly expressed in the source, is expressed in the target by the introduction of cohesive markers such as *because*, *due to*, etc. (Blum-Kulka, 1986). Similarly, Hinkel (2001) conducted a comparative analysis of *explicit cohesive devices* in academic texts by non-native English students, and found that cohesive markers are distributed differently in non-native English productions, compared to their native counterparts. To study this phenomenon, we used the set of over 100 cohesive markers introduced in Hinkel (2001).

4 The status of constrained language

To establish the unique nature of each language variety in our dataset, we perform multiple pairwise binary classifications between N, NN, and T, as well as three-way classifications. Table 2 reports the results; the figures reflect average tenfold cross-validation accuracy (the best result in each column is boldfaced).

In line with previous works (see Section 2), classification of N–T, as well as N–NN, yields excellent results with most features and feature combinations. NN–T appears to be easily distinguishable as well; specifically, FW+POS-trigrams combination with/without positional tokens yields 99.57% accuracy. The word *maybe* is among the most discriminative feature for NN vs. T, being overused in NN, as opposed to *perhaps*, which exhibits a much higher frequency in T; this may indicate a certain degree of formality, typical of translated texts (Olohan, 2003). The words *or*, *which* and *too* are considerably more frequent in T, implying higher sentence complexity. This trait is also reflected by shorter NN sentences, compared to T: the average sentence length in Europarl is 26 tokens for NN vs. 30 for T. Certain decisiveness devices (*sure*, *very*) are underused in T, in accordance with Toury (1995)'s law of standardization (Vanderauwera, 1985). The three-way classification yields excellent results as well; the highest accuracy is obtained using FW+positional tokens with/without POS-trigrams.

feature / dataset	N-NN	N-T	NN-T	3-way
FW	98.72	98.72	96.89	96.60
POS (trigrams)	97.45	98.02	97.45	95.10
pos. tok	99.01	99.01	98.30	98.11
cohesive markers	85.59	87.14	82.06	74.19
FW+POS	99.43	99.57	**99.57**	99.34
FW+pos. tok	99.71	99.85	98.30	**99.52**
POS+pos. tok	99.57	99.57	99.01	99.15
FW+POS+pos. tok	**99.85**	**99.85**	**99.57**	**99.52**

Table 2: Pairwise and three-way classification results of N, NN and T texts.

A careful inspection of the results in Table 2 reveals that NN–T classification is a slightly yet systematically harder task than N–T or N–NN; this implies that NN and T texts are more similar to each other than either of them is to N.

To emphasize this last point, we analyze the separability of the three language varieties by applying unsupervised classification. We perform *bisecting KMeans* clustering procedure previously used for unsupervised identification of translationese by Rabinovich and Wintner (2015). Clustering of N, NN and T using function words into three clusters yields high accuracy, above 90%. For the sake of clusters' visualization in a bidimensional plane, we applied principal component analysis for dimensionality reduction.

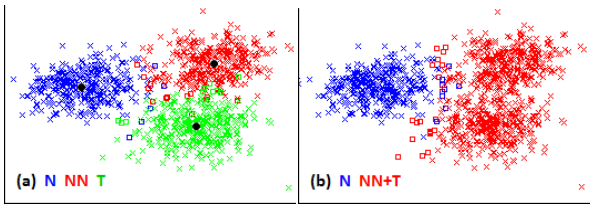


Figure 1: Clustering of N, NN and T into three (a) and two (b) clusters using function words. Clusters' centroids in (a) are marked by black circles; square sign stands for instances clustered wrongly.

The results are depicted in Figure 1 (a). Evidently, NN and T exhibit higher mutual proximity than either of them with N. Fixing the number of expected clusters to 2 further highlights this observation, as demonstrated in Figure 1 (b): both NN and T instances were assigned to a single cluster, distinctively separable from the N cluster.

We conclude that the three language varieties (N, NN, and T) constitute three different, distinguishable ontological categories, characterized by various lexical, syntactic and grammatical properties; in particular, the two varieties of constrained language (NN and T) represent two distinct linguistic systems. Nevertheless, we anticipate NN and T to share more common tendencies and regularities, when compared to N. In the following sections, we put this hypothesis to the test.

5 L1-independent similarities

In this section we address L1-independent similarities between NN and T, distinguishing them from N. We focus on characteristics which are theoretically motivated by translation studies and which are considered to be L1-independent, i.e., unrelated to cross-linguistic influences. We hypothesize that linguistic devices over- or under-represented in translation would behave similarly in highly competent non-native productions, compared to native texts.

To test this hypothesis, we realized various linguistic phenomena as properties that can be easily computed from N, NN and T texts. We refer to the computed characteristics as *metrics*. Our hypothesis is that NN metric values will be similar to T, and that both will differ from N. We used equally-sized texts of 780K tokens for N, NN and T; the exact computation is specified for each metric.

For the sake of visualization, the three values of each metric (for N, NN and T) were zero-one scaled by total-sum normalization. Figure 2 graphically depicts the normalized metric values. We now describe and motivate each metric. We analyze the results in Section 5.1 and establish their statistical significance in Section 5.2.

Lexical richness Translated texts tend to exhibit less lexical diversity (Al-Shabab, 1996). Blum-Kulka (1986) suggested that translated texts *make do with less words*, which is reflected by their lower type-to-token ratio (TTR) compared to that of native productions. We computed the TTR metric by dividing the number of unique (lemmatized) tokens by the total number of tokens.

Mean word rank Halverson (2003) claims that translators use more prototypical language, i.e., *they regress to the mean* (Shlesinger, 1989). We, therefore, hypothesize that rarer words are used more often in native texts than in non-native productions and translationese. To compute this metric we used a BNC-based ranked list of 50K English words[5], excluding the list of function words (see Section 3.3). The metric value was calculated by averaging the rank of all tokens in a text; tokens that do not appear in the list of 50K were excluded.

Collocations Collocations are distributed differently in translations and in originals (Toury, 1980; Kenny, 2001). Common and frequent collocations are used almost subconsciously by native speakers, but will be subjected to a more careful choice by translators and, presumably, by fluent non-native speakers (Erman et al., 2014). For example, the phrase *make sure* appears twice more often in native Europarl texts than in NN, and five times more than in T; *bear in mind* has almost double frequency in N, compared to NN and T. Expressions such as: *bring forward*, *figure out*, *in light of*, *food chain* and *red tape* appear dozens of times in N, as opposed to zero occurrences in NN and T Europarl texts. This metric is defined by computing the frequency of idiomatic expressions[6] in terms of types.

Cohesive markers Translations were proven to employ cohesion intensively (Blum-Kulka, 1986; Øverås, 1998; Koppel and Ordan, 2011). Non-native texts tend to use cohesive markers differently as well: *sentence transitions*, the major cohesion category, was shown to be overused by non-native speakers regardless of their native language (Hinkel, 2001). The metric is defined as the frequency of sentence transitions in the three language varieties.

Qualitative comparison of various markers between NN and T productions, compared to N in the Europarl texts, highlights this phenomenon: *in addition* is twice as frequent in NN and T than in N; *according*, *at the same time* and *thus* occur three times more frequently in NN and T, compared to N; *moreover* is used four times more fre-

[5] `https://www.kilgarriff.co.uk` we used the list extracted from both spoken and written text.

[6] Idioms were taken from `https://en.wiktionary.org/wiki/Category:English_idioms`. The list was minimally cleaned up.

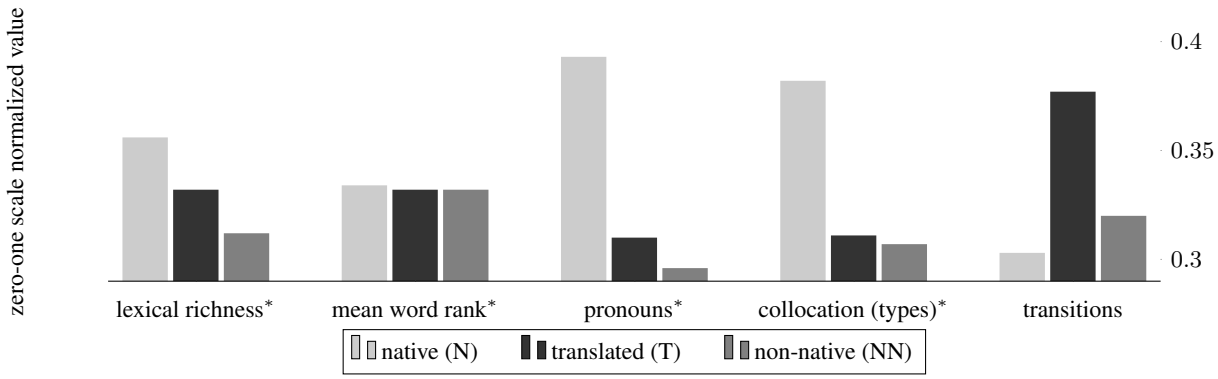


Figure 2: Metric values in N, NN and T. Tree-way differences are significant in all metric categories and "*" indicates metrics with higher pairwise similarity of NN and T, compared individually to N.

quently; and *to conclude* is almost six times more frequent.

Personal pronouns We expect both non-native speakers and translators to spell out entities (both nouns and proper nouns) more frequently, as a means of *explicitation* (Olohan, 2002), thus leading to under-use of personal pronouns, in contrast to native texts. As an example, *his* and *she* are twice more frequent in N than in NN and T.

We define this metric as the frequency of (all) personal and possessive pronouns used in the three language varieties. The over-use of personal pronouns in N utterances, is indeed balanced out by lower frequency of proper and regular nouns in these texts, compared to T and NN.[7]

5.1 Analysis

Evidently (see Figure 2), translationese and non-native productions exhibit a consistent pattern in both datasets, compared to native texts: NN and T systematically demonstrate lower metric values than N for all characteristics (except sentence transitions, where both NN and T expectedly share a higher value). All metrics except mean word rank exhibit substantial (sometimes dramatic) differences between N, on the one hand, and NN and T, on the other, thus corroborating our hypothesis. Mean word rank exhibits a more moderate variability in the three language varieties, yielding near identical value in NN and T; yet, it shows excessive usage in N.

The differences between metric values are statistically significant for all metrics (Section 5.2). Moreover, in all cases (except transitions), the difference between NN and T metrics is significantly lower than the difference between either of them and N, implying a higher proximity of NN and T distributions, compared individually to N. This finding further emphasizes the common tendencies between NN and T.

As shown in Figure 2, NN and T are systematically and significantly different from N. Additionally, we can see that T is consistently positioned between N and NN (except for sentence transitions), implying that translations produced by native speakers tend to resemble native utterances to a higher degree than non-native productions.

5.2 Statistical significance

Inspired by the results depicted in Figure 2, we now put to test two statistical hypotheses: (1) N, NN and T productions do not represent identical underlying distributions, i.e., at least one pair is distributed differently; and consequently, (2) NN and T productions exhibit higher similarity (in terms of *distance*) than either of them with N. We test these hypotheses by applying the *bootstrapping* statistical analysis.

Bootstrapping is a statistical technique involving random re-sampling (with replacement) from the original sample; it is often used to assign a measure of accuracy (e.g., a confidence interval) to an estimate. Specifically, let C_{N}, C_{NN} and C_{T} denote native, non-native and translated sub-corpora of equal size (780K tokens). Let C_{ALL} denote the concatenation of all three sub-corpora, resulting in a total of 2,340M tokens. We further denote a function computing a metric m by f^m; when applied to C, its value is $f^m(C)$. The sum of pair-

[7] Normalized frequencies of nouns and proper nouns are 0.323, 0.331 and 0.345 for N, T, and NN, respectively.

wise distances between the three individual dataset metrics is denoted by D_{total}:

$$D_{\text{total}} = |f^m(C_{\text{N}}) - f^m(C_{\text{NN}})| + |f^m(C_{\text{N}}) - f^m(C_{\text{T}})| + |f^m(C_{\text{NN}}) - f^m(C_{\text{T}})|$$

High values of D_{total} indicate a difference between the three language varieties. To examine whether the observed D_{total} is high beyond chance level, we use the bootstrap approach, and repeat the following process 1,000 times:[8] we sample C_{ALL} with replacement (at sentence granularity), generating in the j-th iteration equal-sized samples $\widehat{C_{\text{N}}}^j$, $\widehat{C_{\text{NN}}}^j$, $\widehat{C_{\text{T}}}^j$. The corresponding distance estimate, therefore, is:

$$\widehat{D_{\text{total}}}^j = |f^m(\widehat{C_{\text{N}}}^j) - f^m(\widehat{C_{\text{NN}}}^j)| + |f^m(\widehat{C_{\text{N}}}^j) - f^m(\widehat{C_{\text{T}}}^j)| + |f^m(\widehat{C_{\text{NN}}}^j) - f^m(\widehat{C_{\text{T}}}^j)|$$

We repeat random re-sampling and computation of $\widehat{D_{\text{total}}}^j$ 1,000 times, and estimate the p-value of $\widehat{D_{\text{total}}}$ by calculation of its percentile within the series of (sorted) $\widehat{D_{\text{total}}}^j$ values, where $j \in (1, \ldots, 1000)$. In all our experiments the original distance D_{total} exceeds the maximum estimate in the series of $\widehat{D_{\text{total}}}^j$, implying highly significant difference, with p-value<0.001 for all metrics.

In order to stress this outcome even further, we now test whether (the constrained) NN and T exhibit higher pairwise similarity, as opposed to N. We achieve this by assessment of the distance between NN and T productions, compared to the distance between N and its closest production (again, in terms of distance): either NN or T. We sample C_{N}, C_{NN} and C_{T} (with replacement) separately, constructing $\widetilde{C_{\text{N}}}$, $\widetilde{C_{\text{NN}}}$ and $\widetilde{C_{\text{T}}}$, respectively, and define the following distance function:

$$\widetilde{D_{\text{dif}}}^j = |f^m(\widetilde{C_{\text{N}}}^j) - f^m(\widetilde{C}_K^j)| - |f^m(\widetilde{C_{\text{NN}}}^j) - f^m(\widetilde{C_{\text{T}}}^j)|$$

where

$$K = \begin{cases} \text{NN} & \text{if } |f^m(C_{\text{N}}) - f^m(C_{\text{NN}})| < |f^m(C_{\text{N}}) - f^m(C_{\text{T}})| \\ \text{T} & \text{otherwise} \end{cases}$$

We repeat re-sampling and computation of $\widetilde{D_{\text{dif}}}^j$ 1,000 times for each metric value in both datasets and sort the results. The end points of the 95% confidence interval are defined by estimate values with 2.5% deviation from the minimum (*min-end-point*) and the maximum (*max-end-point*) estimates. We assess the p-value of the test by inspecting the estimate underlying the min-end-point; specifically, in case the min-end-point is greater than 0, we consider $p<0.05$. Metric categories exhibiting higher NN-T similarity than either N-NN or N-T are marked with "*" in Figure 2.

[8] This sample size is proven sufficient by the highly significant results (very low p-value).

6 L1-related similarities

We hypothesize that both varieties of constrained language exhibit similar (lexical, grammatical, and structural) patterns due to the influence of L1 over the target language. Consequently, we anticipate that non-native productions of speakers of a certain native language (L1) will be closer to translations from L1 than to translations from other languages.

Limited by the amount of text available for each individual language, we set out to test this hypothesis by inspection of two language *families*, Germanic and Romance. Specifically, the Germanic family consists of NN texts delivered by speakers from Austria, Germany, Netherlands and Sweden; and the Romance family includes NN speakers from Portugal, Italy, Spain, France and Romania. The respective T families comprise translations from Germanic and Romance originals, corresponding to the same countries. Table 3 provides details on the datasets.

	sentences	**tokens**	**types**
Germanic NN	5,384	132,880	7,841
Germanic T	269,222	7,145,930	43,931
Romance NN	6,384	180,416	9,838
Romance T	307,296	9,846,215	49,925

Table 3: Europarl Germanic and Romance families: NN and T.

We estimate L1-related traces in the two varieties of constrained language by the fitness of a translationese-based *language model* (LM) to utterances of non-native speakers from the same language family. Attempting to trace structural and grammatical, rather than content similarities, we compile five-gram *POS* language models from Germanic and Romance translationese (GerT and RomT, respectively).[9] We examine the predic-

[9] For building LMs we used the closed vocabulary of Penn

tion power of these models on non-native productions of speakers with Germanic and Romance native languages (GerNN and RomNN), hypothesizing that an LM compiled from Germanic translationese will better predict non-native productions of a Germanic speaker and vice versa. The fitness of a language model to a set of sentences is estimated in terms of *perplexity* (Jelinek et al., 1977).

For building and estimating language models we used the KenLM toolkit (Heafield, 2011), employing modified Kneser-Ney smoothing without pruning. Compilation of language-family-specific models was done using 7M tokens of Germanic and Romance translationese each; the test data consisted of 5350 sentences of Germanic and Romance non-native productions. Consequently, for perplexity experiments with individual languages we utilized 500 sentences from each language. We excluded OOVs from all perplexity computations.

Table 4 reports the results. Prediction of GerNN by the GerT language model yields a slightly lower perplexity (i.e., a better prediction) than prediction by RomT. Similarly, RomNN is much better predicted by RomT than by GerT. These differences are statistically significant: we divided the NN texts into 50 chunks of 100 sentences each, and computed perplexity values by the two LMs for each chunk. Significance was then computed by a two-tailed paired t-test, yielding p-values of 0.015 for GerNN and 6e-22 for RomNN.

LM / NN	GerNN	LM / NN	RomNN
GerT	8.77	GerT	8.64
RomT	8.79	RomT	8.43

Table 4: Perplexity: fitness of Germanic and Romance translationese LMs to Germanic and Romance NN test sets.

As a further corroboration of the above result, we computed the perplexity of the GerT and RomT language models with respect to the language of NN speakers, this time distinguishing speakers by their country of origin. We used the same language models and non-native test chunks of 500 sentences each. Inspired by the outcome of the previous experiment, we expect that NN productions by Germanic speakers will be better predicted by GerT LM, and vice versa. Figure 3 presents a scatter plot with the results.

A clear pattern, evident from the plot, reveals

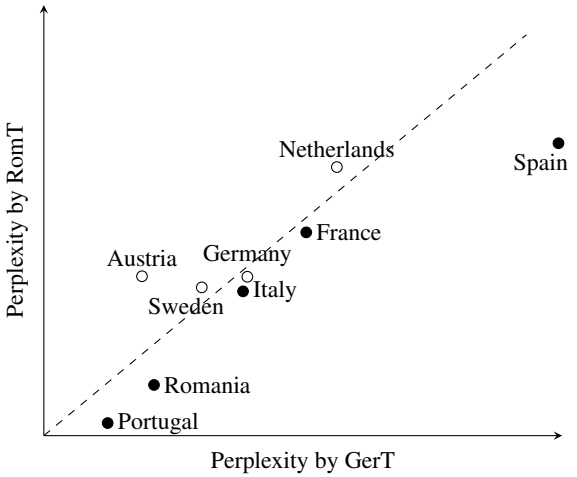


Figure 3: Perplexity of the GerT and RomT language models with respect to non-native utterances of speakers from various countries.

that all English texts with underlying Romance native languages (under the diagonal) are better predicted (i.e., obtain lower perplexity) by the RomT LM. All Germanic native languages (except German), on the other hand, are better predicted by the GerT LM. This finding further supports the hypothesis that non-native productions and translationese tend to exhibit similar L1-related traits.

7 Conclusion

We presented a unified computational approach for studying constrained language, where many of the features were theoretically motivated. We demonstrated that while translations and non-native productions are two distinct language varieties, they share similarities that stem from lower lexical richness, more careful choice of idiomatic expressions and pronouns, and (presumably) subconscious excessive usage of explicitation cohesive devices. More dramatically, the language modeling experiments reveal salient ties between the native language of non-native speakers and the source language of translationese, highlighting the unified L1-related traces of L1 in both scenarios. Our findings are intriguing: native speakers and translators, in contrast to non-native speakers, use their native language, yet translation seems to gravitate towards non-native language use.

The main contribution of this work is empirical, establishing the connection between these types of language production. While we believe that these common tendencies are not incidental, more research is needed in order to establish a theoretical

Treebank POS tag set.

explanation for the empirical findings, presumably (at least partially) on the basis of the cognitive load resulting from the simultaneous presence of two linguistic systems. We are interested in expanding the preliminary results of this work: we intend to replicate the experiments with more languages and more domains, investigate additional varieties of constrained language and employ more complex lexical, syntactic and discourse features. We also plan to investigate how the results vary when limited to specific L1s.

Acknowledgments

This research was supported by the Israeli Ministry of Science and Technology. We are immensely grateful to Yuval Nov for much advice and helpful suggestions. We are also indebted to Roy Bar-Haim, Anca Bucur, Liviu P. Dinu, and Yosi Mass for their support and guidance during the early stages of this work, and we thank our anonymous reviewers for their valuable insights.

References

Omar S. Al-Shabab. 1996. *Interpretation and the language of translation: creativity and conventions in translation*. Janus, Edinburgh.

Mona Baker. 1993. Corpus linguistics and translation studies: Implications and applications. In Mona Baker, Gill Francis, and Elena Tognini-Bonelli, editors, *Text and technology: in honour of John Sinclair*, John Benjamins, Amsterdam, pages 233–252.

Marco Baroni and Silvia Bernardini. 2006. A new approach to the study of Translationese: Machine-learning the difference between original and translated text. *Literary and Linguistic Computing* 21(3):259–274.

Viktor Becher. 2010. Abandoning the notion of "translation-inherent" explicitation: Against a dogma of translation studies. *Across Languages and Cultures* 11(1):1–28.

Shane Bergsma, Matt Post, and David Yarowsky. 2012. Stylometric analysis of scientific articles. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, pages 327–337.

Yevgeni Berzak, Roi Reichart, and Boris Katz. 2015. Contrastive analysis with predictive power: Typology driven estimation of grammatical error distributions in ESL. In *Proceedings of the 19th Conference on Computational Natural Language Learning*. pages 94–102.

David Birdsong. 1992. Ultimate attainment in second language acquisition. *Language* 68(4):706–755.

Daniel Blanchard, Joel Tetreault, Derrick Higgins, Aoife Cahill, and Martin Chodorow. 2013. TOEFL11: A corpus of non-native english. *ETS Research Report Series* 2013(2):i–15.

Shoshana Blum-Kulka. 1986. Shifts of cohesion and coherence in translation. In Juliane House and Shoshana Blum-Kulka, editors, *Interlingual and intercultural communication Discourse and cognition in translation and second language acquisition studies*, Gunter Narr Verlag, volume 35, pages 17–35.

Martin Chodorow, Michael Gamon, and Joel Tetreault. 2010. The utility of article and preposition error correction systems for English language learners: Feedback and assessment. *Language Testing* 27(3):419–436.

Rene Coppieters. 1987. Competence differences between native and near-native speakers. *Language* 63(3):544–573.

Heidi C. Dulay and Marina K. Burt. 1974. Natural sequences in child second language acquisition. *Language learning* 24(1):37–53.

Rod Ellis. 1985. *Understanding Second Language Acquisition*. Oxford Applied Linguistics. Oxford University Press.

Britt Erman, Annika Denke, Lars Fant, and Fanny Forsberg Lundell. 2014. Nativelike expression in the speech of long-residency L2 users: A study of multiword structures in L2 English, French and Spanish. *International Journal of Applied Linguistics* .

Dominique Estival, Tanja Gaustad, Son Bao Pham, Will Radford, and Ben Hutchinson. 2007. Author profiling for English emails. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*. pages 263–272.

Federico Gaspari and Silvia Bernardini. 2008. Comparing non-native and translated language: Monolingual comparable corpora with a twist. In *Proceedings of The International Symposium*

on Using Corpora in Contrastive and Translation Studies.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale L2 databases: The EF-Cambridge open language database (EFCAMDAT). In *Proceedings of the 31st Second Language Research Forum*. Cascadilla Proceedings Project, Somerville, MA.

Martin Gellerstam. 1986. Translationese in Swedish novels translated from English. In Lars Wollin and Hans Lindquist, editors, *Translation Studies in Scandinavia*, CWK Gleerup, Lund, pages 88–95.

Howard Giles and Jane L. Byrne. 1982. An intergroup approach to second language acquisition. *Journal of Multilingual and Multicultural Development* 3(1):17–40.

Sylviane Granger. 2003. The international corpus of learner English: a new resource for foreign language learning and teaching and second language acquisition research. *Tesol Quarterly* pages 538–546.

Jack Grieve. 2007. Quantitative authorship attribution: An evaluation of techniques. *Literary and Linguistic Computing* 22(3):251–270.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explorations* 11(1):10–18.

Sandra Halverson. 2003. The cognitive basis of translation universals. *Target* 15(2):197–241.

Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 187–197.

Eli Hinkel. 2001. Matters of cohesion in L2 academic texts. *Applied Language Learning* 12(2):111–132.

Juliane House. 2008. Beyond intervention: Universals in translation? *trans-kom* 1(1):6–19.

Iustina Ilisei and Diana Inkpen. 2011. Translationese traits in Romanian newspapers: A machine learning approach. *International Journal of Computational Linguistics and Applications* 2(1-2).

Scott Jarvis and Aneta Pavlenko. 2008. *Crosslinguistic influence in language and cognition*. Routledge.

Frederick Jelinek, Robert L. Mercer, Lalit R. Bahl, and J. K. Baker. 1977. Perplexity—a measure of the difficulty of speech recognition tasks. *Journal of the Acoustical Society of America* 62:S63. Supplement 1.

Jacqueline S. Johnson and Elissa L. Newport. 1991. Critical period effects on universal properties of language: The status of subjacency in the acquisition of a second language. *Cognition* 39(3):215–258.

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. 2001. Improvements to platt's smo algorithm for svm classifier design. *Neural Computation* 13(3):637–649.

Eric Kellerman and Michael Sharwood-Smith. 1986. *Crosslinguistic Influence in Second Language Acquisition*. Language Teaching Methodology Series. Pearson College Division.

Dorothy Kenny. 2001. *Lexis and creativity in translation: a corpus-based study*. St. Jerome.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. MT Summit.

Moshe Koppel and Noam Ordan. 2011. Translationese and its dialects. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 1318–1326.

Moshe Koppel, Jonathan Schler, and Kfir Zigdon. 2005. Determining an author's native language by mining a text for errors. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, pages 624–628.

David Kurokawa, Cyril Goutte, and Pierre Isabelle. 2009. Automatic detection of translated text and its impact on machine translation. In *Proceedings of MT-Summit XII*. pages 81–88.

Donna Lardiere. 2006. *Ultimate Attainment in Second Language Acquisition: A Case Study*. L. Erlbaum.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2012. Language models for machine translation: Original vs. translated texts. *Computational Linguistics* 38(4):799–825.

Gennadi Lembersky, Noam Ordan, and Shuly Wintner. 2013. Improving statistical machine translation by adapting translation models to translationese. *Computational Linguistics* 39(4):999–1023.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, Baltimore, Maryland, pages 55–60.

A. Mauranen and P. Kujamäki, editors. 2004. *Translation universals: Do they exist?*. John Benjamins.

Frederick Mosteller and David L Wallace. 1963. Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers. *Journal of the American Statistical Association* 58(302):275–309.

Sergiu Nisioi. 2015a. Feature analysis for native language identification. In Alexander F. Gelbukh, editor, *Proceedings of the 16th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing 2015)*. Springer, Lecture Notes in Computer Science.

Sergiu Nisioi. 2015b. Unsupervised classification of translated texts. In Chris Biemann, Siegfried Handschuh, André Freitas, Farid Meziane, and Elisabeth Métais, editors, *Natural Language Processing and Information Systems: Proceedings of the 20th International Conference on Applications of Natural Language to Information Systems, NLDB*. Springer, volume 9103 of *Lecture Notes in Computer Science*, pages 323–334.

Sergiu Nisioi, Ella Rabinovich, Liviu P. Dinu, and Shuly Wintner. 2016. A corpus of native, non-native and translated texts. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation, LREC 2016*.

Terence Odlin. 1989. *Language Transfer: Cross-Linguistic Influence in Language Learning*. Cambridge Applied Linguistics. Cambridge University Press.

Maeve Olohan. 2002. Leave it out! using a comparable corpus to investigate aspects of explicitation in translation. *Cadernos de Tradução* 1(9):153–169.

Maeve Olohan. 2003. How frequent are the contractions? A study of contracted forms in the translational English corpus. *Target* 15(1):59–89.

Lin Øverås. 1998. In search of the third code: An investigation of norms in literary translation. *Meta* 43(4):557–570.

Martin Potthast, Alberto Barrón-Cedeño, Benno Stein, and Paolo Rosso. 2011. Cross-language plagiarism detection. *Language Resources and Evaluation* 45(1):45–62.

Ella Rabinovich and Shuly Wintner. 2015. Unsupervised identification of translationese. *Transactions of the Association for Computational Linguistics* 3:419–432.

Philip Resnik and Noah A. Smith. 2003. The web as a parallel corpus. *Computational Linguistics* 29(3):349–380.

Larry Selinker. 1972. Interlanguage. *International Review of Applied Linguistics in Language Teaching* 10(1–4):209–232.

Miriam Shlesinger. 1989. *Simultaneous Interpretation as a Factor in Effecting Shifts in the Position of Texts on the Oral-literate Continuum*. Master's thesis, Tel Aviv University, Faculty of the Humanities, Department of Poetics and Comparative Literature.

Miriam Shlesinger. 2003. Effects of presentation rate on working memory in simultaneous interpreting. *The Interpreters Newsletter* 12:37–49.

Bernard Smith and Michael Swan. 2001. *Learner English: A teacher's guide to interference and other problems*. Ernst Klett Sprachen.

Joel Tetreault, Daniel Blanchard, and Aoife Cahill. 2013. A report on the first native language identification shared task. In *Proceedings of the Eighth Workshop on Building Educational Applications Using NLP*. Association for Computational Linguistics.

Laura Mayfield Tomokiyo and Rosie Jones. 2001. You're not from'round here, are you?: naive Bayes detection of non-native utterance text. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*. Association for Computational Linguistics, pages 1–8.

Gideon Toury. 1979. Interlanguage and its manifestations in translation. *Meta* 24(2):223–231.

Gideon Toury. 1980. *In Search of a Theory of Translation*. The Porter Institute for Poetics and Semiotics, Tel Aviv University, Tel Aviv.

Gideon Toury. 1995. *Descriptive Translation Studies and beyond*. John Benjamins, Amsterdam / Philadelphia.

Yulia Tsvetkov, Naama Twitto, Nathan Schneider, Noam Ordan, Manaal Faruqui, Victor Chahuneau, Shuly Wintner, and Chris Dyer. 2013. Identifying the L1 of non-native writers: the CMU-Haifa system. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics, pages 279–287.

Hans van Halteren. 2008. Source language markers in EUROPARL translations. In Donia Scott and Hans Uszkoreit, editors, *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*. pages 937–944.

Ria Vanderauwera. 1985. *Dutch Novels Translated into English: The transformation of a "minority" literature*. Rodopi.

Vered Volansky, Noam Ordan, and Shuly Wintner. 2015. On the features of translationese. *Digital Scholarship in the Humanities* 30(1):98–118.

Appendix A - Distribution of L1s in Translations and Non-native Texts

We assume that native languages of non-native speakers are highly correlated with (although not strictly identical to) their country of origin.

country of origin	tokens(T)	tokens(NN)
Austria	-	2K
Belgium	-	67K
Bulgaria	25K	6K
Cyprus	-	35K
Czech Republic	21K	3K
Denmark	444K	14K
Estonia	32K	50K
Finland	500K	81K
France	3,486K	28K
Germany	3,768K	17K
Greece	944K	13K
Hungary	167K	38K
Italy	1,690K	15K
Latvia	38K	13K
Lithuania	177K	18K
Luxembourg	-	46K
Malta	28K	40K
Netherlands	1,746K	64K
Poland	522K	36K
Portugal	1,633K	54K
Romania	244K	29K
Slovakia	88K	6K
Slovenia	43K	1K
Spain	1,836K	54K
Sweden	951K	52K

Table 5: Distribution of L1s by country.

Learning Text Pair Similarity with Context-sensitive Autoencoders

Hadi Amiri[1], **Philip Resnik**[1], **Jordan Boyd-Graber**[2], **Hal Daumé III**[1]
[1]Institute for Advanced Computer Studies
University of Maryland, College Park, MD
[2]Department of Computer Science
University of Colorado, Boulder, CO
{hadi,resnik,hal}@umd.edu,jordan.boyd.graber@colorado.edu

Abstract

We present a *pairwise context-sensitive Autoencoder* for computing text pair similarity. Our model encodes input text into *context-sensitive* representations and uses them to compute similarity between text pairs. Our model outperforms the state-of-the-art models in two semantic retrieval tasks and a contextual word similarity task. For retrieval, our unsupervised approach that merely ranks inputs with respect to the cosine similarity between their hidden representations shows comparable performance with the state-of-the-art supervised models and in some cases outperforms them.

1 Introduction

Representation learning algorithms learn representations that reveal intrinsic low-dimensional structure in data (Bengio et al., 2013). Such representations can be used to induce similarity between textual contents by computing similarity between their respective vectors (Huang et al., 2012; Silberer and Lapata, 2014).

Recent research has made substantial progress on semantic similarity using neural networks (Rothe and Schütze, 2015; Dos Santos et al., 2015; Severyn and Moschitti, 2015). In this work, we focus our attention on deep autoencoders and extend these models to integrate sentential or document *context* information about their inputs. We represent context information as low dimensional vectors that will be injected to deep autoencoders. To the best of our knowledge, this is the first work that enables integrating context into autoencoders.

In representation learning, context may appear in various forms. For example, the context of a current sentence in a document could be either its neighboring sentences (Lin et al., 2015; Wang and Cho, 2015), topics associated with the sentence (Mikolov and Zweig, 2012; Le and Mikolov, 2014), the document that contains the sentence (Huang et al., 2012), as well as their combinations (Ji et al., 2016). It is important to integrate context into neural networks because these models are often trained with only local information about their individual inputs. For example, recurrent and recursive neural networks only use local information about previously seen words in a sentence to predict the next word or composition.[1] On the other hand, context information (such as topical information) often capture global information that can guide neural networks to generate more accurate representations.

We investigate the utility of context information in three semantic similarity tasks: *contextual word sense similarity* in which we aim to predict semantic similarity between given word pairs in their sentential context (Huang et al., 2012; Rothe and Schütze, 2015), *question ranking* in which we aim to retrieve semantically equivalent questions with respect to a given test question (Dos Santos et al., 2015), and *answer ranking* in which we aim to rank single-sentence answers with respect to a given question (Severyn and Moschitti, 2015).

The contributions of this paper are as follows: (1) integrating context information into deep autoencoders and (2) showing that such integration improves the representation performance of deep autoencoders across several different semantic similarity tasks.

Our model outperforms the state-of-the-art su-

[1]For example, RNNs can predict the word "sky" given the sentence "clouds are in the __," but they are less accurate when longer history or global context is required, e.g. predicting the word "french" given the paragraph "I grew up in France. ... I speak fluent __."

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1882–1892,
Berlin, Germany, August 7-12, 2016.

pervised baselines in three semantic similarity tasks. Furthermore, the unsupervised version of our autoencoder show comparable performance with the supervised baseline models and in some cases outperforms them.

2 Context-sensitive Autoencoders

2.1 Basic Autoencoders

We first provide a brief description of basic autoencoders and extend them to context-sensitive ones in the next Section. Autoencoders are trained using a local unsupervised criterion (Vincent et al., 2010; Hinton and Salakhutdinov, 2006; Vincent et al., 2008). Specifically, the basic autoencoder in Figure 1(a) locally optimizes the hidden representation $\mathbf{h}$ of its input $\mathbf{x}$ such that $\mathbf{h}$ can be used to accurately reconstruct $\mathbf{x}$,

$$\mathbf{h} = \mathbf{g}(\mathbf{W}\mathbf{x} + \mathbf{b}_{\mathbf{h}}) \quad (1)$$

$$\hat{\mathbf{x}} = \mathbf{g}(\mathbf{W}'\mathbf{h} + \mathbf{b}_{\hat{\mathbf{x}}}), \quad (2)$$

where $\hat{\mathbf{x}}$ is the reconstruction of $\mathbf{x}$, the learning parameters $\mathbf{W} \in \mathbb{R}^{d' \times d}$ and $\mathbf{W}' \in \mathbb{R}^{d \times d'}$ are weight matrices, $\mathbf{b}_{\mathbf{h}} \in \mathbb{R}^{d'}$ and $\mathbf{b}_{\hat{\mathbf{x}}} \in \mathbb{R}^{d}$ are bias vectors for the hidden and output layers respectively, and $\mathbf{g}$ is a nonlinear function such as $\tanh(.)$.[2] Equation (1) *encodes* the input into an intermediate representation and Equation (2) *decodes* the resulting representation.

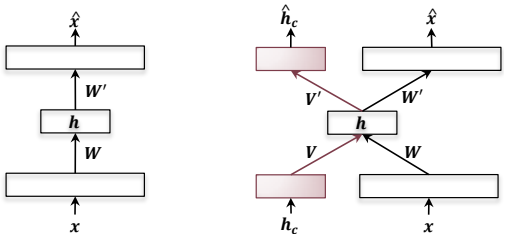
Training a single-layer autoencoder corresponds to optimizing the learning parameters to minimize the overall loss between inputs and their reconstructions. For real-valued $\mathbf{x}$, squared loss is often used, $l(\mathbf{x}) = ||\mathbf{x} - \hat{\mathbf{x}}||^2$, (Vincent et al., 2010):

$$\min_{\mathbf{\Theta}} \sum_{i=1}^{n} l(\mathbf{x}^{(i)}) \qquad \mathbf{\Theta} = \{\mathbf{W}, \mathbf{W}', \mathbf{b}_{\mathbf{h}}, \mathbf{b}_{\hat{\mathbf{x}}}\}. \quad (3)$$

This can be achieved using mini-batch stochastic gradient descent (Zeiler, 2012).

2.2 Integrating Context into Autoencoders

We extend the above basic autoencoder to integrate context information about inputs. We assume that—for each training example $\mathbf{x} \in \mathbb{R}^d$—we have a context vector $\mathbf{c}_{\mathbf{x}} \in \mathbb{R}^k$ that contains contextual information about the input.[3] The nature of this context vector depends on the input and target task. For example, neighboring words can be considered as the context of a target word in contextual word similarity task.



(a) Basic Autoencoder (b) Context Autoencoder

Figure 1: Schematic representation of basic and context-sensitive autoencoders: (a) Basic autoencoder maps its input $\mathbf{x}$ into the representation $\mathbf{h}$ such that it can reconstruct $\mathbf{x}$ with minimum loss, and (b) Context-sensitive autoencoder maps its inputs $\mathbf{x}$ and $\mathbf{h}_c$ into a context-sensitive representation $\mathbf{h}$ ($\mathbf{h}_c$ is the representation of the context information associated to $\mathbf{x}$).

We first learn the hidden representation $\mathbf{h}_c \in \mathbb{R}^{d'}$ for the given context vector $\mathbf{c}_{\mathbf{x}}$. For this, we use the same process as discussed above for the basic autoencoder where we use $\mathbf{c}_{\mathbf{x}}$ as the input in Equations (1) and (2) to obtain $\mathbf{h}_c$. We then use $\mathbf{h}_c$ to develop our context-sensitive autoencoder as depicted in Figure 1(b). This autoencoder maps its inputs $\mathbf{x}$ and $\mathbf{h}_c$ into a *context-sensitive* representation $\mathbf{h}$ as follows:

$$\mathbf{h} = \mathbf{g}(\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{h}_c + \mathbf{b}_{\mathbf{h}}) \quad (4)$$

$$\hat{\mathbf{x}} = \mathbf{g}(\mathbf{W}'\mathbf{h} + \mathbf{b}_{\hat{\mathbf{x}}}) \quad (5)$$

$$\hat{\mathbf{h}}_c = \mathbf{g}(\mathbf{V}'\mathbf{h} + \mathbf{b}_{\hat{\mathbf{h}}_c}). \quad (6)$$

Our intuition is that if $\mathbf{h}$ leads to a good reconstruction of its inputs, it has retained information available in the input. Therefore, it is a context-sensitive representation.

The loss function must then compute the loss between the input pair ($\mathbf{x}$, $\mathbf{h}_c$) and its reconstruction ($\hat{\mathbf{x}}$, $\hat{\mathbf{h}}_c$). For optimization, we can still use squared loss with a different set of parameters to minimize the overall loss on the training examples:

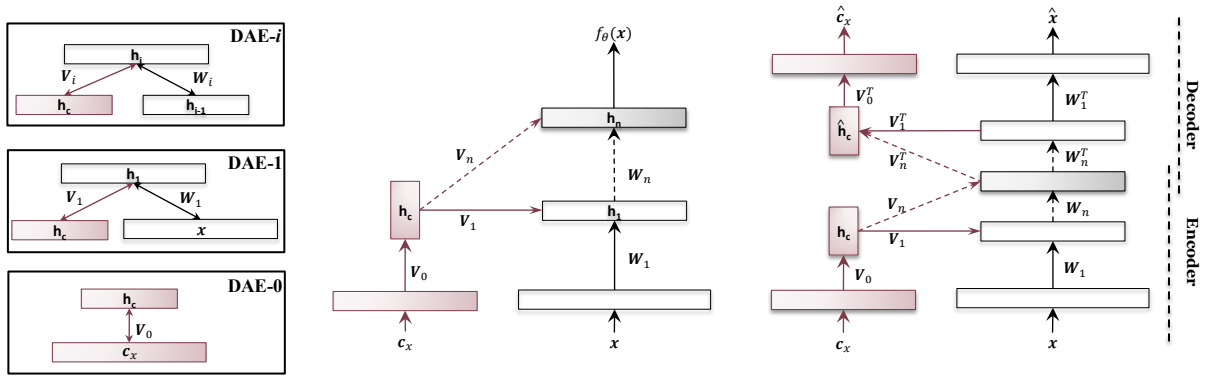
$$l(\mathbf{x}, \mathbf{h}_c) = ||\mathbf{x} - \hat{\mathbf{x}}||^2 + \lambda||\mathbf{h}_c - \hat{\mathbf{h}}_c||^2$$

$$\min_{\mathbf{\Theta}} \sum_{i=1}^{n} l(\mathbf{x}^{(i)}, \mathbf{h}_c^{(i)}) \quad (7)$$

$$\mathbf{\Theta} = \{\mathbf{W}, \mathbf{W}', \mathbf{V}, \mathbf{V}', \mathbf{b}_{\mathbf{h}}, \mathbf{b}_{\hat{\mathbf{x}}}, \mathbf{b}_{\hat{\mathbf{h}}_c}\},$$

[2] If the squared loss is used for optimization, as in Equation (3), nonlinearity is often not used in Equation (2) (Vincent et al., 2010).

[3] We slightly abuse the notation throughout this paper by referring to $\mathbf{c}_x$ or $\mathbf{h}_i$ as vectors, not elements of vectors.



(a) Network initialization (b) Stacking Denoising Autoencoders (c) Unrolling and Fine-tuning

Figure 2: Proposed framework for integrating context into deep autoencoders. Context layer ($\mathbf{c_x}$ and $\mathbf{h}_c$) and context-sensitive representation of input ($\mathbf{h}_n$) are shown in light red and gray respectively. (a) Pre-training properly initializes a stack of context-sensitive denoising autoencoders (DAE), (b) A context-sensitive deep autoencoder is created from properly initialized DAEs, (c) The network in (b) is unrolled and its parameters are fine-tuned for optimal reconstruction.

where $\lambda \in [0, 1]$ is a weight parameter that controls the effect of context information in the reconstruction process.

2.2.1 Denoising

Denoising autoencoders (DAEs) reconstruct an input from a *corrupted* version of it for more effective learning (Vincent et al., 2010). The corrupted input is then mapped to a hidden representation from which we obtain the reconstruction. However, the reconstruction loss is still computed with respect to the *uncorrupted* version of the input as before. Denoising autoencoders effectively learn representations by reversing the effect of the corruption process. We use *masking noise* to corrupt the inputs where a fraction η of input units are randomly selected and set to zero (Vincent et al., 2008).

2.2.2 Deep Context-Sensitive Autoencoders

Autoencoders can be stacked to create deep networks. A deep autoencoder is composed of multiple hidden layers that are stacked together. The initial weights in such networks need to be properly initialized through a greedy layer-wise training approach. Random initialization does not work because deep autoencoders converge to poor local minima with large initial weights and result in tiny gradients in the early layers with small initial weights (Hinton and Salakhutdinov, 2006).

Our deep context-sensitive autoencoder is composed of a stacked set of DAEs. As discussed above, we first need to properly initialize the learning parameters (weights and biases) associated to each DAE. As shown in Figure 2(a), we first train DAE-0, which initializes parameters associated to the context layer. The training procedure is exactly the same as training a basic autoencoder (Section 2.1 and Figure 1(a)).[4] We then treat $\mathbf{h}_c$ and $\mathbf{x}$ as "inputs" for DAE-1 and use the same approach as in training a context-sensitive autoencoder to initialize the parameters of DAE-1 (Section 2.2 and Figure 1(b)). Similarly, the i^{th} DAE is built on the output of the $(i-1)^{\text{th}}$ DAE and so on until the desired number of layers (e.g. n layers) are initialized. For denoising, the corruption is only applied on "inputs" of individual autoencoders. For example, when we are training DAE-i, $\mathbf{h}_{i-1}$ and $\mathbf{h}_c$ are first obtained from the original inputs of the network ($\mathbf{x}$ and $\mathbf{c_x}$) through a single forward pass and then their corrupted versions are computed to train DAE-i.

Figure 2(b) shows that the n properly initialized DAEs can be stacked to form a deep context-sensitive autoencoder. We *unroll* this network to fully optimize its weights through gradient descent and backpropagation (Vincent et al., 2010; Hinton and Salakhutdinov, 2006) .

2.2.3 Unrolling and Fine-tuning

We optimize the learning parameters of our initialized context-sensitive deep autoencoder by unfolding its n layers and making a $2n-1$ layer net-

[4] Figure 2(a) shows compact schematic diagrams of autoencoders used in Figures 1(a) and 1(b)

work whose lower layers form an "encoder" network and whose upper layers form a "decoder" network (Figure 2(c)). A global fine-tuning stage backpropagates through the entire network to fine-tune the weights for optimal reconstruction. In this stage, we update the network parameters again by training the network to minimize the loss between original inputs and their actual reconstruction. We backpropagate the error derivatives first through the decoder network and then through the encoder network. Each decoder layer tries to recover the input of its corresponding encoder layer. As such, the weights are initially symmetric and the decoder weights do need to be learned.

After the training is complete, the hidden layer $\mathbf{h}_n$ contains a context-sensitive representation of the inputs $\mathbf{x}$ and $\mathbf{c_x}$.

2.3 Context Information

Context is task and data dependent. For example, a sentence or document that contains a target word forms the word's context.

When context information is not readily available, we use topic models to determine such context for individual inputs (Blei et al., 2003; Stevens et al., 2012). In particular, we use Non-Negative Matrix Factorization (NMF) (Lin, 2007): Given a training set with n instances, i.e., $\mathbf{X} \in \mathbb{R}^{v \times n}$, where v is the size of a global vocabulary and the scalar k is the number of topics in the dataset, we learn the topic matrix $\mathbf{D} \in \mathbb{R}^{v \times k}$ and context matrix $\mathbf{C} \in \mathbb{R}^{k \times n}$ using the following sparse coding algorithm:

$$\begin{aligned} \min_{\mathbf{D},\mathbf{C}} \quad & \|\mathbf{X} - \mathbf{DC}\|_F^2 + \mu\|\mathbf{C}\|_1, \\ s.t. \quad & \mathbf{D} \geq 0,\ \mathbf{C} \geq 0, \end{aligned} \tag{8}$$

where each column in $\mathbf{C}$ is a sparse representation of an input over all topics and will be used as global context information in our model. We obtain context vectors for test instances by transforming them according to the fitted NMF model on training data. We also note that advanced topic modeling approaches, such as syntactic topic models (Boyd-Graber and Blei, 2009), can be more effective here as they generate linguistically rich context information.

3 Text Pair Similarity

We present unsupervised and supervised approaches for predicting semantic similarity scores

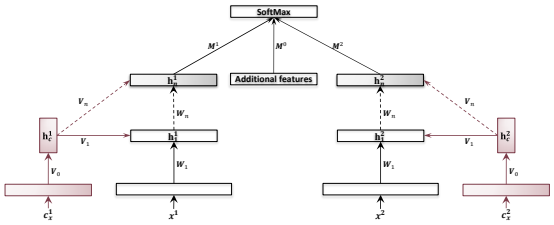


Figure 3: Pairwise context-sensitive autoencoder for computing text pair similarity.

for input texts (e.g., a pair of words) each with its corresponding context information. These scores will then be used to rank "documents" against "queries" (in retrieval tasks) or evaluate how predictions of a model correlate with human judgments (in contextual word sense similarity task).

In unsupervised settings, given a pair of input texts with their corresponding context vectors, ($\mathbf{x}^1$,$\mathbf{c}_{\mathbf{x}^1}$) and ($\mathbf{x}^2$,$\mathbf{c}_{\mathbf{x}^2}$), we determine their semantic similarity score by computing the cosine similarity between their hidden representations $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$ respectively.

In supervised settings, we use a copy of our context-sensitive autoencoder to make a pairwise architecture as depicted in Figure 3. Given ($\mathbf{x}^1$,$\mathbf{c}_{\mathbf{x}^1}$), ($\mathbf{x}^2$,$\mathbf{c}_{\mathbf{x}^2}$), and their binary relevance score, we use $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$ as well as additional features (see below) to train our pairwise network (i.e. further fine-tune the weights) to predict a similarity score for the input pair as follows:

$$rel(\mathbf{x}^1, \mathbf{x}^2) = softmax(\mathbf{M}^0\mathbf{a} + \mathbf{M}^1\mathbf{h}_n^1 + \mathbf{M}^2\mathbf{h}_n^2 + \mathbf{b}) \tag{9}$$

where $\mathbf{a}$ carries additional features, $\mathbf{M}$s are weight matrices, and $\mathbf{b}$ is the bias. We use the difference and similarity between the context-sensitive representations of inputs, $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$, as additional features:

$$\begin{aligned} \mathbf{h}_{sub} &= |\mathbf{h}_n^1 - \mathbf{h}_n^2| \\ \mathbf{h}_{dot} &= \mathbf{h}_n^1 \odot \mathbf{h}_n^2, \end{aligned} \tag{10}$$

where $\mathbf{h}_{sub}$ and $\mathbf{h}_{dot}$ capture the element-wise difference and similarity (in terms of the sign of elements in each dimension) between $\mathbf{h}_n^1$ and $\mathbf{h}_n^2$, respectively. We expect elements in $\mathbf{h}_{sub}$ to be small for semantically similar and relevant inputs and large otherwise. Similarly, we expect elements in $\mathbf{h}_{dot}$ to be positive for relevant inputs and negative otherwise.

We can use any task-specific feature as additional features. This includes features from the

minimal edit sequences between parse trees of the input pairs (Heilman and Smith, 2010; Yao et al., 2013), lexical semantic features extracted from resources such as WordNet (Yih et al., 2013), or other features such as word overlap features (Severyn and Moschitti, 2015; Severyn and Moschitti, 2013). We can also use additional features (Equation 10), computed for BOW representations of the inputs $\mathbf{x}^1$ and $\mathbf{x}^2$. Such additional features improve the performance of our and baseline models.

4 Experiments

In this Section, we use t-test for significant testing and asterisk mark (*) to indicate significance at $\alpha = 0.05$.

4.1 Data and Context Information

We use three datasets: "SCWS" a word similarity dataset with ground-truth labels on similarity of pairs of target words in sentential context from Huang et al. (2012); "qAns" a TREC QA dataset with ground-truth labels for semantically relevant questions and (single-sentence) answers from Wang et al. (2007); and "qSim" a community QA dataset crawled from Stack Exchange with ground-truth labels for semantically equivalent questions from Dos Santos et al. (2015). Table 1 shows statistics of these datasets. To enable direct comparison with previous work, we use the same training, development, and test data provided by Dos Santos et al. (2015) and Wang et al. (2007) for qSim and qAns respectively and the entire data of SCWS (in unsupervised setting).

We consider local and global context for target words in SCWS. The local context of a target word is its ten neighboring words (five before and five after) (Huang et al., 2012), and its global context is a short paragraph that contains the target word (surrounding sentences). We compute average word embeddings to create context vectors for target words.

Also, we consider question title and body and answer text as input in qSim and qAns and use NMF to create global context vectors for questions and answers (Section 2.3).

4.2 Parameter Setting

We use pre-trained word vectors from GloVe (Pennington et al., 2014). However, because qSim questions are about specific technical topics, we only use GloVe as initialization.

Data	Split	#Pairs	%Rel
SCWS	All data	2003	100.0%
qAns	Train-All	53K	12.00%
	Train	4,718	7.400%
	Dev	1,148	19.30%
	Test	1,517	18.70%
qSim	Train	205K	0.048%
	Dev	43M	0.001%
	Test	82M	0.001%

Table 1: Data statistics. (#Pairs: number of word-word pairs in SCWS, question-answer pairs in qAns, and question-question pairs in qSim; %Rel: percentage of positive pairs.)

For the unsupervised SCWS task, following Huang et al. (2012), we use 100-dimensional word embeddings, $d = 100$, with hidden layers and context vectors of the same size, $d' = 100$, $k = 100$. In this unsupervised setting, we set the weight parameter $\lambda = .5$, masking noise $\eta = 0$, depth of our model $n = 3$. Tuning these parameters will further improve the performance of our model.

For qSim and qAns, we use 300-dimensional word embeddings, $d = 300$, with hidden layers of size $d' = 200$. We set the size of context vectors k (number of topics) using the reconstruction error of NMF on training data for different values of k. This leads to $k = 200$ for qAns and $k = 300$ for qSim. We tune the other hyper-parameters (η, n, and λ) using development data.

We set each input $\mathbf{x}$ (target words in SCWS, question titles and bodies in qSim, and question titles and single-sentence answers in qAns) to the average of word embeddings in the input. Input vectors could be initialized through more accurate approaches (Mikolov et al., 2013b; Li and Hovy, 2014); however, averaging leads to reasonable representations and is often used to initialize neural networks (Clinchant and Perronnin, 2013; Iyyer et al., 2015).

4.3 Contextual Word Similarity

We first consider the contextual word similarity task in which a model should predict the semantic similarity between words in their sentential context. For this evaluation, we compute Spearman's ρ correlation (Kokoska and Zwillinger, 2000) between the "relevance scores" predicted by different models and human judgments (Section 3).

The state-of-the-art model for this task is a semi-supervised approach (Rothe and Schütze, 2015). This model use resources like WordNet

to compute embeddings for different senses of words. Given a pair of target words and their context (neighboring words and sentences), this model represents each target word as the average of its sense embeddings weighted by cosine similarity to the context. The cosine similarity between the representations of words in a pair is then used to determine their semantic similarity. Also, the Skip-gram model (Mikolov et al., 2013a) is extended in (Neelakantan et al., 2014; Chen et al., 2014) to learn contextual word pair similarity in an unsupervised way.

Table 2 shows the performance of different models on the SCWS dataset. SAE, CSAE-LC, CSAE-LGC show the performance of our pairwise autoencoders without context, with local context, and with local and global context, respectively. In case of CSAE-LGC, we concatenate local and global context to create context vectors. CSAE-LGC performs significantly better than the baselines, including the semi-supervised approach in Rothe and Schütze (2015). It is also interesting that SAE (without any context information) outperforms the pre-trained word embeddings (Pre-trained embeds.).

Comparing the performance of CSAE-LC and CSAE-LGC indicates that global context is useful for accurate prediction of semantic similarity between word pairs. We further investigate these models to understand why global context is useful. Table 3 shows an example in which global context (words in neighboring sentences) effectively help to judge the semantic similarity between "Airport" and "Airfield." This is while local context (ten neighboring words) are less effective in helping the models to relate the two words.

Furthermore, we study the effect of global context in different POS tag categories. As Figure 4 shows global context has greater impact on A–A and N–N categories. We expect high improvement in the N–N category as noun senses are fairly self-contained and often refer to concrete things. Thus broader (not only local) context is needed to judge their semantic similarity. However, we don't know the reason for improvement on the A–A category as, in context, adjective interpretation is often affected by local context (e.g., the nouns that adjectives modify). One reason for improvement could be because adjectives are often interchangeable and this characteristic makes their meaning to be less sensitive to local context.

Model	**Context**	$\rho\times$**100**
Huang et al. (2012)	LGC	65.7
Chen et al. (2014)	LGC	65.4
Neelakantan et al. (2014)	LGC	69.3
Rothe and Schütze (2015)	LGC	69.8
Pre-trained embeds. (GloVe)	-	60.2
SAE	-	61.1
CSAE	LC	66.4
CSAE	LGC	**70.9***

Table 2: Spearman's ρ correlation between model predictions and human judgments in contextual word similarity. (LC: local context only, LGC: local and global context.)

...No cases in Gibraltar were reported. The **airport** is built on the isthmus which the Spanish Government claim not to have been ceded in the Treaty of Utrecht. Thus the integration of Gibraltar Airport in the Single European Sky system has been blocked by Spain. The 1987 agreement for joint control of the airport with...
...called "Tazi" by the German pilots. On 23 Dec 1942, the Soviet 24th Tank Corps reached nearby Skassirskaya and on 24 Dec, the tanks reached Tatsinskaya. Without any soldiers to defend the **airfield** it was abandoned under heavy fire. In a little under an hour, 108 Ju-52s and 16 Ju-86s took off for Novocherkassk – leaving 72 Ju-52s and many other aircraft burning on the ground. A new base was established...

Table 3: The importance of global context (neighboring sentences) in predicting the semantically similar words (Airport, Airfield).

4.4 Answer Ranking Performance

We evaluate the performance of our model in the answer ranking task in which a model should retrieve correct answers from a set of candidates for test questions. For this evaluation, we rank answers with respect to each test question according to the "relevance score" between question and each answer (Section 3).

The state-of-the-art model for answer ranking on qAns is a pairwise convolutional neural network (PCNN) presented in (Severyn and Moschitti, 2015). PCNN is a supervised model that first maps input question-answer pairs to hidden representations through a standard convolutional neural network (CNN) and then utilizes these representations in a pairwise CNN to compute a relevance score for each pair. This model also utilizes external *word overlap* features for each question-answer pair.[5] PCNN outperforms other competing CNN models (Yu et al., 2014) and models that use

[5]Word overlap and IDF-weighted word overlap computed for (a): all words, and (b): only non-stop words for each question-answer pair (Severyn and Moschitti, 2015).

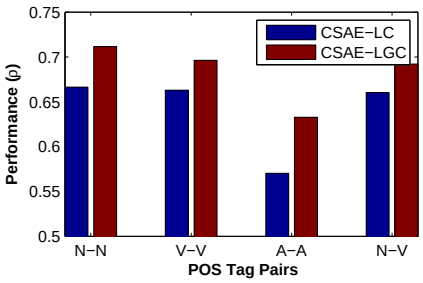


Figure 4: Effect of global context on contextual word similarity in different parts of speech (N: noun, V: verb, A: adjective). We only consider frequent categories.

Model	Train		Train-All	
	MAP	MRR	MAP	MRR
PCNN	62.58	65.91	67.09	72.80
SAE	65.69*	**71.70***	69.54*	75.47*
CSAE	**67.02***	70.99*	**72.29***	**77.29***
PCNN-WO	**73.29**	**79.62**	74.59	80.78
SAE-DST	72.53	76.97	76.38*	82.11*
CSAE-DST	71.26	76.88	**76.75***	**82.90***

Table 4: Answer ranking in *supervised* setting

Model	Train		Train-All	
	MAP	MRR	MAP	MRR
SAE	63.81	69.30	66.37	71.71
CSAE	**64.86***	**69.93***	**66.76***	**73.79***

Table 5: Answer ranking in *unsupervised* setting.

syntax and semantic features (Heilman and Smith, 2010; Yao et al., 2013).

Tables 4 and 5 show the performance of different models in terms of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR) in supervised and unsupervised settings. PCNN-WO and PCNN show the baseline performance with and without word overlap features. SAE and CSAE show the performance of our pairwise autoencoders without and with context information respectively. Their "X-DST" versions show their performance when additional features (Equation 10) are used. These features are computed for the hidden and BOW representations of question-answer pairs. We also include word overlap features as additional features.

Table 4 shows that SAE and CSAE consistently outperform PCNN, and SAE-DST and CSAE-DST outperform PCNN-WO when the models are trained on the larger training dataset, "Train-All." But PCNN shows slightly better performance than our model on "Train," the smaller training dataset. We conjecture this is because PCNN's convolution filter is wider (n-grams, $n > 2$) (Severyn and Moschitti, 2015).

Table 5 shows that the performance of *unsupervised* SAE and CSAE are comparable and in some cases better than the performance of the *supervised* PCNN model. We attribute the high performance of our models to context information that leads to richer representations of inputs.

Furthermore, comparing the performance of CSAE and SAE in both supervised and unsupervised settings in Tables 4 and 5 shows that context information consistently improves the MAP and MRR performance at all settings except for MRR on "Train" (supervised setting) that leads to a comparable performance. Context-sensitive representations significantly improve the performance of our model and often lead to higher MAP than the models that ignore context information.

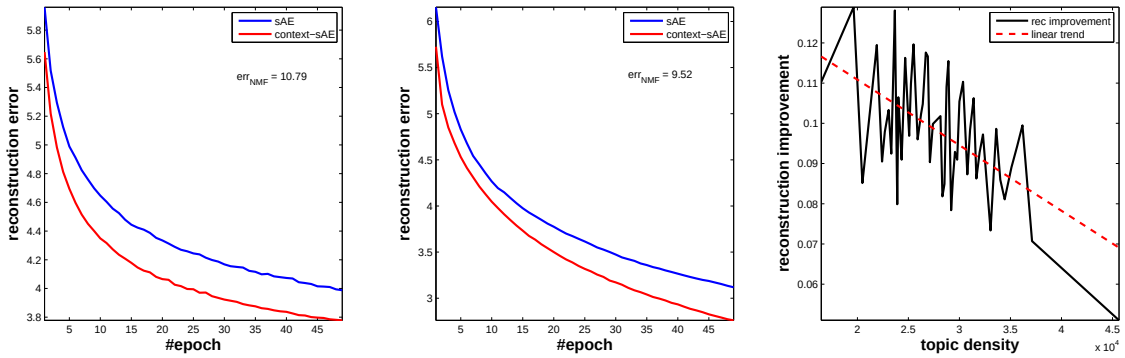
4.5 Question Ranking Performance

In the question ranking task, given a test question, a model should retrieve top-K questions that are semantically equivalent to the test question for $K = \{1, 5, 10\}$. We use qSim for this evaluation.

We compare our autoencoders against PCNN and PBOW-PCNN models presented in Dos Santos et al. (2015). PCNN is a pairwise convolutional neural network and PBOW-PCNN is a joint model that combines vector representations obtained from a pairwise bag-of-words (PBOW) network and a pairwise convolutional neural network (PCNN). Both models are supervised as they require similarity scores to train the network.

Table 6 shows the performance of different models in terms of Precision at Rank K, P@K. CSAE is more precise than the baseline; CSAE and CSAE-DST models consistently outperform the baselines on P@1, an important metric in search applications (CSAE also outperforms PCNN on P@5). Although context-sensitive models are more precise than the baselines at higher ranks, the PCNN and PBOW-PCNN models remain the best model for P@10.

Tables 6 and 7 show that context information consistently improves the results at all ranks in both supervised and unsupervised settings. The performance of the unsupervised SAE and CSAE models are comparable with the supervised PCNN model in higher ranks.



(a) qSim Dataset (b) qAns Dataset (c) qSim Dataset

Figure 5: Reconstruction Error and Improvement: (a) and (b) reconstruction error on qSim and qAns respectively. err_{NMF} shows the reconstruction error of NMF. Smaller error is better, (c) improvement in reconstruction error vs. topic density: greater improvement is obtained in topics with lower density.

Model	P@1	P@5	P@10
PCNN	20.0	33.8	**40.4**
SAE	16.8	29.4	32.8
CSAE	**21.4**	**34.9**	37.2
PBOW-PCNN	22.3	**39.7**	**46.4**
SAE-DST	22.2	35.9	42.0
CSAE-DST	**24.6**	37.9	38.9

Table 6: Question ranking in *supervised* setting

Model	P@1	P@5	P@10
SAE	17.3	32.4	32.8
CSAE	**18.6**	**33.2**	**34.1**

Table 7: Question ranking in *unsupervised* setting

5 Performance Analysis and Discussion

We investigate the effect of context information in reconstructing inputs and try to understand reasons for improvement in reconstruction error. We compute the average reconstruction error of SAE and CSAE (Equations (3) and (7)). For these experiments, we set $\lambda = 0$ in Equation (7) so that we can directly compare the resulting loss of the two models. CSAE will still use context information with $\lambda = 0$ but it *does not* backpropagate the reconstruction loss of context information.

Figures 5(a) and 5(b) show the average reconstruction error of SAE and CSAE on qSim and qAns datasets. Context information conistently improves reconstruction. The improvement is greater on qSim which contains smaller number of words per question as compared to qAns. Also, both models generate smaller reconstruction errors than NMF (Section 2.3). The lower performance of NMF is because it reconstructs inputs merely using global topics identified in datasets, while our models utilize both local and global information to reconstruct inputs.

5.1 Analysis of Context information

The improvement in reconstruction error mainly stems from areas in data where "topic density" is lower. We define topic density for a topic as the number of documents that are assigned to the topic by our topic model. We compute the average improvement in reconstruction error for each topic $\mathcal{T}_j$ using the loss functions for the basic and context-sensitive autoencoders:

$$\Delta_j = \frac{1}{|\mathcal{T}_j|} \sum_{\mathbf{x} \in \mathcal{T}_j} l(\mathbf{x}) - l(\mathbf{x}, \mathbf{h_x})$$

where we set $\lambda = 0$. Figure 5(c) shows improvement of reconstruction error versus topic density on qSim. Lower topic densities have greater improvement. This is because they have insufficient training data to train the networks. However, injecting context information improves the reconstruction power of our model by providing more information. The improvements in denser areas are smaller because neural networks can train effectively in these areas.[6]

5.2 Effect of Depth

The intuition behind deep autoencoders (and, generally, deep neural networks) is that each layer learns a more abstract representation of the input than the previous one (Hinton and Salakhutdinov, 2006; Bengio et al., 2013). We investigate

[6] We observed the same pattern in qAns.

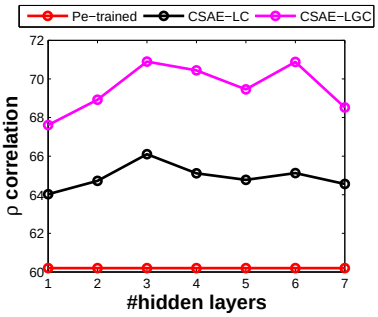


Figure 6: Effect of depth in contextual word similarity. Three hidden layers is optimal for this task.

if adding depth to our context-sensitive autoencoder will improve its performance in the contextual word similarity task.

Figure 6 shows that as we increase the depth of our autoencoders, their performances initially improve. The CSAE-LGC model that uses both local and global context benefits more from greater number of hidden layers than CSAE-LC that only uses local context. We attribute this to the use of global context in CSAE-LGC that leads to more accurate representations of words in their context. We also note that with just a single hidden layer, CSAE-LGC largely improves the performance as compared to CSAE-LC.

6 Related Work

Representation learning models have been effective in many tasks such as language modeling (Bengio et al., 2003; Mikolov et al., 2013b), topic modeling (Nguyen et al., 2015), paraphrase detection (Socher et al., 2011), and ranking tasks (Yih et al., 2013). We briefly review works that use context information for text representation.

Huang et al. (2012) presented an RNN model that uses document-level context information to construct more accurate word representations. In particular, given a sequence of words, the approach uses other words in the document as external (global) knowledge to predict the next word in the sequence. Other approaches have also modeled context at the document level (Lin et al., 2015; Wang and Cho, 2015; Ji et al., 2016).

Ji et al. (2016) presented a context-sensitive RNN-based language model that integrates representations of previous sentences into the language model of the current sentence. They showed that this approach outperforms several RNN language models on a text coherence task.

Liu et al. (2015) proposed a context-sensitive RNN model that uses Latent Dirichlet Allocation (Blei et al., 2003) to extract topic-specific word embeddings. Their best-performing model regards each topic that is associated to a word in a sentence as a pseudo word, learns topic and word embeddings, and then concatenates the embeddings to obtain topic-specific word embeddings.

Mikolov and Zweig (2012) extended a basic RNN language model (Mikolov et al., 2010) by an additional feature layer to integrate external information (such as topic information) about inputs into the model. They showed that such information improves the perplexity of language models.

In contrast to previous research, we integrate context into deep autoencoders. To the best of our knowledge, this is the first work to do so. Also, in this paper, we depart from most previous approaches by demonstrating the value of context information in *sentence-level* semantic similarity and ranking tasks such as QA ranking tasks. Our approach to the ranking problems, both for Answer Ranking and Question Ranking, is different from previous approaches in the sense that we judge the relevance between inputs based on their context information. We showed that adding sentential or document context information about questions (or answers) leads to better rankings.

7 Conclusion and Future Work

We introduce an effective approach to integrate sentential or document context into deep autoencoders and show that such integration is important in semantic similarity tasks. In the future, we aim to investigate other types of linguistic context (such as POS tag and word dependency information, word sense, and discourse relations) and develop a unified representation learning framework that integrates such linguistic context with representation learning models.

Acknowledgments

We thank anonymous reviewers for their thoughtful comments. This paper is based upon work supported, in whole or in part, with funding from the United States Government. Boyd-Graber is supported by NSF grants IIS/1320538, IIS/1409287, and NCSE/1422492. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the view of the sponsors.

References

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Yoshua Bengio, Aaron Courville, and Pierre Vincent. 2013. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8).

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

Jordan L Boyd-Graber and David M Blei. 2009. Syntactic topic models. In *Proceedings of NIPS*.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of EMNLP*.

Stéphane Clinchant and Florent Perronnin. 2013. Aggregating continuous word embeddings for information retrieval. *the Workshop on Continuous Vector Space Models and their Compositionality, ACL*.

Cicero Dos Santos, Luciano Barbosa, Dasha Bogdanova, and Bianca Zadrozny. 2015. Learning hybrid representations to retrieve semantically equivalent questions. In *Proceedings of ACL-IJCNLP*.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Proceedings of NAACL*.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2012. Improving word representations via global context and multiple word prototypes. In *Proceedings of ACL*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of ACL-IJCNLP*.

Yangfeng Ji, Trevor Cohn, Lingpeng Kong, Chris Dyer, and Jacob Eisenstein. 2016. Document context language models. *ICLR (Workshop track)*.

S. Kokoska and D. Zwillinger. 2000. *CRC Standard Probability and Statistics Tables and Formulae, Student Edition*. Taylor & Francis.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of ICML*.

Jiwei Li and Eduard Hovy. 2014. A model of coherence based on distributed sentence representation. In *Proceedings of EMNLP*.

Rui Lin, Shujie Liu, Muyun Yang, Mu Li, Ming Zhou, and Sheng Li. 2015. Hierarchical recurrent neural network for document modeling. In *Proceedings of EMNLP*.

Chuan-bi Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *Proceedings of AAAI*.

Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *Spoken Language Technologies*. IEEE.

Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Proceedings of INTERSPEECH*.

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

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *Proceedings of the EMNLP*.

Dat Quoc Nguyen, Richard Billingsley, Lan Du, and Mark Johnson. 2015. Improving topic models with latent feature word representations. *TACL*, 3:299–313.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of ACL-IJNLP*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *Proceedings of EMNLP*.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of SIGIR*.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of ACL*.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*.

Keith Stevens, Philip Kegelmeyer, David Andrzejewski, and David Buttler. 2012. Exploring topic coherence over many models and many topics. In *Proceedings of EMNLP-CNNL*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings ICML*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research*, 11:3371–3408.

Tian Wang and Kyunghyun Cho. 2015. Larger-context language modelling. *CoRR*, abs/1511.03729.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? a quasi-synchronous grammar for QA. In *Proceedings of EMNLP-CoNLL*.

Xuchen Yao, Benjamin Van Durme, Chris Callison-burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Proceedings of NAACL*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *Proceedings of ACL*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. In *NIPS, Deep Learning Workshop*.

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

Linguistic Benchmarks of Online News Article Quality

Ioannis Arapakis*
Eurecat
Barcelona, Spain
arapakis.ioannis@gmail.com

Filipa Peleja*
Vodafone
Lisbon, Portugal
filipapeleja@gmail.com

B. Barla Cambazoglu*
Independent Researcher
barla@berkantbarlacambazoglu.com

Joao Magalhaes
NOVA-LINCS, DI, FCT
Universidade NOVA Lisboa, Portugal
jmag@fct.unl.pt

Abstract

Online news editors ask themselves the same question many times: *what is missing in this news article to go online?* This is not an easy question to be answered by computational linguistic methods. In this work, we address this important question and characterise the constituents of news article editorial quality. More specifically, we identify 14 aspects related to the content of news articles. Through a correlation analysis, we quantify their independence and relation to assessing an article's editorial quality. We also demonstrate that the identified aspects, when combined together, can be used effectively in quality control methods for online news.

1 Introduction

A recent study[1] found that online news is nowadays the main source of news for the population in the 18-29 age group (71%), and as popular as TV in the 30-39 age group (63%). The readers appetite for high-quality online news result in an offer of thousands of articles published every day in the whole of the Web. For instance, it is not uncommon to find the same facts reported by many different online news articles. However, only a few of them actually grab the attention of the readers. Journalists and editors follow standardised discourse rules and techniques aiming at engaging the reader in the article's narrative of article (Louis and Nenkova, 2013).

Analysing the discourse of such articles is central to properly assessing the quality of online news (van Dijk and Kintsch, 1983). Defining the variables that computational linguistics should quantify is a challenging task. Several questions arise from this exercise. For example, what does the quality refer to? What makes a new article perceived as high quality by the editors/users? What aspects of an article correlate better with its perceived quality? Can we predict the quality of an article using linguistic features extracted from its content? These are the kind of questions we address in this paper.

To this end, we propose a linguistic resource and assessment methodology to quantify the editorial quality of online news discourse. We argue that quality is too complex to be represented by a single number and should be instead decomposed into a set of simpler variables that capture the different linguistic and narrative aspects of online news. Thus, we depart from current literature and propose a multidimensional representation of quality. The first contribution of this paper is a taxonomy of 14 different content aspects that are associated with the editor-perceived quality of online news articles. The proposed 14 aspects are the result of an editorial study involving professional editors, journalists, and computational linguists.

The second contribution of this paper is an expert-annotated corpus of online news articles obtained from a major news portal. This corpus is curated by the editors and journalists who annotated the articles with respect to the 14 aspects and to the general editorial quality. To confirm the independence and relevance of the proposed aspects, we perform a correlation analysis on this ground-truth to determine the strength of the associations between different aspects and article editorial quality. Our analysis shows that the editor-perceived quality of an article exhibits a strong positive correlation with certain aspects, such as

*This work was done while the authors were at Yahoo!Research Barcelona.

[1] http://www.people-press.org/2013/08/08/amid-criticism-support-for-medias-watchdog-role-stands-out

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1893–1902,
Berlin, Germany, August 7-12, 2016.

fluency and *completeness*, while it is weakly correlated with other aspects like *subjectivity* and *polarity*.

As a baseline benchmark, we investigate the feasibility of predicting the quality aspects of an article using features extracted from the article only. Our findings indicate that article editorial quality prediction is a challenging task and that article quality can be predicted to a varying degree, depending on the feature space. The proposed aspects can be used to control the editorial quality with a Root Mean Squared Error (RMSE) of 0.398 on a 5-point Likert-scale.

The rest of the paper is organised as follows. Next, we discuss existing literature in discourse analysis and text quality metrics. In Section 3, we present the aspects that we identified as potential indicators of article quality. Section 4 provides the details of our online news corpus targeting the aspects of editorial quality control. The results of the correlation analysis conducted between the identified aspects and article quality are presented in Section 4. In Section 5, we present a baseline benchmark to automatically infer individual aspects and editorial quality from online news.

2 Related Work

A very recent work related to ours is (Gao et al., 2014), where the authors try to predict the interestingness of a news article for a user who is currently reading another news article. In our work, however, we try to predict the perceived quality of an article without using any context information other than the content of the article itself. Moreover, while the authors of (Gao et al., 2014) take a quite pragmatic approach to handle the problem, we follow a more principled approach and model the quality of a news article according to five orthogonal dimensions: readability, informativeness, style, topic, and sentiment. Work has been done in each one of these dimensions, but none has tackled the problem of modelling overall article quality in a comprehensive and articulated manner as we do. Below, we provide a survey of the previous work on these dimensions.

The readability of a piece of text can be defined as the ease that the text can be processed and understood by a human reader (Richards and Schmidt, 2013; Zamanian and Heydari, 2012). The readability is usually associated with fluency and writing quality (Nenkova et al., 2010; Pitler and Nenkova, 2008). Even though there is a significant amount of research that targets readability, most work (Redish, 2000; Yan et al., 2006) were originally designed to measure the readability of school books and do not suit well to more complex reading materials, such as news articles, which form the focus of our work.

The informativeness of a news article has been tackled from several different angles. In (Tang et al., 2003), news information quality was characterised by a set of nine aspects that were shown to have a good correlation with textual features. Catchy titles were shown to often lead to frustration, as the reader does not get the content that she expects (Louis and Nenkova, 2011). The task of assessing a news title's descriptiveness is related to semantic text similarity and has been researched by the SemEval initiative (Agirre et al., 2013). Moreover, the completeness of a news article is an aspect that has been considered in the past by (Louis and Nenkova, 2014), which showed that reporting the news with adequate detail is key to provide the reader with enough information to grasp the entire story. The freshness of news information also sets the tone of the discourse: information can be novel to the average reader or it can be already known and be presented as a reference to the reader. The novelty of an article is essentially accomplished by either analysing previous articles (Gamon, 2006) or by relying on real-time data from social-media services (Phelan et al., 2009).

The characterisation of the style of text compositions has been an active topic of research in communication sciences and humanities. An excellent example of the research done in this area is the influential work in (McNamara et al., 2009), where the authors found the best predictors of writing quality to be the syntactic complexity (number of words before the main verb), the diversity of words used by the author, and some other shallow features. In NLP, the writing style has been investigated in several contexts. A problem relevant to the one we addressed is the characterisation of an author's writing style to predict the success of novels (Ashok et al., 2013). The authors investigated a wide range of complex linguistic features, ranging from simple unigrams to distribution of word categories, grammar rules, distribution of constituents, sentiment, and connotation. The comparison of novels and news articles revealed a great similar-

ity in the writing style of novels and informative articles.

The broadness of a news topic has an impact on the reader's perceived quality of the article. A technical article is usually targeting niche groups of users and a popular article targets the masses. One of the few corpus (Louis and Nenkova, 2013) addressing quality was limited to the domain of scientific journalism, thus more technical articles. This corpus only considered news from the New York Times, thus contained already very good quality news. Two recent work investigated the feasibility of predicting news articles' feature popularity in social media at cold start (Bandari et al., 2012; Arapakis et al., 2014a). In (Bandari et al., 2012), features extracted from the article's content as well as additional meta-data was used to predict the number of times an article will be shared in Twitter after it went online. In (Arapakis et al., 2014a), a similar study was repeated to predict the popularity of a news article in social media using additional features obtained from external sources.

Sentiment analysis concerns the subjectivity and the strength and sign of the opinions expressed in a given piece of text. In (Arapakis et al., 2014b), it was demonstrated that news articles exhibit considerable variation in terms of the sentimentality and polarity of their content. The work in (Phelan et al., 2009) has provided evidence that sentiment-related aspects are important to profile and assess the quality of news articles. Sentiment analysis has been applied to news articles in other contexts as well (Godbole et al., 2007; Balahur et al., 2010).

3 Modeling News Article Quality

The editorial control of news articles is an unsolved task that involves addressing a number of issues, such as identifying the characteristics of an effective text, determining what methods produce reliable and valid judgments for text quality, as well as selecting appropriate aspects of text evaluation that can be automated using machine learning methods. Underlying these tasks is a main theme: can we identify benchmarks for characterising news article quality? Therefore, there is a need for empirical work to identify the global and local textual features which will help us make an optimal evaluation of news articles.

By doing so, we achieve two goals. On one hand, we can offer valuable insights with respect to what constitutes an engaging, good quality news article. On the other hand, we can identify benchmarks for characterising news article quality in an automatic and scalable way and, thus, predict poor writing before a news article is even published. This can help reduce greatly the burden of manual evaluation which is currently performed by professional editors.

3.1 Methodology

The methodology described here provides a framework for characterising and modelling news article editorial quality. In our work, we follow a bottom-up approach and identify 14 different content aspects that are good predictors (as we demonstrate in Section 6.1) of news article quality. The aspects we identified are informed by input from news editors, journalists and computational linguists, and previous research in NLP and, particularly, the efforts in text summarisation (Bouayad-Agha et al., 2012), document understanding (Dang, 2005; Seki et al., 2006) and question answering (Surdeanu et al., 2008; Shtok et al., 2012).

After discussing the editorial quality control with professionals, we gathered a set of heuristics and examined the literature for ways of designing quantitative measures to achieve our goal. We group the aspects under five headings: readability, informativeness, style, topic, and sentiment (see Fig. 1). Below, we provide a brief description of each aspect.

3.2 Readability

High quality articles are written in a way that makes them easier to read. In our model, we include two different aspects related to readability (Pitler and Nenkova, 2008): fluency and conciseness.

Fluency: Fluent articles are built from sentence to sentence, forming a coherent body of information. Consecutive sentences are meaningfully connected. Similarly, paragraphs are written in a logical sequence.

Conciseness: Concise articles have a focus. Sentences contain information that is related to the main theme of the article. The same or similar information is tried to be not repeated.

3.3 Informativeness

As a main reason for reading online news is to remain well-informed (Tang et al., 2003), informativeness of articles have an effect on their per-

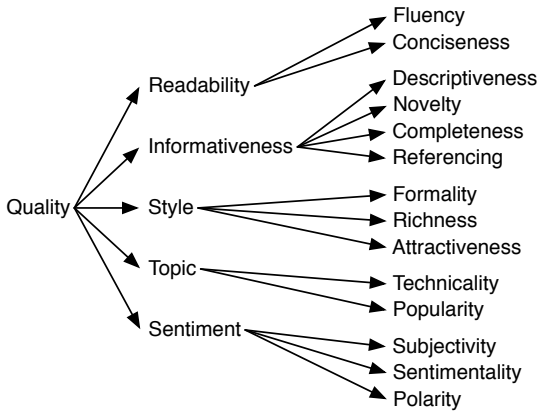


Figure 1: A taxonomy of the identified aspects.

ceived quality. In our model, we consider four different aspects related to informativeness: descriptiveness, novelty, completeness, and referencing.

Descriptiveness: Descriptiveness indicates how well the title of an article reflects its main body content. Titles with low descriptiveness are often click baits (e.g., "You won't believe what you will see"). Such titles may lead to dissatisfaction, as the provided news content usually does not meet the raised user expectation.

Novelty: Novel articles provide new and valuable information to the readers. The provided information is unlikely to be known to an average reader.

Completeness: Complete articles cover the topic in an adequate level of detail (Louis and Nenkova, 2014; Bouayad-Agha et al., 2012). A reader can satisfy her information need after reading such an article.

Referencing: Referencing is about the degree to which the article references external sources (including other people's opinions and related articles). Providing references allows the reader to access related information sources easily, (Gamon, 2006; van Dijk and Kintsch, 1983).

3.4 Style

The language and aesthetics is also related to the article quality (McNamara et al., 2009; Ashok et al., 2013; Pavlick and Tetreault, 2016; Peterson et al., 2011). We consider three style-related aspects: formality, richness, and attractiveness.

Formality: Formal articles are written by following certain writing guidelines. They are more likely to contain formal words and obey punctuation/grammar rules(Peterson et al., 2011).

Richness: The vocabulary of rich articles is perceived as diverse and interesting by the readers. Rich articles are not written in a plain and straightforward manner.

Attractiveness: Attractiveness measures the degree to which the title of an article raises curiosity in its readers. Attractive titles entice people to continue reading the main content of the article.

3.5 Topic

Editors consider the nature of the article with respect to its target audience, i.e., according to the target audience (technical or popular) the other aspects may play a different role. We investigate two topic-related aspects: technicality and popularity.

Technicality: Technical articles (Louis and Nenkova, 2013) usually require some effort to understand as well as previous knowledge on the topic. Examples of usually technical news topics include science and finance.

Popularity: The popularity refers to the size of the audience who would be interested in the topic of the article (Bandari et al., 2012; Arapakis et al., 2014b). For example, while many readers are interested in reading about celebrities, few readers are interested in articles about anthropology.

3.6 Sentiment

Finally, we consider the sentiments expressed in an article. Besides opinion articles (which are subjective by nature), many news may also convey a particular emotion. We evaluate three sentiment-related aspects: subjectivity, sentimentality, and polarity.

Subjectivity: Subjective articles tend to contain opinions, preferences, or possibilities. There are relatively few factual statements.

Sentimentality: Sentimentality is a measure of the total magnitude of positive or negative statements made in the article regarding an object or an event. Highly sentimental articles include relatively few neutral statements.

Polarity: Polarity indicates the overall sign of the sentiments expressed in the article (Arapakis et al., 2014a). Articles with positive (negative) polarity include relatively more statements with positive (negative) sentiment.

4 Corpus: Editorial Quality Control

Our goal is to identify proxies of news article quality that can be learned and predicted in an automatic and scalable manner. To identify these proxies, we rely on the domain knowledge and human intuition of expert judges, whom we employ in a rigorous, crowdsourcing-based evaluation for gen-

erating a ground-truth dataset. Through an editorial study we create an in-domain, annotated news corpus that allows us to learn predictive models which can estimate accurately the perceived quality of news articles.

4.1 Online News Articles

Our analysis was conducted on a dataset consisting of $13,319$ news articles taken from a major news portal[2]. We opted for a single news portal to be able to extract features that are consistent across all news articles. The dataset was constructed by crawling news articles over a period of two weeks. During the crawling period, we connected to the RSS news feed of the portal every 15 minutes and fetched newly published articles written in English. The content of the discovered articles was then downloaded from the portal.

Each article is identified by its unique URI and stored in a database, along with some meta-data, such as article's genre, its publication date, and its HTML content. We applied further filtering on the initial set of 13,319 news articles. The word count distribution of the articles followed a bimodal pattern, with the bulk of the articles located around a mean value of 447.5. Using this value as a reference point, we removed articles that contain less than 150 or more than 800 words. We then sampled a smaller set of articles such that each of the most frequent 15 genres have at least 65 articles in the sample. This left us with 1,043 new articles, out of which a randomly selected set of 561 articles were used in the editorial study.

The selected news articles were preprocessed before the editorial study. The preprocessing was performed in two steps. First, we removed the boilerplate of HTML pages and extracted the main body text of news articles, using Boilerpipe (Kohlschütter et al., 2010). Second, we segmented the body text into sentences and paragraphs. For sentence segmentation, we used the Stanford CoreNLP library, which includes a probabilistic parser (Klein and Manning, 2003; Mihalcea and Csomai, 2007). For each news article we generated a body- and sentence- level annotation form (see example in the supplementary notes).

4.2 Annotations of Editorial Quality Aspects

For our editorial study, we employed ten expert judges (male = 4, female = 6) who had a background in computational linguistics, journalism, or were media monitoring experts. The expert judges were either native English speakers or were proficient with the English language. The expert judges assessed a total of 561 news articles on 15 measures (14 aspects and the main quality measure), using a 5-point Likert scale, where low and high scores suggest weak or strong presence of the assessed measure, respectively.

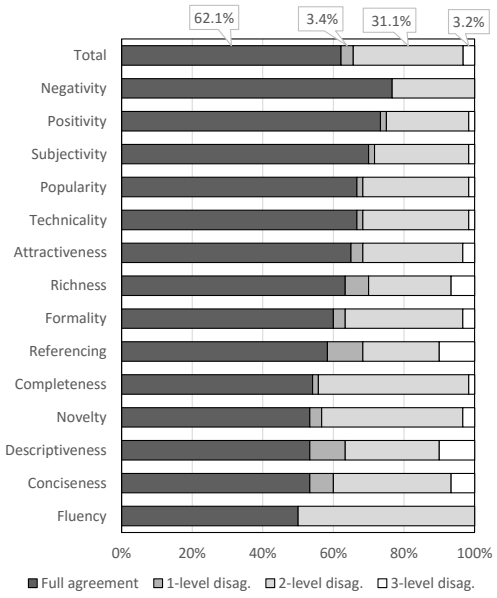


Figure 2: Annotators agreement.

The annotation took place remotely, and each expert judge could annotate up to ten news articles per day (this threshold was set to ensure a high quality of annotation), and each article was annotated by one expert judge and by one of the authors of this paper. Prior to that, there was a pilot session were each expert judge was asked to become familiar with the quality criteria and annotate three trial news articles. Next, a meeting (physical or online) was arranged and the authors discussed with the expert judge the rationale behind assigning the scores, and appropriate corrections and recommendations were made. This step ensured that we had disambiguated any questions prior to the editorial study and also assured that expert judges followed the same scoring procedure. The compensation for annotating was 10€per article. The annotated corpus is publicly available.[3]

Fig. 2 illustrates the details of the overall annotations agreement. We can see that annotations agree on 62.1% of the articles, on 65.5% they vary

[2]Yahoo! News at http://www.yahoo.com/news.

[3]`http://novasearch.org/datasets/`.

only 1-point and in 96.6% they vary 2 points in the 5-point Likert-scale. These results are quite satisfying and show a good level of agreement and consistency across all aspects.

4.3 Corpus Statistics

Table 1 shows the mean (M) and standard deviation (SD) values for five different distributions (number of characters, words, unique words, entities, and sentences) and four different subsets of the corpus. The subsets contain all articles, high-quality articles (labels 4 and 5), medium-quality articles (label 3), or low-quality articles (labels 1 and 2). The last three subsets contain 84, 298, and 179 news articles, respectively. According to these numbers, the article quality follows an unbalanced distribution: about half of the articles are labeled as medium quality, and there are about two times more low-quality articles than high-quality articles. According to Table 1, there is a clear difference between distributions for the high- and low- quality articles. In general, we observe that higher-quality articles are relatively longer (e.g., more words or sentences), on average.

5 Aspects Correlation Analysis

To identify which aspects of a news article are better discriminants of its quality, we perform a correlation analysis. Given that we are looking at ordinal data that violates parametric assumptions, we compute the Spearman's rank correlation coefficients (r_s) between the aspects' scores and the news article quality that we acquired from our ground truth. The motivation behind this analysis is to get a first intuition into the aspects' effectiveness to act as quality predictors, by understanding how they are associated to news article quality.

In Table 2, we report several statistically significant correlations between the different aspects. Given that our correlation analysis involves multiple pairwise comparisons, we need to correct the level of significance for each test such that the overall Type I error rate (α) across all comparisons remains at .05. Given that the Bonferroni correction is too conservative in the Type I error rate, we opt for the more liberal criterion proposed by Benjamini and Hochberg (Benjamini and Hochberg, 1995; Benjamini and Hochberg, 2000) and compute the critical p-value for every pairwise comparisons as

$$p_{\text{crit}} = \frac{j}{k}\alpha, \tag{1}$$

where j is the index of all pairwise comparison p-values, listed in an ascending order, and k is the number of comparisons. If we consider Cohen's conventions for the interpretation of effect size, we observe that most of the correlation coefficients shown in Table 2 represent sizeable effects, which range from small ($\pm.1$) to large ($\pm.5$). For example, *completeness* is highly correlated with quality ($r_s = .70$) while *polarity* is the least correlated with quality ($r_s = .05$). In addition, Table 2 does not provide any evidence of multicollinearity since none of the aspects (with the exception of quality) are significantly highly correlated ($r_s > .80$).

6 Predicting Editorial Quality

6.1 Predicting EQ with the Aspects

In this section, we demonstrate the predictive characteristics of the proposed aspects (Section 3) with respect to news article quality. We formulate the prediction problem as a regression problem, and conduct a 10-fold cross validation to estimate the regression model. For our regression task we use a Generalised Linear Model (`GLM`) via penalized maximum likelihood (Friedman et al., 2010). The regularisation path is computed for the lasso or elasticnet penalty at a grid of values for the regularisation parameter lambda. The `GLM` solves the following problem

$$\min_{\beta_0,\beta} \frac{1}{N}\sum_{i=1}^{N} w_i l(y_i, \beta_0+\beta^T x_i)+\lambda[(1-\alpha)\frac{\|\beta\|_2^2}{2}+\alpha\|\beta\|_1,], \tag{2}$$

over a grid of values of λ covering the entire range. Here $l(y, \eta)$ is the negative log-likelihood contribution for observation i. The elastic-net penalty is controlled by α, and bridges the gap between lasso ($\alpha = 1$, the default) and ridge ($\alpha = 0$). The tuning parameter λ controls the overall strength of the penalty. It is known that the ridge penalty shrinks the coefficients of correlated predictors towards each other while the lasso tends to pick one of them and discard the others, which makes it more robust against predictor collinearity and overfitting. We used the values that minimise RMSE, i.e., $\alpha = 0.95$ and $\lambda = 0.01$.

In Table 3, we see the coefficients of the final `GLM` model which are to be interpreted in the same manner as a Cox model. A positive regression coefficient for an explanatory variable means that the variable is associated with a higher risk of an event. In our case, all coefficients are positive, being *completeness*, *fluency* and *richness* the ones

Table 1: Statistics for the annotated news corpus (M $\pm$ SD values)

	All	High quality	Medium quality	Low quality
Characters	2490.28 $\pm$ 1900.95	4321.92 $\pm$ 2258.51	2698.94 $\pm$ 1641.85	1290.07 $\pm$ 1166.65
Words	413.03 $\pm$ 318.63	717.87 $\pm$ 386.08	447.46 $\pm$ 274.88	213.76 $\pm$ 193.12
Unique words	167.29 $\pm$ 110.25	269.08 $\pm$ 122.14	180.27 $\pm$ 95.72	98.29 $\pm$ 77.09
Entities	18.45 $\pm$ 14.43	23.85 $\pm$ 15.09	19.43 $\pm$ 11.03	14.29 $\pm$ 17.59
Sentences	20.67 $\pm$ 17.89	35.27 $\pm$ 24.92	21.76 $\pm$ 14.02	12.03 $\pm$ 14.42

Table 2: Correlations between different aspects in the ground-truth data

	Conc.	Desc.	Nov.	Comp.	Ref.	Form.	Rich.	Attr.	Tech.	Pop.	Subj.	Sent.	Pol.	Qual.
Fluency	$.61^{**}$	$.38^{**}$	$.34^{**}$	$.57^{**}$	$.37^{**}$	$.40^{**}$	$.53^{**}$	$.41^{**}$	$.15^{**}$	$.27^{**}$	$.12^{**}$	$.11^{**}$	.03	**.66****
Conciseness		$.33^{**}$	$.32^{**}$	$.38^{**}$	$.28^{**}$	$.41^{**}$	$.39^{**}$	$.30^{**}$	$.24^{**}$	$.25^{**}$	$-.00$	.08	.01	$.47^{**}$
Descriptiveness			$.18^{**}$	$.32^{**}$	$.23^{**}$	$.23^{**}$	$.19^{**}$	$.13^{**}$	$.13^{**}$	$.17^{**}$	$.00^{*}$	.09	.00	$.37^{**}$
Novelty				$.39^{**}$	$.40^{**}$	$.44^{**}$	$.35^{**}$	$.37^{**}$	$.16^{**}$	$.32^{**}$	.05	$.25^{**}$	$-.05$	$.41^{**}$
Completeness					$.48^{**}$	$.38^{**}$	$.51^{**}$	$.39^{**}$	$.30^{**}$	$.26^{**}$	$.18^{**}$	$.20^{**}$	.03	**.70****
References						$.51^{**}$	$.35^{**}$	$.33^{**}$	$.30^{**}$	$.29^{**}$	$.27^{**}$	$.44^{**}$	.00	**.52****
Formality							$.43^{**}$	$.30^{**}$	$.46^{**}$	$.25^{**}$	.01	$.31^{**}$	$-.08$	$.47^{**}$
Richness								$.50^{**}$	$.25^{**}$	$.35^{**}$	$.24^{**}$	$.15^{**}$	.04	**.63****
Attractiveness									$.15^{**}$	$.55^{**}$	$.28^{**}$	$.23^{**}$	$-.02$	**.52****
Technicality										$.22^{**}$	$.16^{**}$	$.22^{**}$	$.11^{*}$	$.30^{**}$
Popularity											$.28^{**}$	$.24^{**}$	.02	$.41^{**}$
Subjectiveness												$.42^{**}$	$.11^{*}$	$.23^{**}$
Sentimemtality													$-.13^{*}$	$.27^{**}$
Polarity														.05

Significance levels (two-tailed) are as follows: $^{*} :< .01$; $^{**} :< .001$.

Table 3: The coefficients of the final GLM model. The intercept value is 2.9103.

Group	Aspects	Coefficients
Readability	Fluency	.1730
	Conciseness	.0372
Informativeness	Completeness	.2062
	Descriptiveness	.0723
	Referencing	.0343
	Novelty	-
Style	Richness	.1192
	Formality	.0602
	Attractiveness	.0515
Topic	Popularity	.0578
	Technicality	.0047
Sentiment	Subjectivity	-
	Polarity	-
	Sentimentality	-

Table 4: Average performance across all ten folds for the GL model and for different feature sets.

Group	Aspects	RMSE	RRSE
All groups	All aspects	.3984	-
Readability	w/o Fluency	.4158	-4.36%
	w/o Conciseness	.3984	.00%
Informative.	w/o Completeness	.4233	-6.25%
	w/o Referencing	.4000	-.40%
	w/o Descriptiveness	.3999	-.37%
	w/o Novelty	.3981	-.07%
Style	w/o Richness	.4081	-2.43%
	w/o Attractiveness	.4009	-.62%
	w/o Formality	.3990	-.15%
Topic	w/o Popularity	.4003	-.47%
	w/o Technicality	.3976	.20%
Sentiment	w/o Subjectivity	.3974	.25%
	w/o Polarity	.3984	-.10%
	w/o Sentimentality	.3983	.02%

showing a higher relation to the overall editorial quality.

Next, we replicate our regression experiments for the GLM regression model, but this time we apply a leave-one-aspect-out method, to examine the relative importance of each aspect in explaining our predicted variable, i.e., the news article quality. To this end, we evaluate the 14 regression models, each one with out one of the aspects. The goal is to verify how prediction is affected by each individual quality aspect.

To compare the performance of our GLM regression model against the baseline method (with all quality aspects), we compute the Root Mean Squared Error (RMSE), given by

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y} - y_i)^2}{N}} \quad (3)$$

where $\hat{y}$ is the sample mean and y_i is the i-th estimate. However, while regression results give an idea of the prediction quality of the models they do

not quantify the size of the difference of their performance. We, therefore, also compute the Root Relative Squared Error (RRSE) metric as it provides a good indication of any relative improvement over the baseline methods, given by

$$RRSE = 1 - \frac{RMSE_{GLM}}{RMSE_{Baseline}}. \quad (4)$$

Table 4 shows the RMSE and RRSE, with respect to the GLM regression model trained on all the features. These results show that *completeness*, *fluency* and *richness* are the aspects that most affect RMSE when they are missing from the full model.

6.2 Automatic Prediction of EQ

We examined a baseline model (BaselineM) that always predicts the mean value and a baseline GLM model (BaselineShallow) trained on shallow features, to automatically predict the editorial quality. Shallow or lexical features are commonly used in traditional readability metrics, which are based on the analysis of superficial text properties. Flesh-Kincaid Grade Level (Flesch, 1979; François and Fairon, 2012), SMOG (McLaughlin, 1969), and Gunning Fog (Gunning, 1952) are some examples of readability metrics. The simplicity of these features makes them an attractive solution compared to computationally more expensive features, such as syntactic (Feng et al., 2010). However, as Shriver (Schriver, 1989) points out, the readability metrics can be useful when used as gross index of readability. For our baseline, we consider the Flesh Kincaid, Coleman Liau, ARI, RIX, Gunning Fog, SMOG, LIX features.

In Table 5, we report the average performance of the GLM regression model, BaselineM, and BaselineShallow across all folds. We note that our GLM regression model improves the RMSE by at least 40%, compared to both baselines.

Finally, as a reference for future research with the proposed corpus, we trained GLM regression models to predict each aspect individually. Table 6 presents the RMSE for each aspect, for two different sets of feature: a standard BoW and the shallow features described previously, as well as the BaselineM. Despite the simplicity of the features, we can see that the aspects can be inferred from the articles. In particular, the model trained on the BoW features achieves an RMSE that is very close to that of the BaselineM, whereas the model trained on the shallow features outperforms all other models.

Table 5: Average performance across all ten folds for the GLM, BaselineM and BaselineShallow.

Method	RMSE	RRSE
BaselineM	0.7048	43.47%
BaselineShallow	0.8937	55.41%
GLM	0.3984	-

Table 6: Average performance across all ten folds for the GL model and for different feature sets.

Aspects	**BoW**	**Shallow**	**BaselineM**
Fluency	1.1571	1.1181	1.1462
Conciseness	1.2622	1.1968	1.2456
Completeness	.8408	.7945	.8130
Referencing	.7047	.6613	.7048
Descriptiveness	.9260	.8730	.9073
Novelty	.7994	.7607	.7797
Richness	.9866	.9454	.9568
Attractiveness	.7048	.6702	.6907
Formality	.7025	.6691	.6920
Popularity	.8329	.7825	.8250
Technicality	.7923	.7409	.7907
Subjectivity	.8750	.8283	.9094
Polarity	.8109	.7780	.8009
Sentimentality	.8170	.7668	.8046

7 Conclusions

In this paper, we proposed an annotated corpus for controlling the editorial quality of online news through 14 aspects related to editors perceived quality of news articles. To this end, we performed an editorial study with expert judges either in computational linguistics, journalism, or media monitoring experts. The judges assessed a total of 561 news articles with respect to 14 aspects. The study produced valuable insights. One important finding was that high quality articles share a significant amount of variability with several of the proposed aspects, which supports the claim that the proposed aspects may characterise news article quality in an automatic and scalable way. Another finding was that *fluency*, *completeness* and *richness* are the aspects that best correlate with quality, while *technicality*, *subjectivity* and *polarity* aspects show a poor correlation with quality. This shows that the text comprehension and writing style are aspects that are more relevant than sentiment. Later, we showed that using the entire

set of 14 aspects we could predict the text quality with an RMSE of only 0.400 in a 5-point Likert-scale. This renders a very effective decomposition of news article quality into the 14 aspects. As future work, we plan to investigate other linguistic representations that can improve the automated extraction of the proposed aspects to better predict the article's perceived quality.

Acknowledgments

The first three authors were supported by MULTISENSOR project, partially funded by the European Commission, under the contract number FP7-610411. The fourth author was partially supported by FCT/MCTES through the project NOVA LINCS Ref. UID/CEC/04516/2013.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-agirre, and Weiwei Guo. 2013. sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In *In *SEM 2013: The Second Joint Conference on Lexical and Computational Semantics. Association for Computational Linguistics*.

Ioannis Arapakis, B.Barla Cambazoglu, and Mounia Lalmas. 2014a. On the feasibility of predicting news popularity at cold start. In LucaMaria Aiello and Daniel McFarland, editors, *Social Informatics*, volume 8851 of *Lecture Notes in Computer Science*, pages 290–299. Springer International Publishing.

Ioannis Arapakis, Mounia Lalmas, Berkant Barla Cambazoglu, Mari-Carmen Marcos, and Joemon M. Jose. 2014b. User engagement in online news: Under the scope of sentiment, interest, affect, and gaze. *JASIST*, 65(10):1988–2005.

Vikas Ganjigunte Ashok, Song Feng, and Yejin Choi. 2013. Success with style: Using writing style to predict the success of novels. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*.

Alexandra Balahur, Ralf Steinberger, Mijail Kabadjov, Vanni Zavarella, Erik van der Goot, Matina Halkia, Bruno Pouliquen, and Jenya Belyaeva. 2010. Sentiment analysis in the news. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, Valletta, Malta, may. European Language Resources Association (ELRA).

Roja Bandari, Sitaram Asur, and Bernardo A Huberman. 2012. The pulse of news in social media: Forecasting popularity. In *ICWSM*, pages 26–33.

Y. Benjamini and Y. Hochberg. 1995. Controlling the false discovery rate - a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57(1):289–300.

Y. Benjamini and Y. Hochberg. 2000. On the adaptive control of the false discovery fate in multiple testing with independent statistics. *Journal of Educational and Behavioral Statistics*, 25(1):60–83.

Nadjet Bouayad-Agha, Gerard Casamayor, Simon Mille, and Leo Wanner. 2012. Perspective-oriented generation of football match summaries: Old tasks, new challenges. *ACM Transactions on Speech and Language Processing (TSLP)*, 9(2):3.

Hoa Trang Dang. 2005. Overview of duc 2005. In *Proceedings of the document understanding conference*, volume 2005, pages 1–12.

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noémie Elhadad. 2010. A Comparison of Features for Automatic Readability Assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters (COLING)*, COLING '10, pages 276–284, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rudolf Franz Flesch. 1979. *How to write plain English: A book for lawyers and consumers*. Harpercollins.

Thomas François and Cédrick Fairon. 2012. An AI readability formula for French as a foreign language. In *Proceedings of the Conference on Empirical Methods in Natural Language (EMNLP)*, pages 466–477. Association for Computational Linguistics.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.

Michael Gamon. 2006. Graph-based text representation for novelty detection. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 17–24. Association for Computational Linguistics.

Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, Li Deng, and Yelong Shen. 2014. Modeling interestingness with deep neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

Namrata Godbole, Manjunath Srinivasaiah, and Steven Skiena. 2007. Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media*.

Robert Gunning. 1952. The Technique of Clear Writing. *McGraw-Hill*.

Dan Klein and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems*, volume 15. MIT Press.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 441–450, New York, NY, USA. ACM.

Annie Louis and Ani Nenkova. 2011. Text specificity and impact on quality of news summaries. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 34–42. Association for Computational Linguistics.

Annie Louis and Ani Nenkova. 2013. A corpus of science journalism for analyzing writing quality. *Dialogue & Discourse*, 4(2):87–117.

Annie Louis and Ani Nenkova. 2014. Verbose, laconic or just right: A simple computational model of content appropriateness under length constraints. pages 636–644.

G Harry McLaughlin. 1969. SMOG grading: A new readability formula. *Journal of reading, JSTOR*, 12(8):639–646.

Danielle S McNamara, Scott A Crossley, and Philip M McCarthy. 2009. Linguistic features of writing quality. *Written Communication*.

Rada Mihalcea and Andras Csomai. 2007. Wikify!: Linking documents to encyclopedic knowledge. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, pages 233–242, New York, NY, USA. ACM.

Ani Nenkova, Jieun Chae, Annie Louis, and Emily Pitler. 2010. Structural Features for Predicting the Linguistic Quality of Text. *Proceddings of the Empirical Methods in Natural Language Generation (EMNLP)*, 5790:222–241.

Ellie Pavlick and Joel Tetreault. 2016. An empirical analysis of formality in online communication. *Transactions of the Association for Computational Linguistics*, 4:61–74.

Kelly Peterson, Matt Hohensee, and Fei Xia. 2011. Email formality in the workplace: A case study on the enron corpus. In *Proceedings of the Workshop on Languages in Social Media*, pages 86–95. Association for Computational Linguistics.

Owen Phelan, Kevin McCarthy, and Barry Smyth. 2009. Using twitter to recommend real-time topical news. In *Proceedings of the third ACM conference on Recommender systems*, pages 385–388. ACM.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08.

Janice Redish. 2000. Readability formulas have even more limitations than Klare discusses. *ACM Journal of Computer Documentation (JCD)*, 24(3):132–137.

Jack C Richards and Richard W Schmidt. 2013. *Longman dictionary of language teaching and applied linguistics*, volume 78. Routledge.

Karen A Schriver. 1989. Evaluating text quality: The continuum from text-focused to reader-focused methods. *Professional Communication, IEEE Transactions on*, 32(4):238–255.

Yohei Seki, Koji Eguchi, Noriko Kando, and Masaki Aono. 2006. Opinion-focused summarization and its analysis at duc 2006. In *Proceedings of the Document Understanding Conference (DUC)*, pages 122–130.

Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. 2012. Learning from the past: answering new questions with past answers. In *Proceedings of the 21st international conference on World Wide Web*, pages 759–768. ACM.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to rank answers on large online qa collections. In *ACL*, volume 8, pages 719–727.

Rong Tang, Kwong Bor Ng, Tomek Strzalkowski, and Paul B Kantor. 2003. Automatically predicting information quality in news documents. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers-Volume 2*, pages 97–99. Association for Computational Linguistics.

Teun A. van Dijk and W. Kintsch. 1983. *Strategies of discourse comprehension*. New York: Academic Press.

Xin Yan, Dawei Song, and Xue Li. 2006. Concept-based document readability in domain specific information retrieval. In *Proceedings of the 15th ACM Conference on Information and Knowledge Management (CIKM)*, pages 540–549. ACM.

Mostafa Zamanian and Pooneh Heydari. 2012. Readability of texts: State of the art. *Theory and Practice in Language Studies*, 2(1):43–53.

Alleviating Poor Context with Background Knowledge for Named Entity Disambiguation

Ander Barrena and **Aitor Soroa** and **Eneko Agirre**
IXA NLP Group
UPV/EHU University of the Basque Country
Donostia, Basque Country
ander.barrena,a.soroa,e.agirre@ehu.eus

Abstract

Named Entity Disambiguation (NED) algorithms disambiguate mentions of named entities with respect to a knowledge-base, but sometimes the context might be poor or misleading. In this paper we introduce the acquisition of two kinds of background information to alleviate that problem: entity similarity and selectional preferences for syntactic positions. We show, using a generative Näive Bayes model for NED, that the additional sources of context are complementary, and improve results in the CoNLL 2003 and TAC KBP DEL 2014 datasets, yielding the third best and the best results, respectively. We provide examples and analysis which show the value of the acquired background information.

1 Introduction

The goal of Named Entity Disambiguation (NED) is to link each mention of named entities in a document to a knowledge-base of instances. The task is also known as Entity Linking or Entity Resolution (Bunescu and Pasca, 2006; McNamee and Dang, 2009; Hachey et al., 2012). NED is confounded by the ambiguity of named entity mentions. For instance, according to Wikipedia, *Liechtenstein* can refer to the micro-state, several towns, two castles or a national football team, among other instances. Another ambiguous entity is *Derbyshire* which can refer to a county in England or a cricket team. Most NED research use knowledge-bases derived or closely related to Wikipedia.

For a given mention in context, NED systems (Hachey et al., 2012; Lazic et al., 2015) typically rely on two models: (1) a mention module returns possible entities which can be referred to by the mention, ordered by prior probabilities; (2) a context model orders the entities according to the context of the mention, using features extracted from annotated training data. In addition, some systems check whether the entity is coherent with the rest of entities mentioned in the document, although (Lazic et al., 2015) shows that the coherence module is not required for top performance.

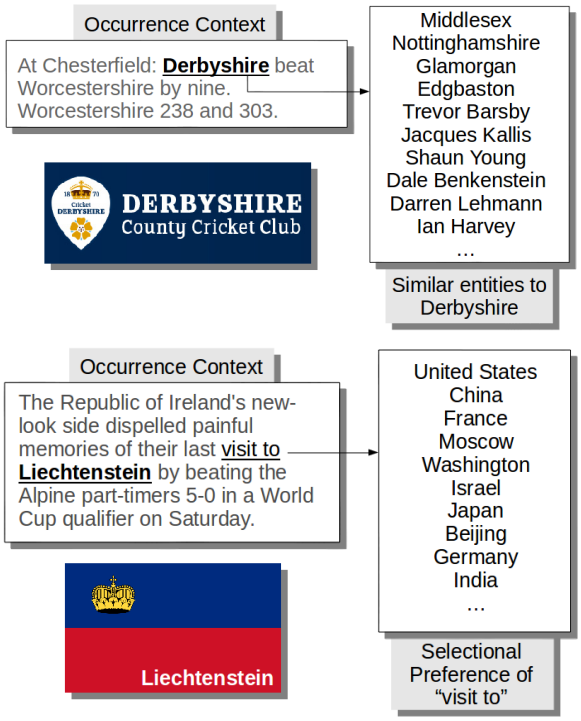


Figure 1: Two examples where NED systems fail, motivating our two background models: similar entities (top) and selectional preferences (bottom). The logos correspond to the gold label.

Figure 1 shows two real examples from the development dataset which contains text from News, where the clues in the context are too weak or misleading. In fact, two mentions in those examples (*Derbyshire* in the first and *Liechtenstein* in the second) are wrongly disambiguated by a bag-of-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1903–1912,
Berlin, Germany, August 7-12, 2016.

words context model.

In the first example, the context is very poor, and the system returns the *county* instead of the *cricket team*. In order to disambiguate it correctly one needs to be aware that *Derbyshire*, when occurring on News, is most notably associated with cricket. This background information can be acquired from large News corpora such as Reuters (Lewis et al., 2004), using distributional methods to construct a list of closely associated entities (Mikolov et al., 2013). Figure 1 shows entities which are distributionally similar to *Derbyshire*, ordered by similarity strength. Although the list might say nothing to someone not acquainted with cricket, all entities in the list are strongly related to cricket: Middlesex used to be a county in the UK that gives name to a cricket club, Nottinghamshire is a county hosting two powerful cricket and football teams, Edgbaston is a suburban area and a cricket ground, the most notable team to carry the name Glamorgan is Glamorgan County Cricket Club, Trevor Barsby is a cricketer, as are all other people in the distributional context. When using these similar entities as context, our system does return the correct entity for this mention.

In the second example, the words in the context lead the model to return the *football team* for *Liechtenstein*, instead of the *country*, without being aware that the nominal event "visit to" prefers locations arguments. This kind of background information, known as selections preferences, can be easily acquired from corpora (Erk, 2007). Figure 1 shows the most frequent entities found as arguments of "visit to" in the Reuters corpus. When using these filler entities as context, the context model does return the correct entity for this mention.

In this article we explore the addition of two kinds of background information induced from corpora to the usual context of occurrence: (1) given a mention we use distributionally similar entities as additional context; (2) given a mention and the syntactic dependencies in the context sentence, we use the selectional preferences of those syntactic dependencies as additional context. We test their contribution separately and combined, showing that they introduce complementary information.

Our contributions are the following: (1) we introduce novel background information to provide additional disambiguation context for NED; (2) we integrate this information in a Bayesian generative NED model; (3) we show that similar entities are useful when no textual context is present; (4) we show that selectional preferences are useful when limited context is present; (5) both kinds of background information help improve results of a NED system, yielding the state-of-the-art in the TAC KBP DEL 2014 dataset and getting the third best results in the CoNLL 2003 dataset; (6) we release both resources for free to facilitate reproducibility. [1]

The paper is structured as follows. We first introduce the method to acquire background information, followed by the NED system. Section 4 presents the evaluation datasets, Section 5 the development experiments and Section 6 the overall results. They are followed by related work, error analysis and the conclusions section.

2 Acquiring background information

We built our two background information resources from the Reuters corpus (Lewis et al., 2004), which comprises 250K documents. We chose this corpus because it is the one used to select the documents annotated in one of our gold standards (cf. Section 4). The documents in this corpus are tagged with categories, which we used to explore the influence of domains.

The documents were processed using a publicly available NLP pipeline, Ixa-pipes,[2] including tokenization, lematization, dependency tagging and NERC.

2.1 Similar entity mentions

Distributional similarity is known to provide useful information regarding words that have similar co-occurrences. We used the popular word2vec[3] tool to produce vector representations for named entities in the Reuters corpus. In order to build a resource that yields similar entity mentions, we took all entity-mentions detected by the NERC tool and, if they were multi word entities, joined them into a single token replacing spaces with underscores, and appended a tag to each of them. We run word2vec with default parameters on the preprocessed corpus. We only keep the vectors for named entities, but note that the corpus contains

[1] http://ixa2.si.ehu.es/anderbarrena/2016ACL_files.zip

[2] http://ixa2.si.ehu.es/ixa-pipes/

[3] https://code.google.com/archive/p/word2vec/

both named entities and other words, as they are needed to properly model co-occurrences.

Given a named entity mention, we are thus able to retrieve the named entity mentions which are most similar in the distributional vector space. All in all, we built vectors for 95K named entity mentions. Figure 1 shows the ten most similar named entities for *Derbyshire* according to the vectors learned from the Reuters corpus. These similar mentions can be seen as a way to encode some notion of a topic-related most frequent sense prior.

2.2 Selectional Preferences

Selectional preferences model the intuition that arguments of predicates impose semantic constraints (or preferences) on the possible fillers for that argument position (Resnik, 1996). In this work, we use the simplest model, where the selectional preference for an argument position is given by the frequency-weighted list of fillers (Erk, 2007).

We extract dependency patterns as follows. After we parse Reuters with the Mate dependency parser (Bohnet, 2010) integrated in IxaPipes, we extract ($H \xrightarrow{D} C$) dependency triples, where D is one of the Subject, Object or Modifier dependencies[4] (SBJ, OBJ, MOD, respectively), H is the head word and C the dependent word. We extract fillers in both directions, that is, the set of fillers in the dependent position $\{C : (H \xrightarrow{D} C)\}$, but also the fillers in the head position $\{H : (H \xrightarrow{D} C)\}$. Each such configuration forms a template, $(H \xrightarrow{D} *)$ and $(* \xrightarrow{D} C)$.

In addition to triples (single dependency relations) we also extracted tuples involving two dependency relations in two flavors: $(H \xrightarrow{D_1} C_1 \xrightarrow{D_2} C_2)$ and $(C_1 \xleftarrow{D_1} H \xrightarrow{D_2} C_2)$. Templates and fillers are defined as done for single dependencies, but, in this case, we extract fillers in any of the three positions and we thus have three different templates for each flavor.

As dependency parsers work at the word level, we had to post-process the output to identify whether the word involved in the dependency was part of a named entity identified by the NERC algorithm. We only keep tuples which involve at least one name entity. Some examples for the three kinds of tuples follow, including the frequency of occurrence, with entities shown in bold:

(beat $\xrightarrow{SBJ}$ **Australia**) 141
(refugee $\xrightarrow{MOD}$ **Hutu**) 1681
(visit $\xrightarrow{MOD}$ to $\xrightarrow{MOD}$ **United States**) 257
(match $\xrightarrow{MOD}$ against $\xrightarrow{MOD}$ **Manchester United**) 12
(Spokesman $\xleftarrow{SBJ}$ tell $\xrightarrow{OBJ}$ **Reuters**) 1378
(**The Middle East** $\xleftarrow{MOD}$ process $\xrightarrow{MOD}$ peace) 1126

When disambiguating a mention of a named entity, we check whether the mention occurs on a known dependency template, and we extract the most frequent fillers of that dependency template. For instance, the bottom example in Figure 1 shows how *Liechtenstein* occurs as a filler of the template (visit $\xrightarrow{MOD}$ to $\xrightarrow{MOD}$ *), and we thus extract the selectional preference for this template, which includes, in the figure 1, the ten most frequent filler entities.

We extracted more than 4.3M unique tuples from Reuters, producing 2M templates and their respective fillers. The most frequent dependency was MOD, followed by SUBJ and OBJ [5] The selectional preferences include 400K different named entities as fillers.

Note that selectional preferences are different from dependency path features. Dependency path features refer to features in the immediate context of the entity mention, and are sometimes added as additional features of supervised classifiers. Selectional preferences are learnt collecting fillers in the same dependency path, but the fillers occur elsewhere in the corpus.

3 NED system

Our disambiguation system is a Näive Bayes model as initially introduced by (Han and Sun, 2011a), but adapted to integrate the background information extracted from the Reuters corpus. The model is trained using Wikipedia,[6] which is also used to generate the entity candidates for each mention.

Following usual practice, candidate generation is performed off-line by constructing an association between strings and Wikipedia articles, which we call dictionary. The association is performed using article titles, redirections, disambiguation pages, and textual anchors. Each association is scored with the number of times the string was

[4] Labels are taken from the Penn Treebank https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

[5] 1.5M, 0.8M and 0.7M respectively

[6] We used a dump from 25-5-2011. This dump is close in time to annotations of the datasets used in the evaluation (c.f. Section 4)

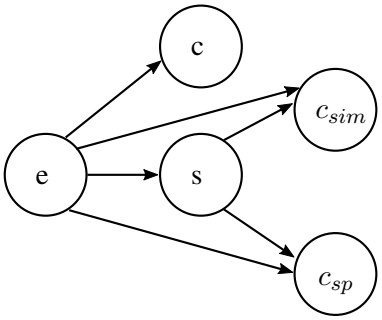


Figure 2: Dependencies among variables in our Bayesian network.

used to refer to the article (Agirre et al., 2015). We also use Wikipedia to extract training mention contexts for all possible candidate entities. Mention contexts for an entity are built by collecting a window of 50 words surrounding any hyper link pointing to that entity.

Both training and test instances are preprocessed the same way: occurrence context is tokenized, multi-words occurring in the dictionary are collapsed as a single token (longest matches are preferred). All occurrences of the same target mention in a document are disambiguated collectively, as we merge all contexts of the multiple mentions into one, following the one-entity-per-discourse hypothesis (Barrena et al., 2014).

The Näive Bayes model is depicted in Figure 2. The candidate entity e of a given mention s, which occurs within a context c, is selected according to the following formula:

$$e = \arg\max_e P(s, c, c_{\text{sp}}, c_{\text{sim}}, e) =$$
$$\arg\max_e P(e)P(s|e)P(c|e)P(c_{\text{sp}}|e, s)P(c_{\text{sim}}|e, s)$$

The formula combines evidences taken from five different probabilities: the entity prior $p(e)$, the mention probability $p(s|e)$, the textual context $p(c|s)$, the selectional preferences $P(c_{\text{sp}}|e, s)$ and the distributional similarity $P(c_{\text{sim}}|e, s)$. This formula is also referred to as the "**Full model**", as we also report results of partial models which use different combinations of the five probability estimations.

Entity prior $P(e)$ represents the popularity of entity e, and is estimated as follows:

$$P(e) \propto \frac{f(*, e) + 1}{f(*, *) + N}$$

where $f(*, e)$ is the number of times the entity e is referenced within Wikipedia, $f(*, *)$ is the total number of entity mentions and N is the number of distinct entities in Wikipedia. The estimation is smoothed using the *add-one* method.

Mention probability $P(s|e)$ represents the probability of generating the mention s given the entity e, and is estimated as follows:

$$P(s|e) \propto \theta\frac{f(s, e)}{f(*, e)} + (1 - \theta)\frac{f(s, *)}{f(*, *)}$$

where $f(s, e)$ is the number of times mention s is used to refer to entity e and $f(s, *)$ is the number of times mention s is used as anchor. We set the θ hyper-parameter to 0.9 according to developments experiments in the CoNLL testa dataset (cf. Section 5.5).

Textual context $P(c|e)$ is the probability of entity e generating the context $c = \{w_1, \ldots, w_n\}$, and is expressed as:

$$P(c|e) = \prod_{w \in c} P(w|e)^{\frac{1}{n}}$$

where $\frac{1}{n}$ is a correcting factor that compensates the effect of larger contexts having smaller probabilities. $P(w|e)$, the probability of entity e generating word w, is estimated following a bag-of-words approach:

$$P(w|e) \propto \lambda\frac{c(w, e)}{c(*, e)} + (1 - \lambda)\frac{f(w, *)}{f(*, *)}$$

where $c(w, e)$ is the number of times word w appears in the mention contexts of entity e, and $c(*, e)$ is the total number of words in the mention contexts. The term in the right is a smoothing term, calculated as the likelihood of word w being used as an anchor in Wikipedia. λ is set to 0.9 according to development experiments done in CoNLL testa.

Distributional Similarity $P(c_{\text{sim}}|e, s)$ is the probability of generating a set of similar entity mentions given an entity mention pair. This probability is calculated and estimated in exactly the same way as the textual context above, but replacing the mention context c with the mentions of the 30 most similar entities for s (cf. Section 2.1).

Selectional Preferences $P(c_{\text{sp}}|e, s)$ is the probability of generating a set of fillers c_{sp} given an entity and mention pair. The probability is again analogous to the previous ones, but using the filler entities of the selectional preferences of s instead of the context c (cf. Section 2.2). In our experiments, we select the 30 most frequent fillers for each selectional preferences, concatenating the filler list when more than one selectional preference is applied.

3.1 Ensemble model

In addition to the Full model, we created an ensemble system that combines the probabilities described above using a weighting schema, which we call "**Full weighted model**". In particular, we add an exponent coefficient to the probabilities, thus allowing to control the contribution of each model.

$$\arg\max_{e} P(e)^{\alpha} P(s|e)^{\beta}$$
$$P(c|e)^{\gamma} P(c_{\text{sp}}|e,s)^{\delta} P(c_{\text{sim}}|e,s)^{\omega}$$

We performed an exhaustive grid search in the interval $(0, 1)$ for each of the weights, using a step size of 0.05, and discarding the combinations whose sum is not one. Evaluation of each combination was performed in the CoNLL testa development set, and the best combination was applied in the test sets.[7]

4 Evaluation Datasets

The evaluation has been performed on one of the most popular datasets, the CoNLL 2003 named-entity disambiguation dataset, also know as the AIDA or CoNLL-Yago dataset (Hoffart et al., 2011). It is composed of 1393 news documents from Reuters Corpora where named entity mentions have been manually identified. It is divided in three main parts: *train*, *testa* and *testb*. We used *testa* for development experiments, and *testb* for the final results and comparison with the state-of-the-art. We ignored the training part.

In addition, we also report results in the Text Analysis Conference 2014 Diagnostic Entity Linking task dataset (TAC DEL 2014).[8] The gold standard for this task is very similar to the CoNLL dataset, where target named entity mentions have been detected by hand. Through the beginning of the task (2009 to 2013) the TAC datasets were query-driven, that is, the input included a document and a challenging and sometimes partial target-mention to disambiguate. As this task also involved mention detection and our techniques are sensitive to mention detection errors, we preferred to factor out that variation and focus on the 2014.

The evaluation measure used in this paper is micro-accuracy, that is, the percentage of linkable mentions that the system disambiguates correctly, as widely used in the CoNLL dataset. Note that TAC2014 EDL included several evaluation measures, including the aforementioned micro-accuracy of linkable mentions, but the official evaluation measure was Bcubed+ F1 score, involving also detection and clustering of mentions which refer to entities not in the target knowledge base. We decided to use the same evaluation measure for both datasets, for easier comparison. Table 1 summarizes the statistics of the datasets used in this paper where document and mention counts are presented.

Dataset	Documents	Mentions
CoNLL testa	216	4791
CoNLL testb	231	4485
TAC2014 DEL test	138	2817

Table 1: Document and linkable mention counts for CoNLL and TAC2014 DEL datasets.

5 Development experiments

We started to check the contribution of the acquired background information in the *testa* section of the CoNLL dataset. In fact, we decided to focus first on a subset of *testa* about sports,[9] and also acquired background information from the sports sub-collection of the Reuters corpus.[10] The rationale was that we wanted to start in a controlled setting, and having assumed that the domain of the test documents and the source of the background information could play a role, we decided to start focusing on the sports domain first. Another motivation is that we noticed that the ambiguity between locations and sport clubs (e.g. football, cricket, rugby, etc.) is challenging, as shown in Figure 1.

5.1 Entity similarity with no context

In our first controlled experiment, we wanted to test whether the entity similarity resource provided any added value for the cases where the target mentions had to be disambiguated out of context. Our hypothesis was that the background information from the unannotated Reuters collection, entity similarity in this case, should provide improved performance. We thus simulated a corpus where mentions have no context, extracting the named entity mentions in the sports subset that

[7] The best combination was $\alpha = 0.05$, $\beta = 0.1$, $\gamma = 0.55$ $\delta = 0.15$, $\omega = 0.15$

[8] http://www.nist.gov/tac/2014/KBP/

[9] Including 102 out of the 216 documents in *testa*, totaling 3319 mentions.

[10] Including approx. 35K documents out of the 250K documents in Reuters

Method	m-acc
$P(e)P(s\|e)$	63.83
$P(e)P(s\|e)P(c_{\text{sim}}\|e,s)$	70.98

Table 2: Results on mentions with no context on the sports subset of testa, limited to 85% of the mentions (cf. Section 5.1).

Method	m-acc
$P(e)P(s\|e)$	63.66
$P(e)P(s\|e)P(c\|e)$	66.18
$P(e)P(s\|e)P(c_{\text{sp}}\|e,s)$	67.33
$P(e)P(s\|e)P(c\|e)P(c_{\text{sp}}\|e,s)$	68.78

Table 3: Results on mentions with access to limited context on the sports subset of testa, limited to the 45% of mentions (cf. Section 5.2).

had an entry in the entity similarity resource (cf. Section 2.1), totaling 85% of the 3319 mentions.

Table 2 shows that the entity similarity resource improves the results of the model combining the entity prior and mention probability, similar to the so-called most frequent sense baseline (MFS). Note that the combination of both entity prior and mention probability is a hard-to-beat baseline, as we will see in Section 6. This experiment confirms that entity similarity information is useful when no context is present.

5.2 Selectional preferences with short context

In our second controlled experiment, we wanted to test whether the selectional preferences provided any added value for the cases where the target mentions had limited context, that of the dependency template. Our hypothesis was that the background information from the unannotated Reuters collection, selectional preferences in this case, should provide improved performance with respect to the baseline generative model of context. We thus simulated a corpus where mentions have only short context, exactly the same as the dependency templates which apply to the example, constructed extracting the named entity mentions in the sports subset that contained matching templates in the selectional preference resource (cf. Section 2.2), totaling 45% of the 3319 mentions.

Table 3 shows that the selectional preference resource (third row) allows to improve the results with respect to the no-context baseline (first row) and, more importantly, with respect to the baseline generative model (second row). The last row shows that the context model and the selectional preference model are complementary, as they produce the best result in the table. This experiment confirms that selectional preference information is effective when limited context is present.

Method	m-acc
$P(e)P(s\|e)P(c\|e)$	69.54
$P(e)P(s\|e)P(c\|e)P(c_{\text{sp}}\|e,s)$	71.25
$P(e)P(s\|e)P(c\|e)P(c_{\text{sim}}\|e,s)$	72.64
Full	73.94

Table 4: Results on mentions with limited context on the sports subset of testa, limited to the 41% of the mentions (cf. Section 5.3)

Models	Spor.	Reut.
$P(e)P(s\|e)$	65.52	65.52
$P(e)P(s\|e)P(c\|e)$	72.81	72.81
$P(e)P(s\|e)P(c\|e)P(c_{\text{sp}}\|e,s)$	73.56	73.06
$P(e)P(s\|e)P(c\|e)P(c_{\text{sim}}\|e,s)$	75.73	76.62
Full	76.30	76.87

Table 5: Results on the entire sports subset of testa: middle column uses the sports subset of Reuters to acquire background information, right column uses the full Reuters (cf. Section 5.4).

5.3 Combinations

In our third controlled experiments, we combine all three context and background models and evaluate them in the subset of the sports mentions that have entries in the similarity resource, and also contain matching templates in the selectional preference resource (41% of the sports subset). Note that, in this case, the context model has access to the entire context. Table 4 shows that, effectively, the background information adds up, with best results for the full combined model (cf. Section 3), confirming that both sources of background information are complementary to the baseline context model and between themselves.

5.4 Sports subsection of CoNLL testa

The previous experiments have been run on a controlled setting, limited to the subset where our constructed resources could be applied. In this section we report results for the entire sports subset of CoNLL testa. The middle column in Table 5 shows the results for the two baselines, and the improvements when adding the two background

models, separately, and in combination. The results show that the improvements reported in the controlled experiments carry over when evaluating to all mentions in the Sport subsection, with an accumulated improvement of 3.5 absolute points over the standard NED system (second row).

The experiments so far have tried to factor out domain variation, and thus the results have been produced using the background information acquired from the sports subset of the Reuters collection. In order to check whether this control of the target domain is necessary, reproduced the same experiment using the full Reuters collection to build the background information, as reported in the rightmost column in Table 5. The results are very similar,[11] with a small decrease for selectional preferences, a small increase for the similarity resource, and a small increase for the full system. In view of these results, we decided to use the full Reuters collection to acquire the background knowledge for the rest of the experiments, and did not perform further domain-related experiments.

5.5 Results on CoNLL testa

Finally, Table 6 reports the results on the full development dataset. The results show that the good results in the sports subsection carry over to the full dataset. The table reports results for the baseline systems (two top rows) and the addition of the background models, including the Full model, which yields the best results.

In addition, the two rows in the bottom report the results of the ensemble methods (cf. Section 3.1) which learn the weights on the same development dataset. These results are reported for completeness, as they are an over-estimation, and are over-fit. Note that all hyper-parameters have been tuned on this development dataset, including the ensemble weights, smoothing parameters λ and θ (cf. Section 3), as well as the number of similar entities and the number of fillers in the selectional preferences. The next section will show that the good results are confirmed in unseen test datasets.

6 Overall Results

In the previous sections we have seen that the background information is effective improving the results on development. In this section we report the result of our model in the popular CoNLL testb and TAC2014 DEL datasets, which allow to compare to the state-of-the-art in NED.

[11] The two first rows do not use background information, and are thus the same.

System	testa
$P(e)P(s\|e)$	73.76
$P(e)P(s\|e)P(c\|e)$	78.98
$P(e)P(s\|e)P(c\|e)P(c_{\text{sp}}\|e,s)$	79.32
$P(e)P(s\|e)P(c\|e)P(c_{\text{sim}}\|e,s)$	81.76
Full	81.90
$P(e)^{\alpha}P(s\|e)^{\beta}P(c\|e)^{\gamma}$	85.20
Full weighted	**86.62**

Table 6: Results on the full testa dataset (cf. Section 5.5).

System	CoNLL	TAC14
$P(e)P(s\|e)$	73.07	78.31
$P(e)P(s\|e)P(c\|e)$	79.98	82.11
$P(e)P(s\|e)P(c\|e)P(c_{\text{sp}}\|e,s)$	81.31	82.61
$P(e)P(s\|e)P(c\|e)P(c_{\text{sim}}\|e,s)$	82.72	83.24
Full	82.85	83.21
$P(e)^{\alpha}P(s\|e)^{\beta}P(c\|e)^{\gamma}$	86.44	81.61
Full weighted	**88.32**	**83.46**

Table 7: Overall micro accuracy results on the CoNLL testb and TAC 2014 DEL datasets.

Table 7 reports our results, confirming that both background information resources improve the results over the standard NED generative system, separately, and in combination, for both datasets (Full row). All differences with respect to the standard generative system are statistically significant according to the Wilcoxon test (p-value < 0.05).

In addition, we checked the contribution of learning the ensemble weights on the development dataset (testa). Both the generative system with and without background information improve considerably.

The error reduction between the weighted model using background information (Full weighted row) and the generative system without background information (previous row) exceeds 10% in both datasets, providing very strong results, and confirming that the improvement due to background information is consistent across both datasets, even when applied on a very strong system. The difference is statistically significant in both datasets.

System	CoNLL	TAC14
Full weighted	88.32	83.46
(Barrena et al., 2015)	83.61	80.69
(Lazic et al., 2015)	86.40	—
(Alhelbawy & Gaizauskas,14)	*87.60	—
(Chisholm and Hachey, 2015)	88.70	—
(Pershina et al., 2015)	*91.77	—
TAC14 best (Ji et al., 2014)	—	82.70

Table 8: Overall micro accuracy results on the CoNLL testb and TAC 2014 DEL datasets, including the current state-of-the-art. Starred results are not comparable, see text.

7 Related Work

Our generative model is based on (Han and Sun, 2011b), which is basically the core method used in later work (Barrena et al., 2015; Lazic et al., 2015) with good results. Although the first do not report results on our datasets the other two do. (Barrena et al., 2015) combines the generative model with a graph-based system yielding strong results in both datasets. (Lazic et al., 2015) adds a parameter estimation method which improved the results using unannotated data. Our work is complementary to those, as we could also introduce additional disambiguation probabilities (Barrena et al., 2015), or apply more sophisticated parameter estimation methods (Lazic et al., 2015).

Table 8 includes other high performing or well-known systems, which usually use complex methods to combine features coming from different sources, where our results are only second to those of (Chisholm and Hachey, 2015) in the CoNLL dataset and best in TAC 2014 DEL. The goal of this paper is not to provide the best performing system, but yet, the results show that our use of background information allows to obtain very good results.

Alhelbawy and Gaizauskas (2014) combines local and coherence features by means of a graph ranking scheme, obtaining very good results on the CONLL 2003 dataset. They evaluate on the full dataset, i.e. they test on train, testa and testb (20K, 4.8K and 4.4K mentions respectively). Our results on the same dataset are 84.25 (Full) and 88.07 (Full weighted), but note that we do tune the parameters on testa, so this might be slighly over-estimated. Our system does not use global coherence, and therefore their method is complementary to our NED system. In principle, our proposal for enriching context should improve the results of their system.

Pershina et al. (2015) propose a system closely resembling (Alhelbawy and Gaizauskas, 2014). They report the best known results on CONNL 2003 so far, but unfortunately, their results are not directly comparable to the rest of the state-of-the-art, as they artificially insert the gold standard entity in the candidate list.[12]

In (Chisholm and Hachey, 2015) the authors explore the use of links gathered from the web as an additional source of information for NED. They present a complex two-staged supervised system that incorporates global coherence features, with large amount of noisy training. Again, using additional training data seems an interesting future direction complementary to ours.

We are not aware of other works which try to use additional sources of context or background information as we do. (Cheng and Roth, 2013) use relational information from Wikipedia to add constraints to the coherence model, and is somehow reminiscent of our use dependency templates, although they focus on recognizing a fixed set of relations between entities (as in information extraction) and do not model selectional preferences. (Barrena et al., 2014) explored the use of syntactic collocations to ensure coherence, but did not model any selectional preferences.

Previous work on word sense disambiguation using selectional preference includes (McCarthy and Carroll, 2003) among others, but they report low results. (Brown et al., 2011) applied wordNet hypernyms for disambiguating verbs, but they did not test the improvement of this feature. (Taghipour and Ng, 2015) use embeddings as features which are fed into a supervised classifier, but our method is different, as we use embeddings to find similar words to be fed as additional context. None of the state-of-the-art systems, e.g. (Zhong and Ng, 2010), uses any model of selectional preferences.

8 Discussion

We performed an analysis of the cases where our background models worsened the disambiguation performance. Both distributional similarity and selectional preferences rely on correct mention detection in the background corpus. We detected

[12] https://github.com/masha-p/PPRforNED/readme.txt

that mentions where missed, which caused some coverage issues. In addition, the small size of the background corpus sometimes produces arbitrary contexts. For instance, subject position fillers of "score" include mostly basketball players like Michael Jordan or Karl Malone. A similar issue was detected in the distributional similarity resource. A larger corpus would produce a broader range of entities, and thus use of larger background corpora (e.g. Gigaword) should alleviate those issues.

Another issue was that some dependencies do not provide any focused context, as for instance arguments of *say* or *tell*. We think that a more sophisticated combination model should be able to detect which selectional preferences and similarity lists provide a focused set of instances.

9 Conclusions and Future Work

In this article we introduced two novel kinds of background information induced from corpora to the usual context of occurrence in NED: (1) given a mention we used distributionally similar entities as additional context; (2) given a mention and the syntactic dependencies in the context sentence, we used the selectional preferences of those syntactic dependencies as additional context. We showed that similar entities are specially useful when no textual context is present, and that selectional preferences are useful when limited context is present.

We integrated them in a Bayesian generative NED model which provides very strong results. In fact, when integrating all knowledge resources we yield the state-of-the-art in the TAC KBP DEL 2014 dataset and get the third best results in the CoNLL 2003 dataset. Both resources are freely available for reproducibility.[13]

The analysis of the acquired information and the error analysis show several avenues for future work. First larger corpora should allow to increase the applicability of the similarity resource, and specially, that of the dependency templates, and also provide better quality resources.

Acknowledgments

We thank the reviewers for their suggestions. This work was partially funded by MINECO (TUNER project, TIN2015-65308-C5-1-R). The IXA group is funded by the Basque Government (A type Research Group). Ander Barrena enjoys an PhD scholarship from the Basque Government.

[13] http://ixa2.si.ehu.es/anderbarrena/2016ACL_files.zip

References

Eneko Agirre, Ander Barrena, and Aitor Soroa. 2015. Studying the wikipedia hyperlink graph for relatedness and disambiguation. *CoRR*, abs/1503.01655.

Ayman Alhelbawy and Robert Gaizauskas. 2014. Graph ranking for collective named entity disambiguation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 75–80, Baltimore, Maryland, June. Association for Computational Linguistics.

Ander Barrena, Eneko Agirre, Bernardo Cabaleiro, Anselmo Peñas, and Aitor Soroa. 2014. "one entity per discourse" and "one entity per collocation" improve named-entity disambiguation. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2260–2269, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Ander Barrena, Aitor Soroa, and Eneko Agirre. 2015. Combining mention context and hyperlinks from wikipedia for named entity disambiguation. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 101–105, Denver, Colorado, June. Association for Computational Linguistics.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 89–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Susan Windisch Brown, Dmitriy Dligach, and Martha Palmer. 2011. Verbnet class assignment as a wsd task. In *Proceedings of the Ninth International Conference on Computational Semantics*, IWCS '11, pages 85–94, Stroudsburg, PA, USA. Association for Computational Linguistics.

R. C. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *Proceesings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 9–16, Trento, Italy. The Association for Computer Linguistics.

X. Cheng and D. Roth. 2013. Relational inference for wikification. In *EMNLP*.

Andrew Chisholm and Ben Hachey. 2015. Entity disambiguation with web links. *Transactions of the Association for Computational Linguistics*, 3:145–156.

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

B. Hachey, W. Radford, J. Nothman, M. Honnibal, and J.R. Curran. 2012. Evaluating Entity Linking with Wikipedia. *Artificial Intelligence*, 194:130–150, January.

X. Han and L. Sun. 2011a. A generative entity-mention model for linking entities with knowledge base. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 945–954, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xianpei Han and Le Sun. 2011b. A generative entity-mention model for linking entities with knowledge base. In *ACL HLT*.

J. Hoffart, M.A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater, and G. Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 782–792, Stroudsburg, PA, USA. Association for Computational Linguistics.

Heng Ji, Joel Nothman, and Ben Hachey. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*.

Nevena Lazic, Amarnag Subramanya, Michael Ringgaard, and Fernando Pereira. 2015. Plato: A selective context model for entity resolution. *Transactions of the Association for Computational Linguistics*, 3:503–515.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *The Journal of Machine Learning Research*, 5:361–397.

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

Paul McNamee and Hoa Dang. 2009. Overview of the TAC 2009 Knowledge Base Population track. In *TAC*.

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

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 238–243, Denver, Colorado, May–June. Association for Computational Linguistics.

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

Kaveh Taghipour and Hwee Tou Ng. 2015. Semi-supervised word sense disambiguation using word embeddings in general and specific domains. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 314–323, Denver, Colorado, May–June. Association for Computational Linguistics.

Zhi Zhong and Hwee Tou Ng. 2010. It makes sense: A wide-coverage word sense disambiguation system for free text. In *ACL: System Demonstrations*.

Mining Paraphrasal Typed Templates from a Plain Text Corpus

Or Biran
Columbia University
orb@cs.columbia.edu

Terra Blevins
Columbia University
tlb2145@columbia.edu

Kathleen McKeown
Columbia University
kathy@cs.columbia.edu

Abstract

Finding paraphrases in text is an important task with implications for generation, summarization and question answering, among other applications. Of particular interest to those applications is the specific formulation of the task where the paraphrases are templated, which provides an easy way to lexicalize one message in multiple ways by simply plugging in the relevant entities. Previous work has focused on mining paraphrases from parallel and comparable corpora, or mining very short sub-sentence synonyms and paraphrases. In this paper we present an approach which combines distributional and KB-driven methods to allow robust mining of sentence-level paraphrasal templates, utilizing a rich type system for the slots, from a plain text corpus.

1 Introduction

One of the main difficulties in Natural Language Generation (NLG) is the *surface realization* of messages: transforming a message from its internal representation to a natural language phrase, sentence or larger structure expressing it. Often the simplest way to realize messages is though the use of templates. For example, any message about the birth year and place of any person can be expressed with the template "[Person] was born in [Place] in [Year]".

Templates have the advantage that the generation system does not have to deal with the internal syntax and coherence of each template, and can instead focus on document-level discourse coherence and on local coreference issues. On the other hand, templates have two major disadvantages. First, having a human manually compose a template for each possible message is costly, especially when a generation system is relatively open-ended or is expected to deal with many domains. In addition, a text generated using templates often lacks variation, which means the system's output will be repetitive, unlike natural text produced by a human.

In this paper, we are concerned with a task aimed at solving both problems: automatically mining paraphrasal templates, i.e. groups of templates which share the same slot types and which, if their slots are filled with the same entities, result in paraphrases. We introduce an unsupervised approach to paraphrasal template mining from the text of Wikipedia articles.

Most previous work on paraphrase detection focuses either on a corpus of aligned paraphrase candidates or on such candidates extracted from a parallel or comparable corpus. In contrast, we are concerned with a very large dataset of templates extracted from a single corpus, where any two templates are potential paraphrases. Specifically, paraphrasal templates can be extracted from sentences which are not in fact paraphrases; for example, the sentences "The population of Missouri includes more than 1 million African Americans" and "Roughly 185,000 Japanese Americans reside in Hawaii" can produce the templated paraphrases "The population of [american state] includes more than [number] [ethnic group]" and "Roughly [number] [ethnic group] reside in [american state]". Looking for paraphrases among templates, instead of among sentences, allows us to avoid using an aligned corpus.

Our approach consists of three stages. First, we process the entire corpus and determine slot locations, transforming the sentences to templates (Section 4). Next, we find most approriate *type* for each slot using a large taxonomy, and group together templates which share the same set of types

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1913–1923,
Berlin, Germany, August 7-12, 2016.

as potential paraphrases (Section 5). Finally, we cluster the templates in each group into sets of paraphrasal templates (Section 6).

We apply our approach to six corpora representing diverse subject domains, and show through a crowd-sourced evaluation that we can achieve a high precision of over 80% with a reasonable similarity threshold setting. We also show that our threshold parameter directly controls the trade-off between the number of paraphrases found and the precision, which makes it easy to adjust our approach to the needs of various applications.

2 Related Work

To our knowledge, although several works exist which utilize paraphrasal templates in some way, the task of extracting them has not been defined as such in the literature. The reason seems to be a difference in priorities. In the context of NLG, Angeli et al. (2010) as well as Kondadadi et al. (2013) used paraphrasal templates extracted from aligned corpora of text and data representations in specific domains, which were grouped by the data types they relate to. Duma and Klein (2013) extract templates from Wikipedia pages aligned with RDF information from DBPedia, and although they do not explicitly mention aligning multiple templates to the same set of RDF templates, the possibility seems to exist in their framework. In contrast, we are interested in extracting paraphrasal templates from non-aligned text for general NLG, as aligned corpora are difficult to obtain for most domains.

While template extraction has been a relatively small part of NLG research, it is very prominent in the field of Information Extraction (IE), beginning with Hearst (1992). There, however, the goal is to extract good data and not to extract templates that are good for generation. Many pattern extraction (as it is more commonly referred to in IE) approaches focus on semantic patterns that are not coherent lexically or syntactically, and the idea of paraphrasal templates is not important (Chambers and Jurafsky, 2011). One exception which expicitly contains a paraphrase detection component is (Sekine, 2006).

Meanwhile, independently of templates, detecting paraphrases is an important, difficult and well-researched problem of Natural Language Processing. It has implications for the general study of semantics as well as many specific applications such as Question Answering and Summarization. Research that focuses on mining paraphrases from large text corpora is especially relevant for our work. Typically, these approaches utilize a parallel (Barzilay and McKeown, 2001; Ibrahim et al., 2003; Pang et al., 2003; Quirk et al., 2004; Fujita et al., 2012; Regneri and Wang, 2012) or comparable corpus (Shinyama et al., 2002; Barzilay and Lee, 2003; Sekine, 2005; Shen et al., 2006; Zhao et al., 2009; Wang and Callison-Burch, 2011), and there have been approaches that leverage bilingual aligned corpora as well (Bannard and Callison-Burch, 2005; Madnani et al., 2008).

Of the above, two are particularly relevant. Barzilay and Lee (2003) produce *slotted lattices* that are in some ways similar to templates, and their work can be seen as the most closely related to ours. However, as they rely on a comparable corpus and produce untyped slots, it is not directly comparable. In our approach, it is precisely the fact that we use a rich type system that allows us to extract paraphrasal templates from sentences that are not, by themselves, paraphrases and avoid using a comparable corpus. Sekine (2005) produces typed phrase templates, but the approach does not allow learning non-trivial paraphrases (that is, paraphrases that do not share the exact same keywords) from sentences that do not share the same entities (thus remaining dependent on a comparable corpus), and the type system is not very rich. In addition, that approach is limited to learning short paraphrases of relations between two entities.

Another line of research is based on contextual similarity (Lin and Pantel, 2001; Paşca and Dienes, 2005; Bhagat and Ravichandran, 2008). Here, shorter (phrase-level) paraphrases are extracted from a single corpus when they appear in a similar lexical (and in later approaches, also syntactic) context. The main drawbacks of these methods are their inability to handle longer paraphrases and their tendency to find phrase pairs that are semantically related but not real paraphrases (e.g. antonyms or taxonomic siblings).

More recent work on paraphrase detection has, for the most part, focused on classifying provided sentence pairs as paraphrases or not, using the Microsoft Paraphrase Corpus (Dolan et al., 2004). Mihalcea et al. (2006) evaluated a wide range of lexical and semantic measures of similarity and introduced a combined metric that outperformed all previous measures. Madnani et al. (2012) showed that metrics from Machine Translation can be used

to find paraphrases with high accuracy. Another line of research uses the similarity of texts in a latent space created through matrix factorization (Guo and Diab, 2012; Ji and Eisenstein, 2013). Other approaches that have been explored are explicit alignment models (Das and Smith, 2009), distributional memory tensors (Baroni and Lenci, 2010) and syntax-aware representations of multi-word phrases using word embeddings (Socher et al., 2011). Word embeddings were also used by Milajevs et al. (2014). These approaches are not comparable to ours because they focus on classification, as opposed to mining, of paraphrases.

Detecting paraphrases is closely related to research on the mathematical representation of sentences and other short texts, which draws on a vast literature on semantics, including but not limited to lexical, distributional and knowledge-based semantics. Of particular interest to us is the work of Blacoe and Lapata (2012), which show that simple combination methods (e.g., vector multiplication) in classic vector space representations outperform more sophisticated alternatives which take into account syntax and which use deep representations (e.g. word embeddings, or the distributional memory approach). This finding is appealing since classic vector space representation (distributional vectors) are easy to obtain and are interpretable, making it possible to drill into errors.

3 Taxonomy

Our method relies on a type system which links entities to one another in a taxonomy. We use a combination of WordNet (Fellbaum, 1998) and DBPedia (Auer et al., 2007), which provides both a rich top-level type system with lexicalizations of multiple senses and a large database of entities linked through the type system (the top-level DBPedia categories all have cognates in WordNet, which make the two easy to combine). Leveraging the fact that DBPedia entities have corresponding Wikipedia pages, we also use the *redirect* terms for those pages as alternative lexicalizations of the entity (e.g., the Wikipedia article "United States" has "USA" as a redirect term, among others).

4 Creating Templates

The first step to creating the templates is to find entities, which are candidates to becoming slots in the templates. Since we are trying to find sentence-level paraphrasal templates, each sentence in the corpus is a potential template.

Entities are found in multiple ways. First, we use regular expressions to find dates, percentages, currencies, counters (e.g., "9th") and general numbers. Those special cases are immediately given their known type (e.g., "date" or "percentage"). Next, after POS-tagging the entire corpus, we look for candidate entities of the following kinds: terms that contain only NNP (including NNPS) tags; terms that begin and end with an NNP and contain only NNP, TO, IN and DT tags; and terms that contain only capitalized words, regardless of the POS tags. Of these candidates, we only keep ones that appear in the taxonomy. Unlike the special cases above, the type of the slots created from these general entities is not yet known and will be decided in the next step.

At the end of this step, we have a set of partially-typed templates: one made from each sentence in the corpus, with its slots (but not their types in most cases) defined by the location of entities. We remove from this set all templates which have less than two slots as these are not likely to be interesting, and all templates which have more than five slots to avoid excessively complicated templates.

We originally experimented with simply accepting any term that appears in the taxonomy as an entity. That method, however, resulted in a large number of both errors and entities that were too general to be useful (e.g, "table", "world" and similar terms are in the taxonomy). Note that NER approaches, even relatively fine-grained ones, would not give us the same richness of types that directly comparing to the taxonomy allows (and the next step requires that each entity we handle exist in the taxonomy, anyway).

5 Template Typing and Grouping

Determining the type of a slot in the template presents two difficulties. First, there is a sense disambiuation problem, as many lexical terms have more than one sense (that is, they can correspond to more than one entry in the taxonomy). Second, even if the sense is known, it is not clear which level of the taxonomy the type should be chosen from. For example, consider the sentence "[JFK] is [New York]'s largest airport" (the terms in square brackets will become slots once their types are determined). "JFK" is ambiguous: it can be an airport, a president, a school, etc. The first

step in this process is, then, to determine which of the possible senses of the term best fits the sentence. But once we determine that the sense of "JFK" here is of an airport, there are different types we can choose. JFK is a New York Airport, which is a type of Airport, which is a type of Air Field, which is a type of Facility and so on. The specificity of the type we choose will determine the correctness of the template, and also which other templates we can consider as potential paraphrases.

Our solution is a two-stage distributional approach: *choosing the sense*, and then *choosing the type level* that best fit the context of the slot. In each stage, we construct a $pseudo-sentence$ (a collection of words in arbitrary, non-grammatical order) from words used in the taxonomy to describe each option (a sense in the first stage, and a type level in the second stage), and then use their vector representations to find the option that best matches the context.

Following the observation of Blacoe and Lapata (2012) that simple similarity metrics in traditional vector representations match and even outperform more sophisticated representations in finding relations among short texts as long as multiplication is used in forming vector representations for the texts, we use traditional context vectors as the basis of our comparisons in both stages. We collect context vectors from the entire English Wikipedia corpus, with a token window of 5. To avoid noise from rarely occuring words and reduce the size of the vectors, we remove any feature with a count below a threshold of $log_{10}(\Sigma)$ where Σ is the sum of all feature counts in the vector. Finally, the vector features are weighted with (normalized) TF*IDF.[1]

For a multi-word collection (e.g. a pseudo-sentence) ψ, we define the features of the combined vector V_ψ using the vectors of member words V_w as:

$$V_{j\psi} = (\prod_{w \in \psi} V_{jw})^{\frac{1}{|S|}} \quad (1)$$

Where V_{jw} is the value of the jth feature of V_w.

To choose the sense of the slot (the first stage), we start with S, the set of all possible senses (in the taxonomy) for the entity in the slot. We create a pseudo-sentence ψ_s from the primary lexicalizations of all types in the hierarchy above each sense s - e.g., for the airport sense of JFK we create a single pseudo-sentence $\psi_{JFK-airport-sense}$ consisting of the terms "New York airport", "airport", "air field", "facility" and so on.[2] We create a vector representation V_{ψ_s} for each ψ_s using Equation 1. Then, we create a pseudo-sentence $\psi_{context}$ for the context of the slot, composed of the words in a 5-word window to the left and right of the slot in the original sentence, and create the vector $V_{\psi_{context}}$. We choose the sense $\hat{s}$ with the highest cosine similarity to the contex:

$$\hat{s} = \arg\max_{s \in S} \cos(V_{\psi_s}, V_{\psi_{context}})$$

Note that this is a deep similarity - the similarity of the (corpus) context of the sense and the (corpus) context of the slot context; the words in the sentence themselves are not used directly.

We use the lexicalizations of all types in the hierarchy to achieve a more robust vector representation that has higher values for features that co-occur with many levels in the sense's hierarchy. For example, we can imagine that "airplane" will co-occur with many of the types for the JFK airport sense, but "La Guardia" will not (helping to lower the score of the first, too-specific sense of "New York airport") and neither will features that co-occur with other senses of a particular type - e.g., "Apple" for the "airport" type.[3]

Once the sense is chosen, we choose the proper type level to use (the second stage). Here we create a pseudo-sentence for each type level separately, composed of all possible lexicalizations for the type. For example, the "air field" type contains the lexicalizations "air field", "landing field", and "flying field". These pseudo-sentences are then compared to the context in the same way as above, and the one with highest similarity is chosen. The reason for using all lexicalizations is similar to the one for using all types when determining the sense: to create a more robust representation that down-scores arbitrary co-occurences.

At the end of this step, the templates are fully typed. Before continuing to the next step of finding paraphrases, we group all *potential* paraphrases together. Potential paraphrases are simply

[1] A "term" being a single feature count, and a "document" being a vector

[2] But we exclude a fixed, small set of the most abstract types from the first few levels of the WordNet hierarchy, as these turn out to never be useful

[3] AirPort is the name of an Apple product

groups of templates which share exactly the same set of slot types (regardless of ordering).

6 Finding Paraphrases within Groups

Each group of potential paraphrases may contain multiple sub-groups such that each of the members of the subgroup is a paraphrase of all the others. In this last stage, we use a clustering algorithm to find these sub-groups.

We define the distance between any two templates in a group as the Euclidean distance between the vectors (created using Equation 1) of the two templates with the entity slots removed (that is, the pseudo-sentences created with all words in the template outside of the slots). We tried other distance metrics as well (for example, averaging the distances between the contexts surrounding each pair of corresponding slots in both templates) but the Euclidean distance seemed to work best.

Using this metric, we apply single-linkage agglomerative clustering, with the stopping criteria defined as a threshold τ for the maximum sum of squared errors (SSE) within any cluster. Specifically, the algorithm stops linking if the cluster C that would be created by the next link satisfies:

$$log(\sum_{v}^{C} d(v, \mu_C)^2) \geq \tau$$

Where μ_C is the centroid of C and d is the Euclidean distance. The logarithm is added for convenience, since the SSE can get quite large and we want to keep τ on a smaller scale.

The intuition behind this algorithm is that some paraphrases will be very similar (lexically or on a deeper level) and easy to find, while some will be more difficult to distinguish from template pairs that are related but not paraphrasal. The single-linkage approach is essentially transductive, allowing the most obvious clusters to emerge first and avoiding the creation of a central model that will become less precise over time. The threshold is a direct mechanism for controlling the trade-off between precision and recall.

At the end of this step, any pair of templates within the same cluster is considered a paraphrase. Clusters that contain only a single template are discarded (in groups that have high distances among their member templates, often the entire group is discarded since even a single link violates the threshold).

7 Evaluation

To evaluate our method, we applied it to the six domains described in Table 1. We tried to choose a set of domains that are diverse in topic, size and degree of repeated structure across documents. For each domain, we collected a corpus composed of relevant Wikipedia articles (as described in the table) and used the method described in Sections 4-6 to extract paraphrasal templates. We used Wikipedia for convenience, since it allows us to easily select domain corpora, but there is nothing in our approach that is specific to Wikipedia; it can be applied to any text corpus.

We sampled 400 pairs of paraphrases extracted from each domain and used this set of 2400 pairs to conduct a crowd-sourced human evaluation on CrowdFlower. For each template pair, we randomly selected one and used its original entities in both templates to create two sentences about the same set of entities. The annotators were presented with this pair and asked to score the extent to which they are paraphrases on a scale from 1 to 5. Table 2 shows the labels and a brief version of the explanations provided for each. To ensure the quality of annotations, we used a set of hidden test questions throughout the evaluation and rejected the contributions of annotators which did not get at least 70% of the test questions correctly. Of those that did perform well on the test questions, we had three annotators score each pair and used the average as the final score for the pair. In 39.4% of the cases, all three annotators agreed; two annotators agreed in another 47% of the cases, and in the remaining 13.6% there was complete disagreement. The inter-annotator agreement for the two annotators that had the highest overlap (27 annotated pairs), using Cohen's Kappa, was $\kappa = 0.35$.

The overall results are shown in Figure 1. Note that because of our clustering approach, we have a choice of similarity threshold. The results are shown across a range of thresholds from 8 to 11 - it is clear from the figure that the threshold provides a way to control the trade-off between the number of paraphrases generated and their precision. Table 3 shows the results with our preferred threshold of 9.5.

The number of paraphrase clusters found changes with the threshold. For the 9.5 threshold we find 512 clusters over all domains, a little over 60% of the number of paraphrases. The distribution of their sizes is Zipfian: a few very large clus-

Domain	Description	Size	Source article link
NBA	NBA teams	30	National_Basketball_Association
States	US states	50	N/A
AuMa	Automobile manufacturers	241	List_of_automobile_manufacturers
Metal	Heavy Metal bands (original movement, 1967-1981)	291	List_of_heavy_metal_bands
CWB	Battles of the American Civil War	446	List_of_American_Civil_War_battles
Marvel	Superheroes from the Marvel Comics universe	932	Category:Marvel_Comics_superheroes

Table 1: Evaluation domains. Source article links are preceded by https://en.wikipedia.org/wiki/

Score	Label	Explanation
5	Perfect Paraphrase	The two sentences are equivalent in meaning (but allow differences in e.g. tense, wordiness or sentiment)
4	Almost Paraphrase	The two sentences are equivalent in meaning with one minor difference (e.g., change or remove one word)
3	Somewhat Paraphrase	The two sentences are equivalent in meaning with a few minor differences, or are complex sentences with a part that is a paraphrase and a part that is not
2	Related	The sentences are related in meaning, but are not paraphrases
1	Unrelated	The meanings of the sentences are unrelated

Table 2: Annotation score labels and explanations

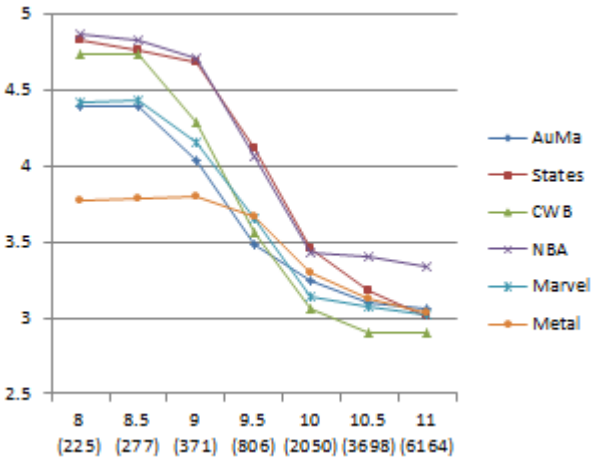


Figure 1: The average scores for each domain, for a range of threshold choices. The number in parentheses for each threshold is the number of paraphrases generated

Domain	# paraphrases	Avg.	%3+	%4+
NBA	30	4.1	88%	70%
States	171	4.1	86%	76%
AuMa	58	3.5	80%	50%
Metal	98	3.7	82%	63%
CWB	81	3.6	75%	56%
Marvel	428	3.7	83%	63%

Table 3: Size, average score, % of pairs with a score above 3 (paraphrases), and % of pairs with a score above 4 (high quality paraphrases) for the different domains with a 9.5 threshold

ters, dozens of increasingly smaller medium-sized ones and a long tail of clusters that contain only two templates.

The vast majority of paraphrase pairs come from sentences that were not originally paraphrases (i.e, sentences that originally had different entities). With a 9.5 threshold, 86% of paraphrases answer that criteria. While that number varies somewhat across thresholds, it is always above 80% and does not consistently increase or decrease as the threshold increases.

	Corpus type	Prec.	PPS
This paper, $\tau = 8$	Unaligned	94%	0.005
This paper, $\tau = 9.5$	Unaligned	82%	0.013
This paper, $\tau = 11$	Unaligned	65%	0.1
Barzilay and McKeown (2001)	Parallel	86.5%	0.1 *
Ibrahim et al. (2003)	Parallel	41.2%	0.11 *
Pang et al. (2003)	Parallel	81.5%	0.33
Barzilay and Lee (2003)	Comparable	78.5%	0.07
Bannard and Callison-Burch (2005)	Parallel bilingual	61.9%	n/a **
Zhao et al. (2009)	Parallel or Comparable	70.6%	n/a **
Wang and Callison-Burch (2011)	Comparable	67%	0.01
Fujita et al. (2012)	Parallel bilingual + unaligned	58%	0.34
Regneri and Wang (2012)	Parallel	79%	0.17

* These papers do not report the number of sentences in the corpus, but do report enough for us to estimate it (e.g. the number of documents or the size in MB)

** These papers do not report the number of paraphrases extracted, or such a number does not exist in their approach

Table 4: Comparison with the precision and paraphrases generated per input sentence (PPS) of relevant prior work

While we wanted to show a meaningful comparison with another method from previous work, none of them do what we are doing here - extraction of sentence-size paraphrasal templates from a non-aligned corpus - and so a comparison using the same data would not be fair (and in most cases, not possible). While it seems that providing the results of human evaluation without comparison to prior methods is the norm in most relevant prior work (Ibrahim et al., 2003; Paşca and Dienes, 2005; Bannard and Callison-Burch, 2005; Fujita et al., 2012), we wanted to at least get some sense of where we stand in comparison to other methods, and so we provide a list of (not directly comparable) results reported by other authors in Table 4.[4] While it is impossible to meaningfully compare and rate such different methods, these numbers support the conclusion that our single-corpus, domain-agnostic approach achieves a precision that is similar to or better than other methods. We also include the *paraphrase per sentence (PPS)* value - the ratio of paraphrases extracted to the number of input sentences of the corpus - for each method in the table. We intend this figure as the closest thing to recall that we can conceive for mining paraphrases. However, keep in mind that it is not a comparable figure across the methods, since different corpora are used. In particular, it is expected to be significantly higher for parallel corpora, where the entire corpus consists of potential paraphrases (and that fact is reflected in Table 4, where some methods that use parallel corpora have a PPS that is an order of magnitude higher than other methods).

8 Discussion and Examples

The first thing to note about the results shown in Figure 1 is that even for the highest threshold considered, which gives us a $\times 21$ improvement in size over the smallest threshold considered, all domains except CWB achieve an average score higher than 3, meaning most of the pairs extracted are paraphrases (CWB is close - a little over 2.9 on average). For the lowest threshold considered, all domains are at a precision above 88%, and for three of them it is 100%. In general, across all domains, there seems to be a significant drop in precision (and a significant boost in size) for thresholds between 9 and 10, while the precisions and sizes are fairly stable for thresholds between 8 and 9 and between 10 and 11. This result is encouraging: since the method seems to behave fairly similarly for different domains with regard to changes in the threshold, we should be able to expect similar behavior for new domains as the threshold is

[4] We always show the results of the best system described. Where needed, if results were reported in a different way than simple percentages, we use averages and other appropriate measures. Some previous work defines related sentences (as opposed to paraphrases) as positives and some does not; we do not change their numbers to fit a single definition, but we use the harsher measure for our own results

adjusted.

The magnitude of precision across domains is another matter. It is clear from the results that some domains are more difficult than others. The Metal domain seems to be the hardest: it never achieves an average score higher than 3.8. For the highest threshold, however, Metal is not different from most of the others, while CWB is significantly lower in precision. The reason seems to be the styles of the domain articles: some domains tend to have a more structured form. For example, each article in the States domain will discuss the economy, demographics, formation etc. of the state, and we are more likely to find paraphrases there (simply by virtue of there being 50×49 candidates). Articles in the Metal domain are much less structured, and there are fewer obvious paraphrase candidates. In CWB articles, there are a few repetitive themes: the outcome of the battle, the casualties, the generals involved etc., but beyond that it is fairly unstructured. This "structurality" of the domain also affects the number of paraphrases that can be found, as evident from the number of paraphrases found in the states domain in Table 3 as compared with the (much larger) Metal and CWB domains.

Table 5 shows a number of examples from each domain, along with the score given to each by the annotators. In an informal error analysis, we saw a few scenarios recurring in low-scored pairs. The Metal example at the bottom of Table 5 is a double case of bad sense disambiguation: the *album* in the second sentence ("Pyromania" in the original) happened to have a name that is also a pathological state. In addition, the *number* in the second sentence really was a date ("1980"). If we had correctly assigned the senses, these two templates would not be paraphrase candidates. The process of grouping by type is an important part of improving precision: two sentences can be misleadingly similar in the vector space, but it is less likely to have two sentences with the exact same entity types and a high vector similarity that are not close in meaning.

Another scenario is the one seen in the NBA example that was scored as 1. Here the senses were chosen correctly, but the level of the hierarchy chosen for the *person* slot was too high. If instead we had chosen *basketball coach* and *basketball player* for the two sentences respectively, they would not be considered as paraphrase candidates (and note that both meanings are implied by the templates). This sort of error does not create a problem (in our evaluation, at least) if the more accurate sense is the same in both sentences - for example, in the other NBA example (which scored 4), the *place* slot could be more accurately replaced with *sports arena* in both templates.

Cases where the types are chosen correctly do not always result in perfect paraphrases, but are typically at least related (e.g. in the examples that scored 2, and to a lesser extent those that scored 3). That scenario can be controlled using a lower threshold, with the downside that the number of paraphrases found decreases.

9 Conclusion and Future Work

We presented a method for extracting paraphrasal templates from a plain text corpus in three steps: templatizing the sentences of the corpus; finding the most appropriate type for each slot; and clustering groups of templates that share the same set of types into paraphrasal sub-groups. We conducted a crowd-sourced human evaluation and showed that our method performs similarly to or better than prior work on mining paraphrases, with three major improvements. First, we do not rely on a parallel or comparable corpus, which are not as easily obtained; second, we produce typed templates that utilize a rich, fine-grained type system, which can make them more suitable for generation; and third, by using such a type system we are able to find paraphrases from sentence pairs that are not, before templatization, really paraphrases.

Many, if not most, of the worst misidentifications seem to be the result of errors in the second stage of the approach - disambiguating the sense and specificity of the slot types. In this paper we focused on a traditional distributional approach that has the advantage of being explainable, but it would be interesting and useful to explore other options such as word embeddings, matrix factorization and semantic similarity metrics. We leave these to future work.

Another task for future work is semantic alignment. Our approach discovers paraphrasal templates without aligning them to a semantic meaning representation; while these are perfectly usable by summarization, question answering, and other text-to-text generation applications, it would be useful for concept-to-text generation and other applications to have each cluster of templates aligned

Score	Domain	Templates
5	States	Per dollar of federal tax collected in [date 1], [american state 1] citizens received approximately [money 1] in the way of federal spending.
		In [date) 1] the federal government spent [money 1] on [american state 1] for every dollar of tax revenue collected from the state.
	AuMa	Designed as a competitor to the [car 1], [car 2] and [car 3].
		It is expected to rival the [car 1], [car 2], and [car 3].
4	CWB	Federal casualties were heavy with at least [number 1] killed or mortally wounded, [number 2] wounded , and [number 3] made prisoner.
		Federal losses were [number 1] killed, [number 2] wounded, and [number 3] unaccounted for – primarily prisoners.
	NBA	For the [date 1] season, the [basketball team 1] moved into their new arena , the [place 1], with a seating capacity of [number 1].
		As a result of their success on the court, the [basketball team 1] moved into the [place 1] in [date 1], which seats over [number 1] fans.
3	Marvel	[imaginary being 1] approached [imaginary being 2], hunting for leads about the whereabouts of the X-Men.
		[imaginary being 1] and [imaginary being 2] eventually found the X-Men and became full time members.
	Metal	In [date 1], [band) 1] recorded their third studio album, "[album 1]", which was produced by Kornelije Kovač.
		[band 1] released their next full-length studio album, "[album 1]" in [date 1].
2	Auma	[company 1] and its subsidiaries created a variety of initiatives in the social sphere, initially in [country 1] and then internationally as the company expanded.
		[company 1] participated in [country 1]'s unprecedented economic growth of the 1950s and 1960s.
	Marvel	Using her powers of psychological deduction, she picked up on [first name 1]'s attraction towards her, and then [first name 2] admits she is attracted to him as well.
		While [first name 1] became shy, reserved and bookish, [first name 2] became athletically inclined, aggressive, and arrogant.
1	NBA	Though the [date 1] 76ers exceeded many on-court expectations, there was a great deal of behind-the-scenes tension between [person 1], his players, and the front office.
		After an [date 1] start, with [person 1] already hurt, these critics seemed to have been proven right.
	Metal	Within [number 1] hours of the statement, he died of bronchial pneumonia, which was brought on as a complication of [pathological state 1].
		With the album's massive success, "[pathological state 1]" was the catalyst for the [number 1] pop-metal movement.

Table 5: Examples of template pairs and their scores

to a semantic representation of the meaning expressed. Since we already discover all the entity types involved, all that is missing is the proposition (or frame, or set of propositions); this seems to be a straightforward, though not necessarily easy, task to tackle in the near future.

References

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 502–512, Stroudsburg, PA, USA. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg. Springer-Verlag.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 597–604, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Comput. Linguist.*, 36(4):673–721, December.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 16–23, Stroudsburg, PA, USA. Association for Computational Linguistics.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, ACL '01, pages 50–57, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rahul Bhagat and Deepak Ravichandran. 2008. Large Scale Acquisition of Paraphrases for Learning Surface Patterns. In *Proceedings of ACL-08: HLT*, pages 674–682, Columbus, Ohio, June. Association for Computational Linguistics.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 546–556, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 976–986, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dipanjan Das and Noah A. Smith. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proc. of ACL-IJCNLP*.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proceedings of Coling 2004*, pages 350–356, Geneva, Switzerland, Aug 23–Aug 27. COLING.

Daniel Duma and Ewan Klein. 2013. Generating natural language from linked data: Unsupervised template extraction. In *Proceedings of the 10th International Conference on Computational Semantics (IWCS 2013) – Long Papers*, pages 83–94. ASSOC COMPUTATIONAL LINGUISTICS-ACL.

Christiane Fellbaum, editor. 1998. *WordNet An Electronic Lexical Database*. The MIT Press.

Atsushi Fujita, Pierre Isabelle, and Roland Kuhn. 2012. Enlarging paraphrase collections through generalization and instantiation. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 631–642, Stroudsburg, PA, USA. Association for Computational Linguistics.

Weiwei Guo and Mona Diab. 2012. Modeling sentences in the latent space. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 864–872, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting structural paraphrases from aligned monolingual corpora. In *Proceedings of the Second International Workshop on Paraphrasing - Volume 16*, PARAPHRASE '03, pages 57–64, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 891–896. ACL.

Ravi Kondadadi, Blake Howald, and Frank Schilder. 2013. A statistical nlg framework for aggregated planning and realization. In *ACL (1)*, pages 1406–1415. The Association for Computer Linguistics.

Dekang Lin and Patrick Pantel. 2001. Dirt @sbt@discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 323–328, New York, NY, USA. ACM.

N. Madnani, Philip Resnik, Bonnie J Dorr, and R. Schwartz. 2008. Applying automatically generated semantic knowledge: A case study in machine translation. *NSF Symposium on Semantic Knowledge Discovery, Organization and Use*.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 182–190, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rada Mihalcea, Courtney Corley, and Carlo Strapparava. 2006. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 1*, AAAI'06, pages 775–780. AAAI Press.

D. Milajevs, D. Kartsaklis, M. Sadrzadeh, and M. Purver. 2014. Evaluating neural word representations in tensor-based compositional settings. In *Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Association for Computational Linguistics, Association for Computational Linguistics.

Marius Paşca and Péter Dienes. 2005. Aligning needles in a haystack: Paraphrase acquisition across the web. In *Proceedings of the Second International Joint Conference on Natural Language Processing*, IJCNLP'05, pages 119–130, Berlin, Heidelberg. Springer-Verlag.

Bo Pang, Kevin Knight, and Daniel Marcu. 2003. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 102–109, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149.

Michaela Regneri and Rui Wang. 2012. Using discourse information for paraphrase extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 916–927, Stroudsburg, PA, USA. Association for Computational Linguistics.

Satoshi Sekine. 2005. Automatic paraphrase discovery based on context and keywords between ne pairs. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, pages 80–87.

Satoshi Sekine. 2006. On-demand information extraction. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 731–738, Stroudsburg, PA, USA. Association for Computational Linguistics.

Siwei Shen, Dragomir R. Radev, Agam Patel, and Güneş Erkan. 2006. Adding syntax to dynamic programming for aligning comparable texts for the generation of paraphrases. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 747–754, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yusuke Shinyama, Satoshi Sekine, and Kiyoshi Sudo. 2002. Automatic paraphrase acquisition from news articles. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 313–318, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rui Wang and Chris Callison-Burch. 2011. Paraphrase fragment extraction from monolingual comparable corpora. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, BUCC '11, pages 52–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 834–842, Stroudsburg, PA, USA. Association for Computational Linguistics.

How to Train Dependency Parsers with Inexact Search for Joint Sentence Boundary Detection and Parsing of Entire Documents

Anders Björkelund and **Agnieszka Faleńska** and **Wolfgang Seeker** and **Jonas Kuhn**
Institut für Maschinelle Sprachverarbeitung
University of Stuttgart
{anders,falensaa,seeker,jonas}@ims.uni-stuttgart.de

Abstract

We cast sentence boundary detection and syntactic parsing as a joint problem, so an entire text document forms a training instance for transition-based dependency parsing. When trained with an early update or max-violation strategy for inexact search, we observe that only a tiny part of these very long training instances is ever exploited. We demonstrate this effect by extending the ArcStandard transition system with swap for the joint prediction task. When we use an alternative update strategy, our models are considerably better on both tasks and train in substantially less time compared to models trained with early update/max-violation. A comparison between a standard pipeline and our joint model furthermore empirically shows the usefulness of syntactic information on the task of sentence boundary detection.

1 Introduction

Although punctuation mostly provides reliable cues for segmenting longer texts into sentence units, human readers are able to exploit their understanding of the syntactic and semantic structure to (re-)segment input in the absence of such cues.

When working with carefully copy-edited text documents, sentence boundary detection can be viewed as a minor preprocessing task in Natural Language Processing, solvable with very high accuracy. However, when dealing with the output of automatic speech recognition or "noisier" texts such as blogs and emails, non-trivial sentence segmentation issues do occur. Dridan and Oepen (2013), for example, show how much impact fully automatic preprocessing can have on parsing quality for well-edited and less-edited text.

Two possible strategies to approach this problem are (i) to exploit other cues for sentence boundaries, such as prosodic phrasing and intonation in speech (e.g., Kolář et al. (2006)) or formatting cues in text documents (Read et al., 2012), and (ii) to emulate the human ability to exploit syntactic competence for segmentation. We focus here on the latter, which has received little attention, and propose to cast sentence boundary detection and syntactic (dependency) parsing as a joint problem, such that segmentations that would give rise to suboptimal syntactic structures can be discarded early on.

A joint model for parsing and sentence boundary detection by definition operates on documents rather than single sentences, as is the standard case for parsing. The task is illustrated in Figure 1, which shows the beginning of a document in the Switchboard corpus, a collection of transcribed telephone dialogues. The parser must predict the syntactic structure of the three sentences as well as the start points of each sentence.[1]

The simple fact that documents are considerably longer than sentences, often by orders of magnitude, creates some interesting challenges for a joint system. First of all, the decoder needs to handle long inputs efficiently. This problem is easily solved by using transition-based decoders, which excel in this kind of setting due to their incremental approach and their low theoretical complexity. Specifically, we use a transition-based decoder that extends the Swap transition system of Nivre (2009) in order to introduce sentence boundaries during the parsing process. The parser performs inexact search for the optimal structure by

[1] We chose this example for its brevity. For this particular example, the task of sentence boundary prediction could be solved easily with speaker information since the second sentence is from another speaker's turn. The interesting cases involve sentence segmentation within syntactically complex turns of a single speaker.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1924–1934,
Berlin, Germany, August 7-12, 2016.

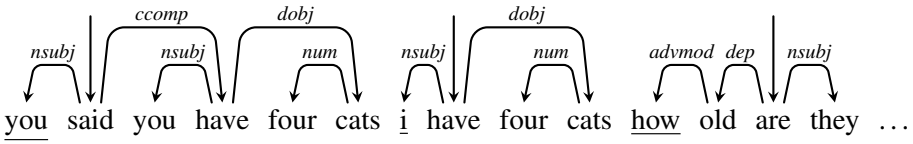


Figure 1: The beginning of a sample document from the Switchboard corpus. Tokens that start a sentence are underlined. The task is to predict syntactic structure and sentence boundaries jointly.

maintaining a beam of several candidate derivations throughout the parsing process.

We will show in this paper that, besides efficient decoding, a second, equally significant challenge lies in the way such a parser is trained. Normally, beam-search transition-based parsers are trained with structured perceptrons using either *early update* (Zhang and Clark, 2008; Collins and Roark, 2004) or *max-violation updates* (Huang et al., 2012). Yet our analysis demonstrates that neither of these update strategies is appropriate for training on very long input sequences as they discard a large portion of the training data.[2] A significant part of the training data is therefore never used to train the model. As a remedy to this problem, we instead use an adaptation of the update strategy in Björkelund and Kuhn (2014). They apply early update in a coreference resolution system and observe that the task is inherently so difficult that the correct item practically never stays in the beam. So early updates are unable to exploit the full instances during training. They propose to apply the updates iteratively on the same document until the full document has been observed. In our case, i.e. when parsing entire documents, the problem is similar in that early updates do not reach the point where the learning algorithm exploits the full training data within reasonable time. Training instead with the iterative update strategy gives us significantly better models in substantially less training time.

The second contribution in this paper is to demonstrate empirically that syntactic information can make up to a large extent for missing or unreliable cues from punctuation. The joint system implements this hypothesis and allows us to test the influence of syntactic information on the prediction of sentence boundaries as compared to a pipeline baseline where both tasks are performed independently of each other. For our analysis, we use the Wall Street Journal as the standard benchmark set and as a representative for copy-edited text. We also use the Switchboard corpus of transcribed dialogues as a representative for data where punctuation cannot give clues to a sentence boundary predictor (other types of data that may show this property to varying degrees are web content data, e.g. forum posts or chat protocols, or (especially historical) manuscripts). While the Switchboard corpus gives us a realistic scenario for a setting with unreliable punctuation, the syntactic complexity of telephone conversations is rather low compared to the Wall Street Journal. Therefore, as a controlled experiment for assessing how far syntactic competence alone can take us if we stop trusting punctuation and capitalization entirely, we perform joint sentence boundary detection/parsing on a lower-cased, no-punctuation version of the Wall Street Journal. In this setting, where the parser must rely on syntactic information alone to predict sentence boundaries, syntactic information makes a difference of 10 percentage point absolute for the sentence boundary detection task, and two points for labeled parsing accuracy.

2 Transition system

We start from the ArcStandard system extended with a swap transition to handle non-projective arcs (Nivre, 2009). We add a transition SB (for sentence boundary) that flags the front of the buffer as the beginning of a new sentence. SB blocks the SHIFT transition until the stack has been reduced and a tree has been constructed, which prevents the system from introducing arcs between separate sentences. To track the predicted sentence boundaries, we augment the configurations with a set S to hold the predicted sentence boundaries. Conceptually this leads to a representation where a document has a single artificial root

[2] We make one simplifying assumption in our experimental setup by assuming gold tokenization. Tokenization is often taken for granted, mostly because it is a fairly easy task in English. For a realistic setting, tokenization would have to be predicted as well, but since we are interested in the effect of long sequences on training, we do not complicate our setting by including tokenization.

Transition				Preconditions
LEFTARC	$(\sigma\|s_1\|s_0, \beta, A, S)$	$\Rightarrow$	$(\sigma\|s_0, \beta, A \cup \{s_0 \rightarrow s_1\}, S)$	$s_1 \neq 0$
RIGHTARC	$(\sigma\|s_1\|s_0, \beta, A, S)$	$\Rightarrow$	$(\sigma\|s_1, \beta, A \cup \{s_1 \rightarrow s_0\}, S)$	
SHIFT	$(\sigma, b_0\|\beta, A, S)$	$\Rightarrow$	$(\sigma\|b_0, \beta, A, S)$	$b_0 \neq \text{LAST}(S) \vee \|\sigma\| = 1 \vee \text{SWAPPED}(\beta)$
SWAP	$(\sigma\|s_1\|s_0, \beta, A, S)$	$\Rightarrow$	$(\sigma\|s_0, s_1\|\beta, A, S)$	$s_1 < s_0$
SB	$(\sigma, b_0\|\beta, A, S)$	$\Rightarrow$	$(\sigma, b_0\|\beta, A, S \cup \{b_0\})$	$\text{LAST}(S) < b_0 \wedge \neg\text{SWAPPED}(\beta)$

Figure 2: Transition system. $\sigma|s_1|s_0$ denotes the stack with s_0 and s_1 on top, $b_0|\beta$ denotes the buffer with b_0 in front. $\text{LAST}(S)$ denotes the most recent sentence boundary, and $\text{SWAPPED}(\beta)$ is true iff the buffer contains swapped items.

node that replaces the artificial root nodes for individual sentences.

The transition types of the system are shown in Figure 2. The configurations consist of four data structures: the stack σ, the input buffer β, the set of constructed arcs A, and the set of sentence boundaries S. LEFTARC, RIGHTARC, and SWAP have the same semantics and preconditions as in Nivre (2009). We modify the preconditions of SHIFT in order to block shifts when necessary. Whether shift is allowed can be categorized into three cases subject to the most recently predicted sentence boundary:

- If $\text{LAST}(S) < b_0$: The last predicted sentence boundary has already been shifted onto the stack. At this point, the system is building a new sentence and has not yet decided where it ends. SHIFT is therefore allowed.
- If $\text{LAST}(S) > b_0$: This situation can only occur if the system predicted a sentence boundary and subsequently made a SWAP. $\text{LAST}(S)$ then denotes the end of the current sentence and is deeper in the buffer than b_0. Thus SHIFT is allowed since b_0 belongs to the current sentence.
- If $\text{LAST}(S) = b_0$: The system must complete the current sentence by reducing the stack before it can continue shifting and SHIFT is generally not allowed, with two exceptions. (1) If the stack consists only of the root (i.e., $|\sigma| = 1$), the current sentence has been completed and the system is ready to begin parsing the next one. (2) If b_0 denotes the beginning of a new sentence, but it has been swapped back onto the buffer, then it belongs to the same sentence as the tokens currently on the stack.

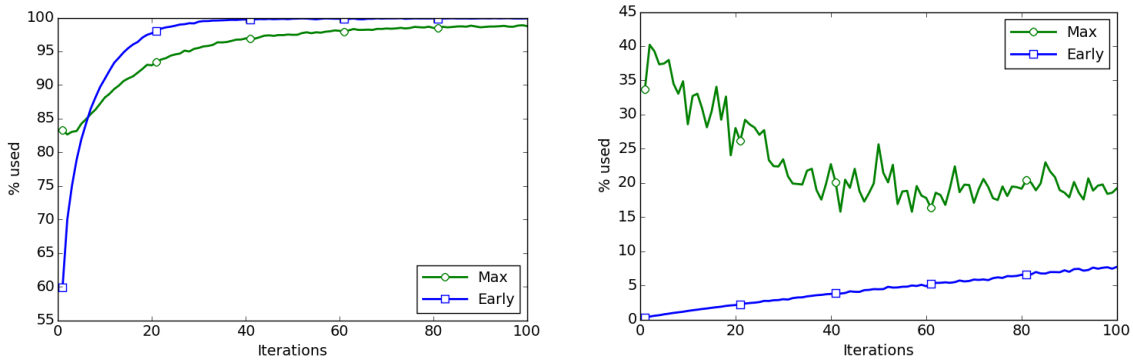
The preconditions for SB are straightforward. It is only allowed if the current b_0 is ahead of the most recently predicted sentence boundary. Additionally, the transition is not allowed if b_0 has been swapped out from the stack. If it were, then b_0 would be part of the following sentence and sentences would no longer be continuous.

Extending the transition system to also handle sentence boundaries does not affect the computational complexity. While the swap transition system has worst case $O(n^2)$ complexity, Nivre (2009) shows that swaps are rare enough that the system maintains a linear time complexity on average. A naive implementation of the configurations that make the arc set A and the sentence boundary set S explicit could result in configurations that require linear time for copying during beam search. Goldberg et al. (2013) show how this problem can be circumvented in the case of a sentence-based parser. Instead of making the arc set explicit, the arcs are reconstructed after parsing by following back-pointers to previous states. Only a small set of arcs required for feature extraction are saved in the states. We note that the same trick can be applied to avoid keeping an explicit representation of S since the system only needs to know the last predicted sentence boundary.

Snt 1	sh sh la sh sh la sh sh sb_{early} la ra ra ra sb_{late}
Snt 2	sh sh la sh sh sb_{early} la ra ra sb_{late}
Snt 3	sh sh la sh la sh sb_{early} ra ra sb_{late}

Table 1: Transition sequences including sentence boundary transitions for the example in Figure 1.

Oracle. Since each sentence constitutes its own subtree under the root node, a regular sentence-based oracle can be used to derive the oracle transition sequence for complete documents. Specifically, we apply the sentence-based oracle to each sentence, and then join the sequences with a SB transition in between. In the oracle sequences derived this way we can either apply the SB transition as *late* as possible, requiring the system to completely reduce the stack before introducing a sentence boundary. Alternatively, sentence boundaries can be introduced as *early* as possible, i.e., applying the SB transition as soon as b_0 starts a new sentence. Table 1 shows this difference in the



(a) sentences (b) documents

Figure 3: Average length of training sequences used during training for early update and max violation.

oracle transitions of each sentence from Figure 1. During preliminary experiments we compared the two alternatives and found that the early version performed better than the late.[3]

3 Learning

We focus on training structured perceptrons for search-based dependency parsing. Here, the score of a parse is defined as the scalar product of a weight vector w and a global feature vector Φ. The feature vector in turn is defined as the sum of *local* feature vectors ϕ, corresponding to the features extracted for a single transition t given a configuration c. During prediction we thus aim to obtain

$$\begin{aligned}\hat{y} &= \arg\max_{y \in \mathcal{Y}} \Phi(y) \cdot w \\ &= \arg\max_{y \in \mathcal{Y}} \sum_{(t,c) \in \textsc{TrSeq}(y)} \phi(t,c) \cdot w \end{aligned} \quad (1)$$

where TRSEQ represents the sequence of configurations and transitions executed to obtain the tree y. As the space of possible transition sequences is too large to search exhaustively, we use beam search for approximate search.

Early Update. Using approximate search while training structured perceptrons is complicated by the fact that the correct solution may actually obtain the highest score given the current model, but it was pruned off by the search procedure and therefore never considered. Collins and Roark (2004) solve this by halting search as soon as the correct solution is pruned and then making an *early update* on the partial sequences. Intuitively this makes sense, since once the correct solution is no longer reachable, it makes no sense to continue searching.

Max-violation updates. Huang et al. (2012) note that early updates require a considerable number of training iterations as it often discards a big portion of the training data. Moreover, they show that updates covering a greater subsequence can also be valid and define the *max-violation* update. Specifically, max-violation updates extend the beam beyond early updates and apply updates where the maximum difference in scores as defined by Equation (1) between the correct solution and the best prediction (i.e., the maximal violation) is used for update. They show that this leads to faster convergence compared to early update.

The curse of long sequences. Neither early nor max-violation updates commit to using the full training sequence for updates. In standard sentence-level tasks such as part-of-speech tagging or sentence-based dependency parsing these updates suffice and reasonably quickly reach a level where all or almost all of the training sequences are used for training. An entire document, however, may be composed of tens or hundreds of sentences, leading to transition sequences that are orders of magnitude longer.

To illustrate the difference between sentence-level and document-level parsing Figure 3 shows plots of the average percentage of the gold training sequences that are being used as training progresses on the Switchboard training set.

The left plot shows the parser trained on sen-

[3] One could also imagine leaving the decision of when to apply SB latent and let the machine learning decide. However, preliminary experiments again suggested that this strategy was inferior to the earliest possible point.

tences, the right one when it is trained on documents (where it also has to predict sentence boundaries). On the sentence level we see that both update strategies quite quickly reach close to 100%, i.e., they see more or less complete transition sequences during training. On the document level the picture is considerably different. The average length of seen transition sequences never even goes above 50%. In other words, more than half of the training data is never used. Early update shows a slow increase over time, presumably because the parser sees a bit more of every training instance at every iteration and therefore advances. However, max violation starts off using much more training data than early update, but then drops and settles around 20%. This illustrates the fact that max violation does not commit to exploiting more training data, but rather selects the update which constitutes the maximum violation irrespective of how much of the instance is being used. Empirically, even though the percentage of used training data decreases over iterations, max violation is still profiting from more iterations in the document-level task (cf. Figure 4 in Section 4).

Delayed LaSO. To solve the problem with the discarded training data, we follow Björkelund and Kuhn (2014) and apply the DLaSO[4] update. This idea builds on early update, but crucially differs in the sense that the remainder of a training sequence is not discarded when a mistake is made. Rather, the corresponding update is stored and the beam is reseeded with the correct solution. This enables the learning algorithm to exploit the full training data while still making sound updates (or, using the terminology of Huang et al. (2012), a number of updates that are all *violations*).

Pseudocode for DLaSO is shown in Algorithm 1. Similar to early update it performs beam search until the correct item falls off the beam (lines 9-12). Here, early update would halt, update the weights w and move on to the next instance. Instead, DLaSO computes the corresponding update, i.e., a change in the w, and stores it away. It then resets the beam to the correct solution c_i, and continues beam search. This procedure is repeated until the end of a sequence, with a final check for correctness after search has finished (line 15). After a complete pass-through of the training instance an update is made if any updates were recorded during beam search (lines 17-18).

[4] *Delayed Learning as Search Optimization*

Algorithm 1 DLaSO

Input: Training data $D = \{(x_i, y_i)\}_{i=1}^{n}$, epochs T, beam size B.
Output: Weight vector w.

```
 1: w = 0
 2: for t ∈ 1..T do
 3:     for (x, y) ∈ D do
 4:         c_{0..n} = ORACLE(y)
 5:         Beam = {c_0}
 6:         Updates = {}
 7:         for i ∈ 1 .. (n − 1) do
 8:             Beam = EXPANDANDFILTER(Beam, B)
 9:             if c_i ∉ Beam then
10:                 ŷ = BEST(Beam)
11:                 Updates = Updates ∪ CALCUPDATE(c_i, ŷ)
12:                 Beam = {c_i}
13:         Beam = EXPANDANDFILTER(Beam, B)
14:         ŷ = BEST(Beam)
15:         if c_n ≠ ŷ then
16:             Updates = Updates ∪ CALCUPDATE(c_n, ŷ)
17:         if |Updates| > 0 then
18:             w = APPLYUPDATES(w, Updates)
19: return w
```

The DLaSO update is closely related to LaSO (Daumé III and Marcu, 2005), but differs in that it delays the updates until the full instance has been decoded. Björkelund and Kuhn (2014) show that the difference is important, as it prevents the learning algorithm from getting feedback within instances. Without the delay the learner can bias the weights for rare (e.g., lexicalized) features that occur within a single instance which renders the learning setting quite different from test time inference where no such feedback is available.

4 Experimental setup

Data sets. We experiment with two parts of the English Penn Treebank (Marcus et al., 1993). We use the Wall Street Journal (WSJ) as an example of copy-edited newspaper-quality texts with proper punctuation and capitalized sentences. We also use the Switchboard portion which consists of (transcribed) telephone conversations between strangers. Following previous work on Switchboard we lowercase all text and remove punctuation and disfluency markups.

We use sections 2-21 of the WSJ for training, 24 as development set and 23 as test set. For Switchboard we follow Charniak and Johnson (2001). We convert both data sets to Stanford dependencies with the Stanford dependency converter (de Marneffe et al., 2006). We predict part-of-speech tags with the CRF tagger MarMoT (Müller et al., 2013) and annotate the training sets via 10-fold jackknifing. Depending on the experimental sce-

nario we use MARMOT in two different settings – standard sentence-level where we train and apply it on sentences, and document-level where a whole document is fed to the tagger, implicitly treating it as a single very long sentence.

Sentence boundary detection. We work with two well-established sentence boundary detection baselines. Following (Read et al., 2012) we use the tokenizer from the Stanford CoreNLP (Manning et al., 2014) and the sentence boundary detector from OpenNLP[5] which has been shown to achieve state-of-the-art results on WSJ. We evaluate the performance of sentence boundary detection on the token level using F-measure (F_1).[6]

Typical sentence boundary detectors such as CORENLP or OPENNLP focus on punctuation marks and are therefore inapplicable to data like Switchboard that does not originally include punctuation. In such cases CRF taggers are commonly selected as baselines, e.g. for punctuation prediction experiments (Zhang et al., 2013a). We therefore introduce a third baseline using MARMOT. For this, we augment the POS tags with information to indicate if a token starts a new sentence or not. We prepare the training data accordingly and train the document-level sequence labeler on them. Table 2 shows the accuracies of all baseline systems on the development sets. For WSJ all three algorithms achieve similar results which shows that MARMOT is a competitive baseline. As can be seen, predicting sentence boundaries for the Switchboard dataset is a more difficult task than for well-formatted text like the WSJ.

	WSJ	Switchboard
OPENNLP	98.09	–
CORENLP	98.60	–
MARMOT	98.21	71.78

Table 2: Results (F_1) for baselines for sentence boundary detection on dev sets.

Parser implementation. Our parser implements the labeled version of the transition system described in Section 2 with a default beam size of 20. We use the oracle by Nivre et al. (2009) to create transition sequences for each sentence of a document, and then concatenate them with SB transitions that occur as early as possible (cf. Section 2). The feature set is based on previous work (Zhang and Nivre, 2011; Bohnet and Kuhn, 2012; Bohnet et al., 2013) and was developed for a sentence-based parser for the WSJ. We made initial experiments trying to introduce new features aimed at capturing sentence boundaries such as trying to model verb subcategorization or sentence length, however none of these proved useful compared to the baseline feature set. Following the line of work by Bohnet et al., we use the passive-aggressive algorithm (Crammer et al., 2006) instead of the vanilla perceptron, parameter averaging (Collins, 2002), and a hash function to map features (Bohnet, 2010).[7]

5 Analysis

Comparison of training methods. Figure 4 shows learning curves of the different training algorithms where sentence boundary F_1 and parsing accuracy LAS are plotted as a function of training iterations. The plots show performance for early update, max-violation, and DLASO updates. In addition, a greedy version of the parser is also included. The greedy parser uses a plain averaged perceptron classifier that is trained on all the training data. The straight dashed line corresponds to the MARMOT baseline.

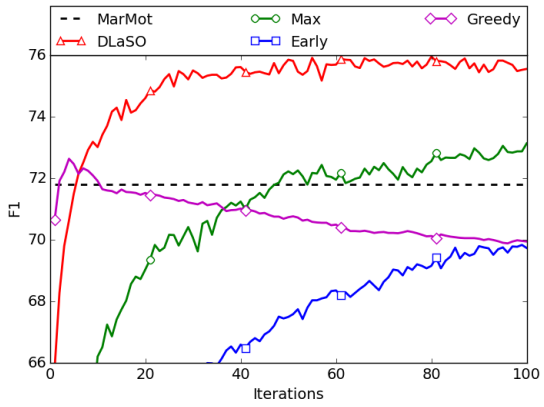
While the greedy parser, DLASO, and the MARMOT baseline all exploit the full training data during training, early update and max-violation do not (as shown in Section 3). This fact has a direct impact on the performance of these systems. DLASO reaches a plateau rather quickly, whereas even after 100 iterations, early update and max-violation perform considerably worse.[8] We also see that the greedy parser quickly reaches an optimum and then starts degrading, presumably due to overfitting. It is noteworthy, however, that max-violation needs something between 40 to 60 iterations until it reaches a level similar to the greedy parsers optimal value. This effect is quite different from single sentence parsing scenarios, where it is known that beam search parsers easily outperform the greedy counterparts, requiring not nearly as many training iterations.

[5] http://opennlp.apache.org

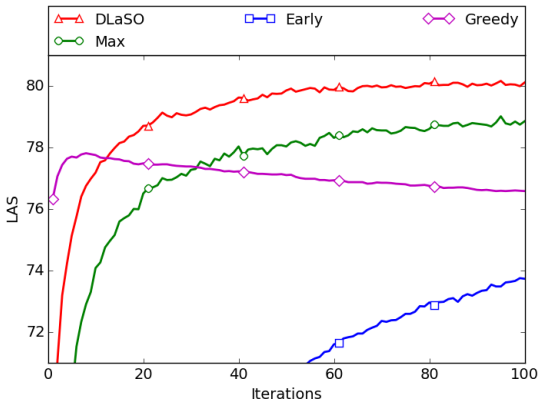
[6] A true positive is defined as a token that was correctly predicted to begin a new sentence.

[7] We make the source code of the parser available on the first author's website.

[8] The x-axis is cut at 100 iterations for simplicity. Although early update and max-violation still are growing at this point, the overall effect does not change – even after 200 iterations the DLASO update outperforms the other two by at least two points absolute.



(a) sentence boundary detection



(b) parsing

Figure 4: Performance of different update strategies on the Switchboard development set.

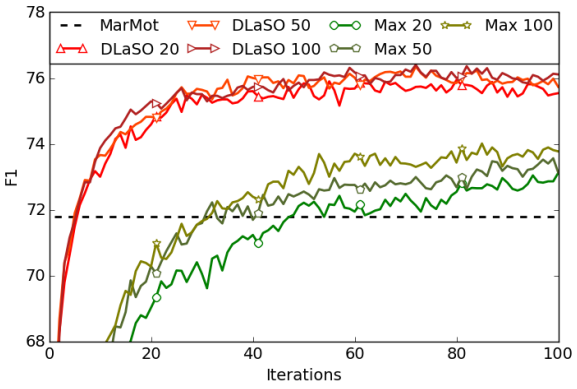


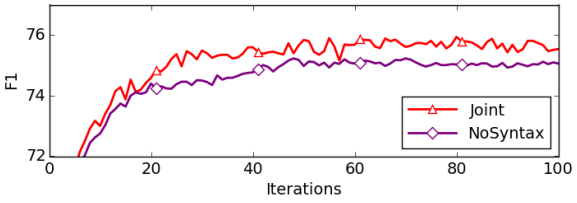
Figure 5: The effect of increasing beam size.

Increasing the beam size. Intuitively, using a bigger beam size might alleviate the problem of discarded training data and enable max-violation to exploit more training data. Figure 5 shows the sentence boundary F_1 as a function of training iterations for different beam sizes for DLaSO and max-violation. For DLaSO, we see that a bigger beam provides a slight improvement. Max-violation shows a greater correlation between greater beam size and improved F_1. However, even with a beam of size 100 max-violation is nowhere near DLaSO. In theory a beam size orders of magnitude greater may rival DLaSO but as the beam size directly influences the time complexity of the parser, this is not a viable option.

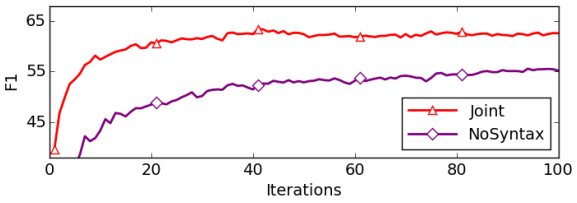
Does syntax help? One of the underlying assumptions of the joint model is our expectation that access to syntactic information should support the model in finding the sentence boundaries. We have already seen that the joint parser outperforms the MarMoT baseline by a big margin in terms of sentence boundary F_1 (Figure 4). However, the comparison is not entirely fair as the two systems use different feature sets and learning algorithms.

To properly measure the effect of syntactic information on the sentence boundary detection task, we therefore trained another model for the joint system on a treebank where we replaced the gold-standard trees with trivial trees that connect the last token of each sentence to the root node, and everything in between as a left-branching chain. We dub this setting NoSyntax and it allows us to use exactly the same machine learning for a fair comparison between a system that has access to syntax and one without.

As the syntactic complexity in Switchboard is rather low, we compare these two systems also on a version of the WSJ where we removed all punctuation and lower-cased all words, effectively making it identical to the Switchboard setting (henceforth WSJ*). Figure 6 shows sentence boundary F_1 over training iterations when training with and without access to syntax. On both data sets, the system with access to syntax stays consistently above the other system. The striking difference between the data sets is that syntactic information has a much bigger impact on WSJ* than on Switchboard, which we attribute to the higher syntactic complexity of newswire text. Overall the comparison shows clearly that syntactic structure provides useful information to the task of sentence boundary detection.



(a) Switchboard



(b) WSJ*

Figure 6: The effect of syntactic information on sentence boundary prediction, on dev sets.

6 Final results

Sentence boundary detection. We optimize the number of iterations on the dev sets: for the joint model we take the iteration with the highest average between F_1 and LAS, NOSYNTAX is tuned according to F_1. Table 3 gives the performance of the sentence boundary detectors on test sets.[9]

On WSJ all systems are close to 98 and this high number once again affirms that the task of segmenting newspaper-quality text does not leave much space for improvement. Although the parsing models outperform MARMOT, the improvements in F_1 are not significant.

In contrast, all systems fare considerably worse on WSJ* which confirms that the orthographic clues in newspaper text suffice to segment the sentences properly. Although NOSYNTAX outperforms MARMOT, the difference is not significant. However, when real syntax is used (JOINT) we see a huge improvement in F_1 – 10 points absolute – which is significantly better than both NOSYNTAX and MARMOT.

On Switchboard MARMOT is much lower and both parsing models outperform it significantly. Surprisingly the NOSYNTAX system achieves a very high result beating the baseline significantly by almost 4.5 points. The usage of syntax in the JOINT model raises this gain to 4.8 points.

[9]We test for significance using the Wilcoxon signed-rank test with $p < 0.01$. † and ‡ denote significant increases over MARMOT and NOSYNTAX, respectively. * denotes significant increases over JOINT (Table 4).

	WSJ	Switchboard	WSJ*
MARMOT	97.64	71.87	53.02
NOSYNTAX	98.21	76.31†	55.15
JOINT	98.21	76.65†	65.34†‡

Table 3: Sentence boundary detection results (F_1) on test sets.

Parsing. In order to evaluate the joint model on the parsing task separately we compare it to pipeline setups. We train a basic parser on single sentences using gold standard sentence boundaries, predicted POS tags and max-violation updates (GOLD). The number of training iterations is tuned to optimize LAS on the dev set. This parser is used as the second stage in the pipeline models. Additionally, we also build a pipeline where we use JOINT only as a sentence segmenter and then parse once again (denoted JOINT-REPARSED).

Table 4 shows the results on the test sets. For WSJ, where sentence segmentation is almost trivial, we see only minor drops in LAS between GOLD and the systems that use predicted sentence boundaries. Among the systems that use predicted boundaries, no differences are significant.

	WSJ	Switchboard	WSJ*
GOLD	90.22	84.99	88.71
MARMOT	89.81	78.93	83.37
NOSYNTAX	89.95	80.30†	83.61
JOINT	89.71	79.97†	85.66†‡
JOINT-REPARSED	89.93	80.61†‡*	85.38†‡

Table 4: Parsing results (LAS) on test sets for different sentence boundaries.

For WSJ* and Switchboard the picture is much different. Compared to GOLD, all systems show considerable drops in accuracy which asserts that errors from the sentence boundary detection task propagate to the parser and worsen the parser accuracy. On Switchboard the parsers yield significantly better results than MARMOT. The best result is obtained after reparsing and this is also significantly better than any other system. Although there is a slight drop in accuracy between NOSYNTAX and JOINT, this difference is not significant.

The results on WSJ* show that not only does syntax help to improve sentence segmentation, it does so to a degree that parsing results deteriorate when simpler sentence boundary detectors are used. Here, both JOINT and JOINT-REPARSED

obtain significantly better parsing accuracies than the systems that do not have access to syntax during sentence boundary prediction. Although JOINT-REPARSED performs a bit worse, the difference compared to JOINT is not significant.

7 Related work

Zhang and Clark (2008) first showed how to train transition-based parsers with the structured perceptron (Collins, 2002) using beam search and early update (Collins and Roark, 2004). It has since become the de facto standard way of training search-based transition-based dependency parsers (Huang and Sagae, 2010; Zhang and Nivre, 2011; Bohnet et al., 2013). Huang et al. (2012) showed how max-violation leads to faster convergence for transition-based parsers and max-violation updates have subsequently been applied to other tasks such as machine translation (Yu et al., 2013) and semantic parsing (Zhao and Huang, 2015).

Sentence Boundary Detection. Sentence boundary detection has attracted only modest attention by the research community even though it is a component in every real-world NLP application. Previous work is divided into rule-based, e.g., CoreNLP (Manning et al., 2014), and machine learning approaches (e.g., OpenNLP, a re-implementation of Reynar and Ratnaparkhi (1997)'s MxTerminator). The task is often simplified to the task of period disambiguation (Kiss and Strunk, 2006), which only works on text that uses punctuation consistently. The current state of the art uses sequence labelers, e.g., a CRF (Evang et al., 2013; Dridan and Oepen, 2013). For a broad survey of methodology and tools, we refer the reader to Read et al. (2012).

Joint models. Solving several tasks jointly has lately been popular in transition-based parsing, e.g., combining parsing with POS tagging (Hatori et al., 2011; Bohnet and Nivre, 2012) and tokenization (Zhang et al., 2013b; Zhang et al., 2014). Joint approaches avoid error propagation between the subtasks and often lead to overall better models, especially for the lower level tasks that suddenly have access to syntactic information.

Our transition system is inspired by the work of Zhang et al. (2013a). They present a projective transition-based parser that jointly predicts punctuation and syntax. Their ArcEager transition system (Nivre, 2003) includes an additional transition that introduces punctuation similar to our SB transition. They also use beam search and circumvent the problem of long training sequences by chopping up the training data into pseudo-documents of at most 10 sentences. As we have shown, this solution works because the training instances are not long enough to hurt the performance. However, while this is possible for parsing, other tasks may not be able to chop up their training data.

8 Conclusion

We have demonstrated that training a structured perceptron for inexact search on very long input sequences ignores significant portions of the training data when using early update or max-violation. We then showed how this effect can be avoided by applying a different update strategy, DLASO, which leads to considerably better models in significantly less time. This effect only occurs when the training instances are very long, e.g., on whole documents, but not when training on single sentences. We also showed that the lower performance of early update and max-violation cannot be compensated for by increasing beam size or number of iterations. We compared our system for joint sentence boundary detection and dependency parsing to competitive pipeline systems showing that syntax can provide valuable information to sentence boundary prediction when punctuation and capitalization is not available.

Acknowledgments

This work was supported by the German Research Foundation (DFG) in project D8 of SFB 732. We also thank spaCy GmbH for making it possible for the third author to finish his contribution to this work.

References

Anders Björkelund and Jonas Kuhn. 2014. Learning Structured Perceptrons for Coreference Resolution with Latent Antecedents and Non-local Features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 47–57, Baltimore, Maryland, June. Association for Computational Linguistics.

Bernd Bohnet and Jonas Kuhn. 2012. The best of bothworlds – a graph-based completion model for transition-based parsers. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 77–87,

Avignon, France, April. Association for Computational Linguistics.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea, July. Association for Computational Linguistics.

Bernd Bohnet, Joakim Nivre, Igor Boguslavsky, Richárd Farkas, Filip Ginter, and Jan Hajič. 2013. Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1:415–428.

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China, August. Coling 2010 Organizing Committee.

Eugene Charniak and Mark Johnson. 2001. Edit detection and parsing for transcribed speech. In *In Proc. NAACL*, pages 118–126.

Michael Collins and Brian Roark. 2004. Incremental Parsing with the Perceptron Algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, July.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 1–8. Association for Computational Linguistics, July.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive–aggressive algorithms. *Journal of Machine Learning Reseach*, 7:551–585, March.

Hal Daumé III and Daniel Marcu. 2005. Learning as search optimization: approximate large margin methods for structured prediction. In *ICML*, pages 169–176.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, pages 449–454.

Rebecca Dridan and Stephan Oepen. 2013. Document parsing: Towards realistic syntactic analysis. In *Proceedings of the 13th International Conference on Parsing Technologies*, Nara, Japan.

Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence labeling for word and sentence segmentation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1422–1426, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yoav Goldberg, Kai Zhao, and Liang Huang. 2013. Efficient implementation of beam-search incremental parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 628–633, Sofia, Bulgaria, August. Association for Computational Linguistics.

Jun Hatori, Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2011. Incremental joint pos tagging and dependency parsing in chinese. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1216–1224, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

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

Liang Huang, Suphan Fayong, and Yang Guo. 2012. Structured Perceptron with Inexact Search. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–151, Montréal, Canada, June. Association for Computational Linguistics.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Jáchym Kolář, Elizabeth Shriberg, and Yang Liu. 2006. Using prosody for automatic sentence segmentation of multi-party meetings. In *Text, Speech and Dialogue*, pages 629–636. Springer.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

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

Thomas Müller, Helmut Schmid, and Hinrich Schütze. 2013. Efficient higher-order CRFs for morphological tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 322–332, Seattle, Washington, USA, October. Association for Computational Linguistics.

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 73–76, Paris, France, October. Association for Computational Linguistics.

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

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, August. Association for Computational Linguistics.

Jonathon Read, Rebecca Dridan, Stephan Oepen, and Lars Jørgen Solberg. 2012. Sentence boundary detection: A long solved problem? In *Proceedings of COLING 2012: Posters*, pages 985–994, Mumbai, India, December. The COLING 2012 Organizing Committee.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19, Washington, DC, USA, March. Association for Computational Linguistics.

Heng Yu, Liang Huang, Haitao Mi, and Kai Zhao. 2013. Max-violation perceptron and forced decoding for scalable MT training. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1112–1123, Seattle, Washington, USA, October. Association for Computational Linguistics.

Yue Zhang and Stephen Clark. 2008. A Tale of Two Parsers: Investigating and Combining Graph-based and Transition-based Dependency Parsing. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 562–571, Honolulu, Hawaii, October. Association for Computational Linguistics.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193, Portland, Oregon, USA, June. Association for Computational Linguistics.

Dongdong Zhang, Shuangzhi Wu, Nan Yang, and Mu Li. 2013a. Punctuation prediction with transition-based parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 752–760, Sofia, Bulgaria, August. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2013b. Chinese parsing exploiting characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 125–134, Sofia, Bulgaria, August. Association for Computational Linguistics.

Meishan Zhang, Yue Zhang, Wanxiang Che, and Ting Liu. 2014. Character-level chinese dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1326–1336, Baltimore, Maryland, June. Association for Computational Linguistics.

Kai Zhao and Liang Huang. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1416–1421, Denver, Colorado, May–June. Association for Computational Linguistics.

MUTT: Metric Unit TesTing for Language Generation Tasks

Willie Boag, Renan Campos, Kate Saenko, Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
198 Riverside St, Lowell, MA 01854
{wboag,rcampos,saenko,arum}@cs.uml.edu

Abstract

Precise evaluation metrics are important for assessing progress in high-level language generation tasks such as machine translation or image captioning. Historically, these metrics have been evaluated using correlation with human judgment. However, human-derived scores are often alarmingly inconsistent and are also limited in their ability to identify precise areas of weakness. In this paper, we perform a case study for metric evaluation by measuring the effect that systematic sentence transformations (e.g. active to passive voice) have on the automatic metric scores. These sentence "corruptions" serve as unit tests for precisely measuring the strengths and weaknesses of a given metric. We find that not only are human annotations heavily inconsistent in this study, but that the Metric Unit TesT analysis is able to capture precise shortcomings of particular metrics (e.g. comparing passive and active sentences) better than a simple correlation with human judgment can.

1 Introduction

The success of high-level language generation tasks such as machine translation (MT), paraphrasing and image/video captioning depends on the existence of reliable and precise automatic evaluation metrics.

A man is holding a frog	There is no man holding a frog	2.1
The man is playing with a skull	There is no man playing with a skull	2.3
A man is driving a car	There is no man driving the car	3.6
A woman is combing her hair	There is no woman combing her hair	3.6
A man is walking outside	There is no man walking outside	4.6
A man is playing soccer	There is no man playing soccer	4.8

Figure 1: A few select entries from the SICK dataset. All of these entries follow the same "Negated Subject" transformation between sentence 1 and sentence 2, yet humans annotated them with an inconsistently wide range of scores (from 1 to 5). Regardless of whether the gold labels for this particular transformation should score this high or low, they *should* score be scored consistently.

Efforts have been made to create standard metrics (Papineni et al., 2001; Lin, 2004; Denkowski and Lavie, 2014; Vedantam et al., 2014) to help advance the state-of-the-art. However, most such popular metrics, despite their wide use, have serious deficiencies. Many rely on ngram matching and assume that annotators generate all reasonable reference sentences, which is infeasible for many tasks. Furthermore, metrics designed for one task, e.g., MT, can be a poor fit for other tasks, e.g., video captioning.

To design better metrics, we need a principled approach to evaluating their performance. Historically, MT metrics have been evaluated by how well they correlate with human annotations (Callison-Burch et al., 2010; Machacek and Bojar, 2014). However, as we demonstrate in Sec. 5, human judgment can result in inconsistent scoring. This presents a serious problem for determining whether

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1935–1943,
Berlin, Germany, August 7-12, 2016.

a metric is "good" based on correlation with inconsistent human scores. When "gold" target data is unreliable, even good metrics can appear to be inaccurate.

Furthermore, correlation of system output with human-derived scores typically provides an overall score but fails to isolate specific errors that metrics tend to miss. This makes it difficult to discover system-specific weaknesses to improve their performance. For instance, an ngram-based metric might effectively detect non-fluent, syntactic errors, but could also be fooled by legitimate paraphrases whose ngrams simply did not appear in the training set. Although there has been some recent work on paraphrasing that provided detailed error analysis of system outputs (Socher et al., 2011; Madnani et al., 2012), more often than not such investigations are seen as above-and-beyond when assessing metrics.

The goal of this paper is to propose a process for consistent and informative automated analysis of evaluation metrics. This method is demonstrably more consistent and interpretable than correlation with human annotations. In addition, we extend the SICK dataset to include un-scored fluency-focused sentence comparisons and we propose a toy metric for evaluation.

The rest of the paper is as follows: Section 2 introduces the corruption-based metric unit testing process, Section 3 lists the existing metrics we use in our experiments as well as the toy metric we propose, Section 4 describes the SICK dataset we used for our experiments, Section 5 motivates the need for corruption-based evaluation instead of correlation with human judgment, Section 6 describes the experimental procedure for analyzing the metric unit tests, Section 7 analyzes the results of our experiments, and in Section 8 we offer concluding remarks.

2 Metric Unit TesTs

We introduce metric unit tests based on *sentence corruptions* as a new method for automatically evaluating metrics developed for language generation tasks. Instead of obtaining human ranking for system output and comparing it with the metric-based ranking, the idea is to modify existing references with specific transformations, and examine the scores assigned by various metrics to such corruptions. In this paper, we analyze three broad categories of transformations – meaning-altering, meaning-preserving, and fluency-disrupting sentence corruptions – and we evaluate how successfully several common metrics can detect them.

As an example, the original sentence "A man is playing a guitar." can be corrupted as follows:

> *Meaning-Altering*: A man is not playing guitar.
> *Meaning-Preserving*: A guitar is being played by a man.
> *Fluency-Disrupting*: A man a guitar is playing.

Examples for each corruption type we consider are shown in Tables 1 and 2.

2.1 Meaning-altering corruptions

Meaning-altering corruptions modify the semantics of a sentence, resulting in a new sentence that has a different meaning. Corruptions (1–2) check whether a metric can detect small lexical changes that cause the sentence's semantics to entirely change. Corruption (3) is designed to fool distributed and distributional representations of words, whose vectors often confuse synonyms and antonyms.

2.2 Meaning-preserving corruptions

Meaning-preserving corruptions change the lexical presentation of a sentence while still preserving meaning and fluency. For such transformations, the "corruption" is actually logically equivalent to the original sentence, and we would expect that consistent annotators would assign roughly the same score to each. These transformations include changes such as rephrasing a sentence from active voice to passive voice (4) or paraphrasing within a sentence (5).

2.3 Fluency disruptions

Beyond understanding semantics, metrics must also recognize when a sentence lacks fluency and grammar. Corruptions (7–9) were created for this reason, and do so by generating ungrammatical sentences.

Meaning Altering			
1	*negated subject* (337)	"A man is playing a harp"	"There is no man playing a harp"
2	*negated action* (202)	"A jet is flying"	"A jet is not flying"
3	*antonym replacement* (246)	"a dog with short hair"	"a dog with long hair"
Meaning Preserving			
4	*active-to-passive* (238)	"A man is cutting a potato"	"A potato is being cut by a man"
5	*synonymous phrases* (240)	"A dog is eating a doll"	"A dog is biting a doll"
6	*determiner substitution* (65)	"A cat is eating food"	"The cat is eating food"

Table 1: Corruptions from the SICK dataset. The left column lists the number of instances for each corruption type.

Fluency disruptions			
7	*double PP* (500)	"A boy walks at night"	"A boy walks at night at night"
8	*remove head from PP* (500)	"A man danced in costume"	"A man danced costume"
9	*re-order chunked phrases* (500)	"A woman is slicing garlics"	"Is slicing garlics a woman"

Table 2: Generated corruptions. The first column gives the total number of generated corruptions in parentheses.

3 Metrics Overview

3.1 Existing Metrics

Many existing metrics work by identifying lexical similarities, such as n-gram matches, between the candidate and reference sentences. Commonly-used metrics include BLEU, CIDEr, and TER:

- BLEU, an early MT metric, is a precision-based metric that rewards candidates whose words can be found in the reference but penalizes short sentences and ones which overuse popular n-grams (Papineni et al., 2001).
- CIDEr, an image captioning metric, uses a consensus-based voting of tf-idf weighted ngrams to emphasize the most unique segments of a sentence in comparison (Vedantam et al., 2014).
- TER (Translation Edit Rate) counts the changes needed so the surface forms of the output and reference match (Snover et al., 2006).

Other metrics have attempted to capture similarity beyond surface-level pattern matching:

- METEOR, rather than strictly measuring ngram matches, accounts for soft similarities between sentences by computing synonym and paraphrase scores between sentence alignments (Denkowski and Lavie, 2014).
- BADGER takes into account the contexts over the entire set of reference sentences by using a simple compression distance calculation after performing a series of normalization steps (Parker, 2008).
- TERp (TER-plus) minimizes the edit distance by stem matches, synonym matches, and phrase substitutions before calculating the TER score, similar to BADGER's normalization step (Snover et al., 2009).

We evaluate the strengths and weaknesses of these existing metrics in Section 7.

3.2 Toy Metric: W2V-AVG

To demonstrate how this paper's techniques can also be applied to measure a new evaluation metric, we create a toy metric, W2V-AVG, using the cosine of the centroid of a sentence's word2vec embeddings (Mikolov et al., 2013). The goal for this true bag-of-words metric is to serve as a sanity check for how corruption unit tests can identify metrics that capture soft word-level similarities, but cannot handle directed relationships between entities.

4 Datasets

4.1 SICK

All of our experiments are run on the Sentences Involving Compositional Knowledge (SICK) dataset,

which contains entries consisting of a pair of sentences and a human-estimated semantic relatedness score to indicate the similarity of the sentence pair (Marelli et al., 2014). The reason we use this data is twofold:

1. it is a well-known and standard dataset within semantic textual similarity community.

2. it contains many common sentence transformation patterns, such as those described in Table 1.

The SICK dataset was built from the 8K ImageFlickr dataset[1] and the SemEval 2012 STS MSR-Video Description corpus[2]. Each of these original datasets contain human-generated descriptions of images/videos – a given video often has 20-50 reference sentences describing it. These reference sets prove very useful because they are more-or-less paraphrases of one another; they all describe the same thing. The creators of SICK selected sentence pairs and instructed human annotators to ensure that all sentences obeyed proper grammar. The creators of SICK ensured that two of the corruption types – meaning-altering and meaning-preserving – were generated in the annotated sentence pairs. We then filtered through SICK using simple rule-based templates[3] to identify each of the six corruption types listed in Table 1. Finally, we matched the sentences in the pair back to their original reference sets in the Flickr8 and MSR-Video Description corpora to obtain reference sentences for our evaluation metrics experiments.

4.2 SICK+

Since all of the entries in the SICK dataset were created for compositional semantics, every sentence was manually checked by annotators to ensure fluency. For our study, we also wanted to measure the effects of bad grammar between sentences, so we automatically generated our own corruptions to the SICK dataset to create SICK+, a set of fluency-disrupting corruptions. The rules to generate these corruptions were simple operations involving chunking and POS-tagging. Fortunately, these corruptions were, by design, meant to be ungrammatical, so there was no need for (difficult) automatic correctness checking for fluency.

5 Inconsistencies in Human Judgment

A major issue with comparing metrics against human judgment is that human judgments are often inconsistent. One reason for this is that high-level semantic tasks are difficult to pose to annotators. Consider SICK's semantic relatedness annotation as a case study for human judgment. Annotators were shown two sentences, were asked "To what extent are the two sentences expressing related meaning?", and were instructed to select an integer from 1 (completely unrelated) to 5 (very related). We can see the difficulty annotators faced when estimating semantic relatedness, especially because the task description was intentionally vague to avoid biasing annotator judgments with strict definitions. In the end, "the instructions described the task only through [a handful of] examples of relatedness" (Marelli et al., 2014).

As a result of this hands-off annotation guideline, the SICK dataset contains glaring inconsistencies in semantic relatedness scores, even for sentence pairs where the only difference between two sentences is due to the same known transformation. Figure 1 demonstrates the wide range of human-given scores for the pairs from SICK that were created with the *Negated Subject* transformation. Since the guidelines did not establish how to handle the effect of this transformation, some annotators rated it high for describing the same actions, while others rated it low for having completely opposite subjects.

To better appreciate the scope of these annotation discrepancies, Figure 2 displays the distribution of "gold" human scores for every instance of the "Negated Subject" transformation. We actually find that the relatedness score approximately follows a normal distribution centered at 3.6 with a standard

[1] http://nlp.cs.illinois.edu/HockenmaierGroup/data.html

[2] http://www.cs.york.ac.uk/semeval-2012/task6/index.php?id=data

[3] For instance, the "Antonym Replacement" template checked to see if the two sentences were one word apart, and if so whether they had a SICK-annotated NOT_ENTAILMENT relationsip.

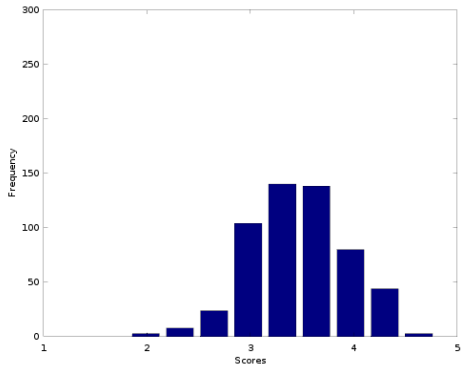


Figure 2: Human annotations for the *Negated Subject* corruption.

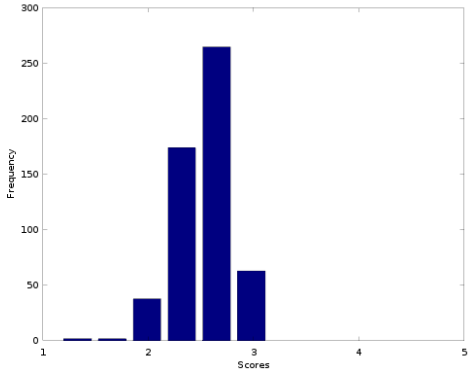


Figure 3: Metric predictions for the *Negated Subject* corruption.

deviation of about .45. The issue with this distribution is that regardless of whether the annotators rank this specific transformation as low or high, they *should* be ranking it consistently. Instead, we see that their annotations span all the way from 2.5 to 4.5 with no reasonable justification as to why.

Further, a natural question to ask is whether all sentence pairs within this common *Negated Subject* transformation do, in fact, share a structure of how similar their relatedness scores "should" be. To answer this question, we computed the similarity between the sentences in an automated manner using three substantially different evaluation metrics: METEOR, BADGER, and TERp. These three metrics were chosen because they present three very different approaches for quantifying semantic similarity, namely: sentence alignments, compression redundancies, and edit rates. We felt that these different approaches for processing the sentence pairs would allow for different views of their underlying relatedness.

To better understand how similar an automatic metric would rate these sentence pairs, Figure 3 shows the distribution over scores predicted by the METEOR metric. The first observation is that the metric produces scores that are far more peaky than the gold scores in Figure 2, which indicates that they have a significantly more consistent structure about them.

In order to see how each metric's scores compare, Table 3 lists all pairwise correlations between the gold and the three metrics. As a sanity check, we can see that the 1.0s along the diagonal indicate perfect correlation between a prediction and itself. More interestingly, we can see that the three metrics have alarmingly low correlations with the gold scores: 0.09, 0.03, and 0.07. However, we also see that the three metrics all have significantly higher correlations amongst one another: 0.80, 0.80, and 0.91. This is a very strong indication that the three metrics all have approximate agreement about how the various sentences should be scored, but this consensus is not at all reflected by the human judgments.

6 MUTT Experiments

In our Metric Unit TesTing experiments, we wanted to measure the fraction of times that a given metric is able to appropriately handle a particular corruption type. Each (original,corruption) pair is considered a trial, which the metric either gets correct or

	gold	METEOR	BADGER	TERp
gold	1.00	0.09	0.03	0.07
METEOR	0.09	1.00	0.91	0.80
BADGER	0.03	0.91	1.00	0.80
TERp	0.07	0.80	0.80	1.00

Table 3: Pairwise correlation between the predictions of three evaluation metrics and the gold standard.

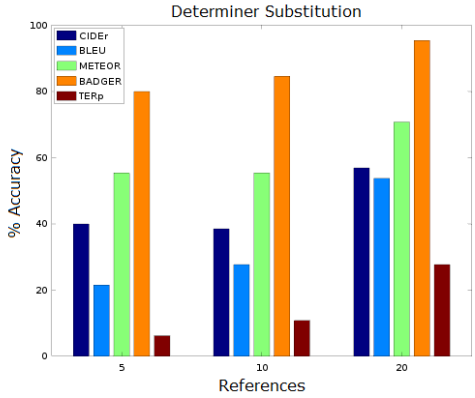


Figure 4: Results for the *Determiner Substitution* corruption (using *Difference formula* scores).

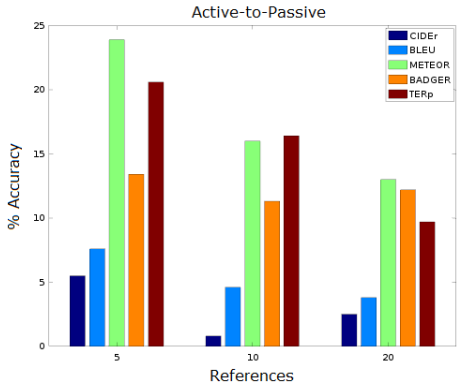


Figure 5: Results for the *Active-to-Passive* corruption (using *Difference formula* scores).

incorrect. We report the percent of successful trials for each metric in Tables 4, 5, and 6. Experiments were run using 5, 10, and 20 reference sentences to understand which metrics are able perform well without much data and also which metrics are able to effectively use more data to improve. An accuracy of 75% would indicate that the metric is able to assign appropriate scores 3 out of 4 times.[4]

For Meaning-altering and Fleuncy-disrupting corruptions, the corrupted sentence will be truly different from the original and reference sentences. A trial would be successful when the score of the original s_{orig} is rated higher than the score of the corruption s_{corr}:

$$s_{orig} > s_{corr}$$

Alternatively, Meaning-preserving transformations create a "corruption" sentence which is just as correct as the original. To reflect this, we consider a trial to be successful when the score of the corruption s_{corr} is within 15% of the score of the original s_{orig}:

$$\left| \frac{s_{orig} - s_{corr}}{s_{orig} + \epsilon} \right| \leq 0.15$$

where ϵ is a small constant (10^{-9}) to prevent division by zero. We refer to this alternative trial formulation as the *Difference formula*.

[4]Our code is made available at `https://github.com/text-machine-lab/MUTT`

7 Discussion

7.1 Meaning-altering corruptions

As shown by the middle figure in Table 4, it is CIDEr which performs the best for Antonym Replacement. Even with only a few reference sentences, it is already able to score significantly higher than the other metrics. We believe that a large contributing factor for this is CIDEr's use of tf-idf weights to emphasize the important aspects of each sentence, thus highlighting the modified when compared against the reference sentences.

The success of these metrics reiterates the earlier point about metrics being able to perform more consistently and reliably than human judgment.

7.2 Meaning-preserving corruptions

The graph in Figure 4 of the determiner substitution corruption shows an interesting trend: as the number of references increase, all of the metrics increase in accuracy. This corruption replaces "a" in the candidate with a "the', or vice versa. As the references increase, there we tend to see more examples which use these determiners interchanagably while keeping the rest of the sentence's meaning the same. Since a large number of references results in far more for the pair to agree on, the two scores are very close.

Conversely, the decrease in accuracy in the

1. Negated Subject			
num refs	5	10	20
CIDEr	99.4	99.4	99.4
BLEU	99.1	99.7	99.7
METEOR	97.0	98.5	98.2
BADGER	97.9	97.6	98.2
TERp	99.7	99.7	99.4

2. Negated Action			
num refs	5	10	20
CIDEr	98.5	98.5	98.5
BLEU	97.5	97.5	98.0
METEOR	96.0	96.0	97.0
BADGER	93.6	95.5	96.5
TERp	95.5	97.0	95.0

3. Antonym Replacement			
num refs	5	10	20
CIDEr	**86.2**	**92.7**	**93.5**
BLEU	76.4	85.4	88.6
METEOR	80.9	86.6	91.5
BADGER	76.0	85.8	88.6
TERp	75.2	79.7	80.1

Table 4: **Meaning-altering corruptions**. These % accuracies represent the number of times that a given metric was able to correctly score the original sentence higher than the corrupted sentence. Numbers referenced in the prose analysis are highlighted in bold.

4. Active-to-Passive			
num refs	5	10	20
CIDEr	5.5	0.8	2.5
BLEU	7.6	4.6	3.8
METEOR	**23.9**	**16.0**	**13.0**
BADGER	13.4	11.3	12.2
TERp	20.6	16.4	9.7

5. Synonymous Phrases			
num refs	5	10	20
CIDEr	32.1	26.2	30.0
BLEU	45.0	36.7	34.2
METEOR	62.1	62.1	62.1
BADGER	80.8	80.4	86.7
TERp	53.3	46.7	41.2

6. DT Substitution			
num refs	5	10	20
CIDEr	40.0	38.5	56.9
BLEU	21.5	27.7	53.8
METEOR	55.4	55.4	70.8
BADGER	**80.0**	**84.6**	**95.4**
TERp	6.2	10.8	27.7

Table 5: **Meaning-preserving corruptions**. These % accuracies represent the number of times that a given metric was able to correctly score the semantically-equaivalent "corrupted" sentence within 15% of the original sentence. Numbers referenced in the prose analysis are highlighted in bold.

7. Duplicate PP			
num refs	5	10	20
CIDEr	100	99.0	100
BLEU	100	100	100
METEOR	95.1	98.5	99.5
BADGER	**63.5**	**70.0**	**74.9**
TERp	96.6	99.0	99.0

8. Remove Head From PP			
num refs	5	10	20
CIDEr	69.5	76.8	80.8
BLEU	**63.5**	**81.3**	**87.7**
METEOR	60.6	72.9	84.2
BADGER	63.1	67.0	71.4
TERp	52.7	66.5	70.4

9. Re-order Chunks			
num refs	5	10	20
CIDEr	91.4	95.6	96.6
BLEU	83.0	91.4	94.2
METEOR	**81.2**	**89.6**	**92.4**
BADGER	95.4	96.6	97.8
TERp	91.0	93.4	93.4

Table 6: **Fluency-disrupting corruptions**. These % accuracies represent the number of times that a given metric was able to correctly score the original sentence higher than the corrupted sentence. Numbers referenced in the prose analysis are highlighted in bold.

Active-to-Passive table reflects how adding more (mostly active) references makes the system more (incorrectly) confident in choosing the active original. As the graph in Figure 5 shows, METEOR performed the best, likely due to its sentence alignment approach to computing scores.

7.3 Fluency disruptions

All of the metrics perform well at identifying the duplicate prepositional phrase corruption, except for BADGER which has noticeably lower accuracy scores than the rest. These lower scores may be attributed to the compression algorithm that it uses to compute similarity. Because BADGER's algorithm works by compressing the candidate and references jointly, we can see why a repeated phrase would be of little effort to encode – it is a compression algorithm, after all. The result of easy-to-compress redundancies is that the original sentence and its corruption have very similar scores, and BADGER gets fooled.

Unlike the other two fluency-disruptions, none of

the accuracy scores of the "Remove Head from PP" corruption reach 90%, so this corruption could be seen as one that metrics could use improvement on. BLEU performed the best on this task. This is likely due to its ngram-based approach, which is able to identify that deleting a word breaks the fluency of a sentence.

All of the metrics perform well on the "Re-order Chunks" corruption. METEOR, however, does slightly worse than the other metrics. We believe this to be due to its method of generating an alignment between the words in the candidate and reference sentences. This alignment is computed while minimizing the number of chunks of contiguous and identically ordered tokens in each sentence pair (Chen et al., 2015). Both the original sentence and the corruption contain the same chunks, so it makes sense that METEOR would have more trouble distinguishing between the two than the n-gram based approaches.

7.4 W2V-AVG

The results for the W2V-AVG metric's success on each corruption are shown in Table 7. "Shuffled Chunks" is one of the most interesting corruptions for this metric, because it achieves an accuracy of 0% across the board. The reason for this is that W2V-AVG is a pure bag-of-words model, meaning that word order is entirely ignored, and as a result the model cannot distinguish between the original sentence and its corruption, and so it can never rank the original greater than the corruption.

Surprisingly, we find that W2V-AVG is far less fooled by active-to-passive than most other metrics. Again, we believe that this can be attributed to its bag-of-words approach, which ignores the word order imposed by active and passive voices. Because each version of the sentence will contain nearly all of the same tokens (with the exception of a few "is" and "being" tokens), the two sentence representations are very similar. In a sense, W2V-AVG does well on passive sentences for the wrong reasons - rather than understanding that the semantics are unchanged, it simply observes that most of the words are the same. However, we still see the trend that performance goes down as the number of reference sentences increases.

average word2vec metric			
num references	5	10	20
1. Negated Action	**72.8**	**74.5**	**60.7**
2. Antonym Replacement	91.5	93.0	92.3
3. Negated Subject	**98.2**	**99.1**	**97.8**
4. Active-to-Passive*	**84.9**	**83.6**	**80.3**
5. Synonymous Phrase*	98.3	99.1	98.3
6. DT Substitution*	100.0	100.0	100.0
7. Duplicate PP	87.6	87.6	87.6
8. Remove Head From PP	78.4	82.5	82.5
9. Shuffle Chunks	**00.0**	**00.0**	**00.0**
1. Negated Action*	**100.0**	**100.0**	**100.0**
3. Negated Subject*	**82.8**	**87.3**	**87.1**

Table 7: **Performance of the AVG-W2V metric**. These % accuracies represent the number of successful trials. Numbers referenced in the prose analysis are highlighted in bold. * indicates the scores computed with the *Difference formula*.

Interestingly, we can see that although W2V-AVG achieved 98% accuracy on "Negated Subject", it scored only 75% on "Negated Action". This initially seems quite counter intuitive - either the model should be good at the insertion of a negation word, or it should be bad. The explanation for this reveals a bias in the data itself: in every instance where the "Negated Subject" corruption was applied, the sentence was transformed from "A/The [subject] is" to "There is no [subject]". This is differs from the change in "Negated Action", which is simply the insertion of "not" into the sentence before an action. Because one of these corruptions resulted in 3x more word replacements, the model is able to identify it fairly well.

To confirm this, we added two final entries to Table 7 where we applied the *Difference formula* to the "Negated Subject" and "Negated Action" corruptions to see the fraction of sentence pairs whose scores are within 15% of one another. We found that, indeed, the "Negated Action" corruption scored 100% (meaning that the corruption embeddings were very similar to the original embeddings), while the "Negated Subject" corruption pairs were only similar about 85% of the time. By analyzing these interpretable errors, we can see that stop

words play a larger role than we'd want in our toy metric. To develop a stronger metric, we might change W2V-AVG so that it considers only the content words when computing the centroid embeddings.

8 Conclusion

The main contribution of this work is a novel approach for analyzing evaluation metrics for language generation tasks using Metric Unit TesTs. Not only is this evaluation procedure able to highlight particular metric weaknesses, it also demonstrates results which are far more consistent than correlation with human judgment; a good metric will be able to score well regardless of how noisy the human-derived scores are. Finally, we demonstrate the process of how this analysis can guide the development and strengthening of newly created metrics that are developed.

References

Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. 2010. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Uppsala, Sweden, July. Association for Computational Linguistics.

X. Chen, H. Fang, TY Lin, R. Vedantam, S. Gupta, P. Dollr, and C. L. Zitnick. 2015. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Chin-Yew Lin. 2004. Rouge: a package for automatic evaluation of summaries. pages 25–26.

Matous Machacek and Ondrej Bojar. 2014. Results of the wmt14 metrics shared task. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 293–301, Baltimore, Maryland, USA, June. Association for Computational Linguistics.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 182–190, Montréal, Canada, June. Association for Computational Linguistics.

M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, R. Zamparelli, and Fondazione Bruno Kessler. 2014. A sick cure for the evaluation of compositional distributional semantic models.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *In Proceedings of NIPS*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2001. Bleu: a method for automatic evaluation of machine translation. Technical report, September.

Steven Parker. 2008. Badger: A new machine translation metric.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Matthew G. Snover, Nitin Madnani, Bonnie Dorr, and Richard Schwartz. 2009. Ter-plus: Paraphrase, semantic, and alignment enhancements to translation edit rate.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. In *Advances in Neural Information Processing Systems 24*.

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2014. Cider: Consensus-based image description evaluation. *CoRR*, abs/1411.5726.

N-gram language models for massively parallel devices

Nikolay Bogoychev and **Adam Lopez**
University of Edinburgh
Edinburgh, United Kingdom

Abstract

For many applications, the query speed of N-gram language models is a computational bottleneck. Although massively parallel hardware like GPUs offer a potential solution to this bottleneck, exploiting this hardware requires a careful rethinking of basic algorithms and data structures. We present the first language model designed for such hardware, using B-trees to maximize data parallelism and minimize memory footprint and latency. Compared with a single-threaded instance of KenLM (Heafield, 2011), a highly optimized CPU-based language model, our GPU implementation produces identical results with a smaller memory footprint and a sixfold increase in throughput on a batch query task. When we saturate both devices, the GPU delivers nearly twice the throughput per hardware dollar even when the CPU implementation uses faster data structures.

Our implementation is freely available at `https://github.com/XapaJIaMnu/gLM`

1 Introduction

N-gram language models are ubiquitous in speech and language processing applications such as machine translation, speech recognition, optical character recognition, and predictive text. Because they operate over large vocabularies, they are often a computational bottleneck. For example, in machine translation, Heafield (2013) estimates that decoding a single sentence requires a million language model queries, and Green et al. (2014) estimate that this accounts for more than 50% of decoding CPU time.

To address this problem, we turn to massively parallel hardware architectures, exemplified by general purpose graphics processing units (GPUs), whose memory bandwidth and computational throughput has rapidly outpaced that of CPUs over the last decade (Figure 1). Exploiting this increased power is a tantalizing prospect for any computation-bound problem, so GPUs have begun to attract attention in natural language processing, in problems such as parsing (Canny et al., 2013; Hall et al., 2014), speech recognition (Chong et al., 2009; Chong et al., 2008), and phrase extraction for machine translation (He et al., 2015). As these efforts have shown, it is not trivial to exploit this computational power, because the GPU computational model rewards data parallelism, minimal branching, and minimal access to global memory, patterns ignored by many classic NLP algorithms (Section 2).

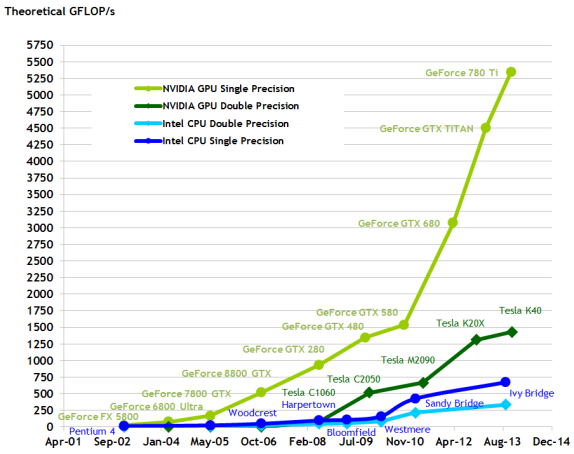


Figure 1: Theoretical floating point performance of CPU and GPU hardware over time (Nvidia Corporation, 2015).

We present the first language model data structure designed for this computational model. Our data structure is a trie in which individual nodes are represented by B-trees, which are searched in parallel (Section 3) and arranged compactly in

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1944–1953,
Berlin, Germany, August 7-12, 2016.

memory (Section 4). Our experiments across a range of parameters in a batch query setting show that this design achieves a throughput six times higher than KenLM (Heafield, 2011), a highly efficient CPU implementation (Section 5). They also show the effects of device saturation and of data structure design decisions.

2 GPU computational model

GPUs and other parallel hardware devices have a different computational profile from widely-used x86 CPUs, so data structures designed for serial models of computation are not appropriate. To produce efficient software for a GPU we must be familiar with its design (Figure 2).

2.1 GPU design

A GPU consists of many simple computational *cores*, which have neither complex caches nor branch predictors to hide latencies. Because they have far fewer circuits than CPU cores, GPU cores are much smaller, and many more of them can fit on a device. So the higher throughput of a GPU is due to the sheer number of cores, each executing a single *thread* of computation (Figure 2). Each core belongs to a *Streaming Multiprocessor (SM)*, and all cores belonging to a SM must execute the same instruction at each time step, with exceptions for branching described below. This execution model is very similar to single instruction, multiple data (SIMD) parallelism.[1]

Computation on a GPU is performed by an inherently parallel function or *kernel*, which defines a *grid* of data elements to which it will be applied, each processed by a *block* of parallel threads. Once scheduled, the kernel executes in parallel on all cores allocated to blocks in the grid. At minimum, it is allocated to a single *warp*—32 cores on our experimental GPU. If fewer cores are requested, a full warp is still allocated, and the unused cores idle.

A GPU offers several memory types, which differ in size and latency (Table 1). Unlike a CPU program, which can treat memory abstractly, a GPU program must explicitly specify in which physical memory each data element resides. This choice has important implications for efficiency that entail design tradeoffs, since memory closer to a core is small and fast, while memory further away is large and slow (Table 1).

[1]Due to differences in register usage and exceptions for branching, this model is not pure SIMD. Nvidia calls it SIMT (single instruction, multiple threads).

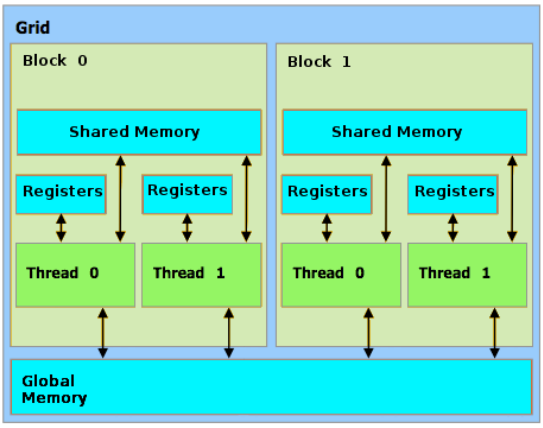


Figure 2: GPU memory hierarchy and computational model (Nvidia Corporation, 2015).

Memory type	Latency	Size
Register	0	4B
Shared	4–8	16KB–96KB
Global GPU	200–800	2GB–12GB
CPU	10K+	16GB–1TB

Table 1: Latency (in clock cycles) and size of different GPU memory types. Estimates are adapted from Nvidia Corporation (2015) and depend on several aspects of hardware configuration.

2.2 Designing efficient GPU algorithms

To design an efficient GPU application we must observe the constraints imposed by the hardware, which dictate several important design principles.

Avoid branching instructions. If a branching instruction occurs, threads that meet the branch condition run while the remainder idle (a *warp divergence*). When the branch completes, threads that don't meet the condition run while the first group idles. So, to maximize performance, code must be designed with little or no branching.

Use small data structures. Total memory on a state-of-the-art GPU is 12GB, expected to rise to 24GB in the next generation. Language models that run on CPU frequently exceed these sizes, so our data structures must have the smallest possible memory footprint.

Minimize global memory accesses. Data in the CPU memory must first be transferred to the device. This is very slow, so data structures must reside in GPU memory. But even when they

Data structure	Size	Query speed	Ease of backoff	Construction time	Lossless
Trie (Heafield, 2011)	Small	Fast	Yes	Fast	Yes
Probing hash table (Heafield, 2011)	Larger	Faster	Yes	Fast	Yes
Double array (Yasuhara et al., 2013)	Larger	Fastest	Yes	Very slow	Yes
Bloom filter (Talbot and Osborne, 2007)	Small	Slow	No	Fast	No

Table 2: A survey of language model data structures and their computational properties.

reside in global GPU memory, latency is high, so wherever possible, data should be accessed from shared or register memory.

Access memory with coalesced reads. When a thread requests a byte from global memory, it is copied to shared memory along with many surrounding bytes (between 32 and 128 depending on the architecture). So, if consecutive threads request consecutive data elements, the data is copied in a single operation (a *coalesced read*), and the delay due to latency is incurred only once for all threads, increasing throughput.

3 A massively parallel language model

Let w be a sentence, w_i its ith word, and N the order of our model. An N-gram language model defines the probability of w as:

$$P(w) = \prod_{i=1}^{|w|} P(w_i|w_{i-1}...w_{i-N+1}) \quad (1)$$

A backoff language model (Chen and Goodman, 1999) is defined in terms of n-gram probabilities $P(w_i|w_{i-1}...w_{i-n+1})$ for all n from 1 to N, which are in turn defined by n-gram parameters $\hat{P}(w_i...w_{i-n+1})$ and backoff parameters $\beta(w_{i-1}...w_{i-n+1})$. Usually $\hat{P}(w_i...w_{i-n+1})$ and $\beta(w_{i-1}...w_{i-n+1})$ are probabilities conditioned on $w_{i-1}...w_{i-n+1}$, but to simplify the following exposition, we will simply treat them as numeric parameters, each indexed by a reversed n-gram. If parameter $\hat{P}(w_i...w_{i-n+1})$ is nonzero, then:

$$P(w_i|w_{i-1}...w_{i-n+1}) = \hat{P}(w_i...w_{i-n+1})$$

Otherwise:

$$P(w_i|w_{i-1}...w_{i-n+1}) = P(w_i|w_{i-1}...w_{i-n+2}) \times \beta(w_{i-1}...w_{i-n+1})$$

This recursive definition means that the probability $P(w_i|w_{i-1}...w_{i-N+1})$ required for Equation 1 may depend on multiple parameters. If r ($< N$) is the largest value for which $\hat{P}(w_i|w_{i-1}...w_{i-r+1})$ is nonzero, then we have:

$$P(w_i|w_{i-1}...w_{i-N+1}) = \hat{P}(w_i...w_{i-r+1}) \prod_{n=r+1}^{N} \beta(w_{i-1}...w_{i-n+1}) \quad (2)$$

Our data structure must be able to efficiently access these parameters.

3.1 Trie language models

With this computation in mind, we surveyed several popular data structures that have been used to implement N-gram language models on CPU, considering their suitability for adaptation to GPU (Table 2). Since a small memory footprint is crucial, we implemented a variant of the trie data structure of Heafield (2011). We hypothesized that its slower query speed compared to a probing hash table would be compensated for by the throughput of the GPU, a question we return to in Section 5.

A trie language model exploits two important guarantees of backoff estimators: first, if $\hat{P}(w_i...w_{i-n+1})$ is nonzero, then $\hat{P}(w_i...w_{i-m+1})$ is also nonzero, for all $m < n$; second, if $\beta(w_{i-1}...w_{i-n+1})$ is one, then $\beta(w_{i-1}...w_{i-p+1})$ is one, for all $p > n$. Zero-valued n-gram parameters and one-valued backoff parameters are not explicitly stored. To compute $P(w_i|w_{i-1}...w_{i-N+1})$, we iteratively retrieve $\hat{P}(w_i...w_{i-m+1})$ for increasing values of m until we fail to find a match, with the final nonzero value becoming $\hat{P}(w_i...w_{i-r+1})$ in Equation 2. We then iteratively retrieve $\beta(w_{i-1}...w_{i-n+1})$ for increasing values of n starting from $r+1$ and continuing until $n = N$ or we fail to find a match, multiplying all retrieved terms to compute $P(w_i|w_{i-1}...w_{i-N+1})$ (Equation 2). The trie is designed to execute these iterative parameter retrievals efficiently.

Let Σ be a our vocabulary, Σ^n the set of all n-grams over the vocabulary, and $\Sigma^{[N]}$ the

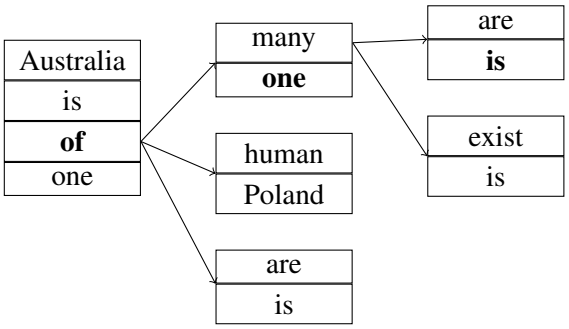


Figure 3: Fragment of a trie showing the path of N-gram **is one of** in bold. A query for the N-gram **every one of** traverses the same path, but since **every** is not among the keys in the final node, it returns the n-gram parameter $\hat{P}$(**of**|**one**) and returns to the root to seek the backoff parameter β(**every one**). Based on image from Federico et al. (2008).

set $\Sigma^1 \cup ... \cup \Sigma^N$. Given an n-gram *key* $w_i...w_{i-n+1} \in \Sigma^{[N]}$, our goal is to retrieve *value* $\langle \hat{P}(w_i...w_{i-n+1}), \beta(w_i...w_{i-n+1}) \rangle$. We assume a bijection from Σ to integers in the range $1, ..., |\Sigma|$, so in practice all keys are sequences of integers.

When $n = 1$, the set of all possible keys is just Σ. For this case, we can store keys with nontrivial values in a sorted array A and their associated values in an array V of equal length so that $V[j]$ is the value associated with key $A[j]$. To retrieve the value associated with key k, we seek j for which $A[j] = k$ and return $V[j]$. Since A is sorted, j can be found efficiently with binary or interpolated search (Figure 4).

When $n > 1$, queries are recursive. For $n < N$, for every $w_i...w_{i-n+1}$ for which $\hat{P}(w_i...w_{i-n+1}) > 0$ or $\beta(w_i...w_{i-n+1}) < 1$, our data structure contains associated arrays $K_{w_i...w_{i-n+1}}$ and $V_{w_i...w_{i-n+1}}$. When key k is located in $A_{w_i...w_{i-n+1}}[j]$, the value stored at $V_{w_i...w_{i-n+1}}[j]$ includes the address of arrays $A_{w_i...w_{i-n+1}k}$ and $V_{w_i...w_{i-n+1}k}$. To find the values associated with an n-gram $w_i...w_{i-n+1}$, we first search the root array A for j_1 such that $A[j_1] = w_i$. We retrieve the address of A_{w_i} from $V[j_1]$, and we then search for j_2 such that $A_{w_i}[j_2] = w_{i-1}$. We continue to iterate this process until we find the value associated with the longest suffix of our n-gram stored in the trie. We therefore iteratively retrieve the parameters needed to compute Equation 2, returning to the root exactly once if backoff parameters are required.

3.1.1 K-ary search and B-trees

On a GPU, the trie search algorithm described above is not efficient because it makes extensive use of binary search, an inherently serial algorithm. However, there is a natural extension of binary search that is well-suited to GPU: K-ary search (Hwu, 2011). Rather than divide an array in two as in binary search, K-ary search divides it into K equal parts and performs $K - 1$ comparisons simultaneously (Figure 5).

To accommodate large language models, the complete trie must reside in global memory, and in this setting, K-ary search on an array is inefficient, since the parallel threads will access non-consecutive memory locations. To avoid this, we require a data structure that places the K elements compared by K-ary search in consecutive memory locations so that they can be copied from global to shared memory with a coalesced read. This data structure is a B-tree (Bayer and McCreight, 1970), which is widely used in databases, filesystems and information retrieval.

Informally, a B-tree generalizes binary trees in exactly the same way that K-ary search generalizes binary search (Figure 6). More formally, a B-tree is a recursive data structure that replaces arrays A and V at each node of the trie. A B-tree node of size K consists of three arrays: a 1-indexed array B of $K - 1$ keys; a 1-indexed array V of $K - 1$ associated values so that $V[j]$ is the value associated with key $B[j]$; and, if the node is not a leaf, a 0-indexed array C of K addresses to child B-trees. The keys in B are sorted, and the subtree at address pointed to by child $C[j]$ represents only key-value pairs for keys between $B[j]$ and $B[j + 1]$ when $1 \leq j < K$, keys less than $B[1]$ when $j = 0$, or keys greater than $B[K]$ when $j = K$.

To find a key k in a B-tree, we start at the root node, and we seek j such that $B[j] \leq k < B[j + 1]$. If $B[j] = k$ we return $V[j]$, otherwise if the node is not a leaf node we return the result of recursively querying the B-tree node at the address $C[j]$ ($C[0]$ if $k < B[1]$ or $C[K]$ if $k > B[K]$). If the key is not found in array B of a leaf, the query fails.

Our complete data structure is a trie in which each node except the root is a B-tree (Figure 7). Since the root contains all possible keys, its keys are simply represented by an array A, which can be indexed in constant time without any search.

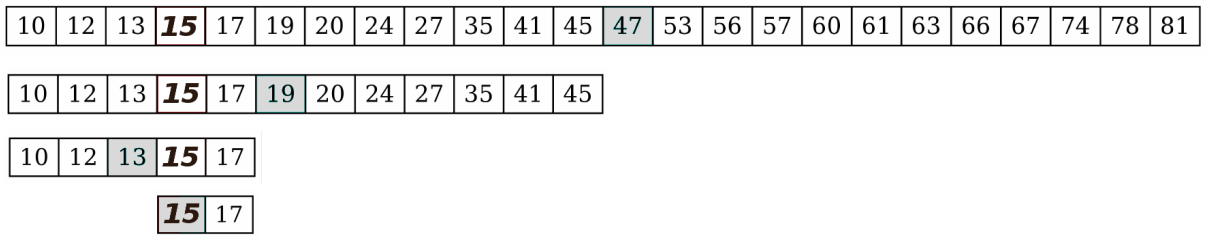


Figure 4: Execution of a binary search for key 15. Each row represents a time step and highlights the element compared to the key. Finding key 15 requires four time steps and four comparisons.

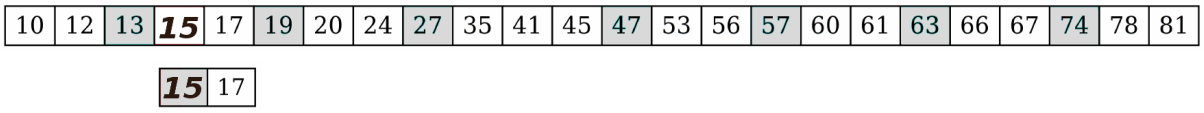


Figure 5: Execution of K-ary search with the same input as Figure 4, for $K = 8$. The first time step executes seven comparisons in parallel, and the query is recovered in two time steps.

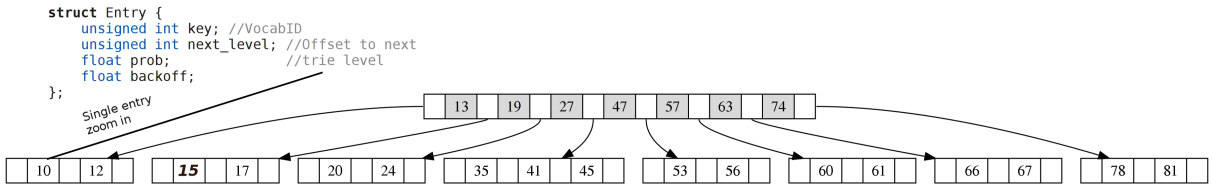


Figure 6: In a B-tree, the elements compared in K-ary search are consecutive in memory. We also show the layout of an individual entry.

4 Memory layout and implementation

Each trie node represents a unique n-gram $w_i...w_{i-n+1}$, and if a B-tree node within the trie node contains key w_{i-n}, then it must also contain the associated values $\hat{P}(w_i...w_{i-n})$, $\beta(w_i...w_{i-n})$, and the address of the trie node representing $w_i...w_{i-n}$ (Figure 6, Figure 3). The entire language model is laid out in memory as a single byte array in which trie nodes are visited in breadth-first order and the B-tree representation of each node is also visited in breadth-first order (Figure 7).

Since our device has a 64-bit architecture, pointers can address 18.1 exabytes of memory, far more than available. To save space, our data structure does not store global addresses; it instead stores the difference in addresses between the parent node and each child. Since the array is aligned to four bytes, these relative addresses are divided by four in the representation, and multiplied by four at runtime to obtain the true offset. This enables us to encode relative addresses of 16GB, still larger than the actual device memory. We estimate that relative addresses of this size allow us to store a model containing around one billion n-grams.[2] Unlike CPU language model implementations such as those of Heafield (2011) and Watanabe et al. (2009), we do not employ further compression techniques such as variable-byte encoding or LOUDS, because their runtime decompression algorithms require branching code, which our implementation must avoid.

We optimize the node representation for coalesced reads by storing the keys of each B-tree consecutively in memory, followed by the corresponding values, also stored consecutively (Figure 6). When the data structure is traversed, only key arrays are iteratively copied to shared memory until a value array is needed. This design minimizes the number of reads from global memory.

4.1 Construction

The canonical B-tree construction algorithm (Cormen et al., 2009) produces nodes that are not fully saturated, which is desirable for B-trees that

[2] We estimate this by observing that a model containing 423M n-grams takes 3.8Gb of memory, and assuming an approximately linear scaling, though there is some variance depending on the distribution of the n-grams.

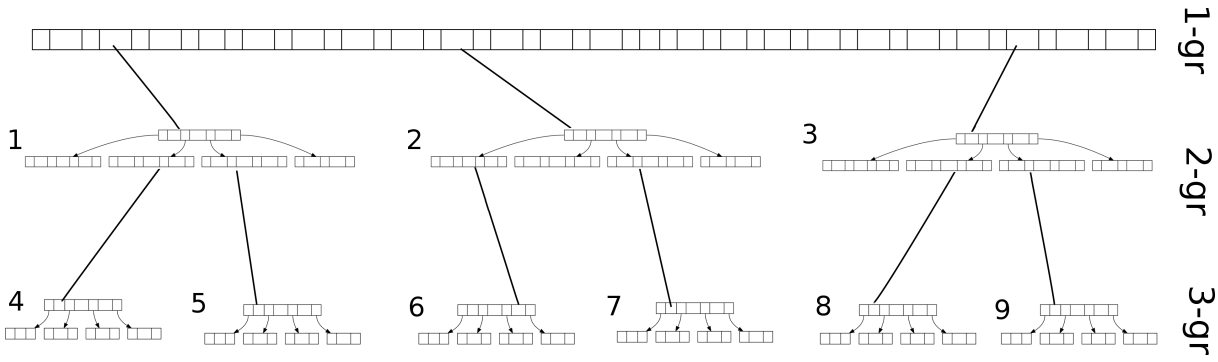


Figure 7: Illustration of the complete data structure, showing a root trie node as an array representing unigrams, and nine B-trees, each representing a single trie node. The trie nodes are numbered according to the order in which they are laid out in memory.

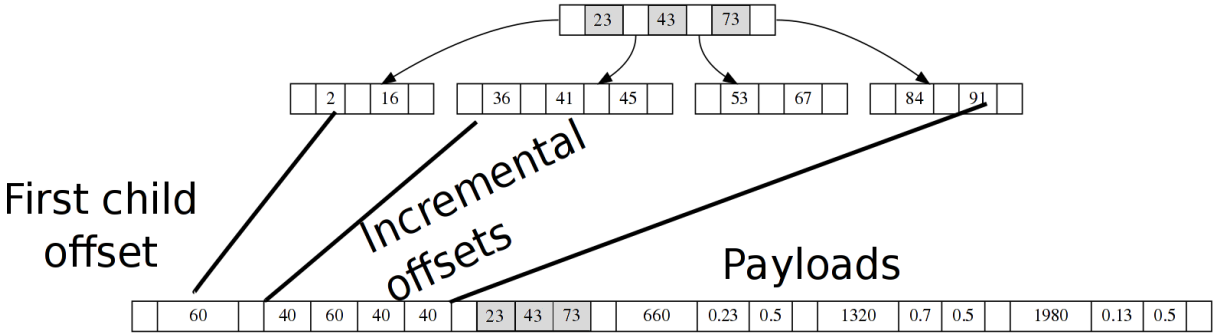


Figure 8: Layout of a single B-tree node for $K = 4$. Relative addresses of the four child B-tree nodes (array C) are followed by three keys (array B), and three values (array V), each consisting of an n-gram probability, backoff, and address of the child trie node.

support insertion. However, our B-trees are immutable, and unsaturated nodes of unpredictable size lead to underutilization of threads, warp divergence, and deeper trees that require more iterations to query. So, we use a construction algorithm inspired by Cesarini and Soda (1983) and Rosenberg and Snyder (1981). It is implemented on CPU, and the resulting array is copied to GPU memory to perform queries.

Since the entire set of keys and values is known in advance for each n-gram, our construction algorithm receives them in sorted order as the array A described in Section 3.1. The procedure then splits this array into K consecutive subarrays of equal size, leaving $K-1$ individual keys between each subarray.[3] These $K-1$ keys become the keys of the root B-tree. The procedure is then applied recursively to each subarray. When applied to an array whose size is less than K, the algorithm returns a leaf node. When applied to an array whose size is greater than or equal to K but less than $2K$, it splits the array into a node containing the first $K-1$ keys, and a single leaf node containing the remaining keys, which becomes a child of the first.

4.2 Batch queries

To fully saturate our GPU we execute many queries simultaneously. A grid receives the complete set of N-gram queries and each block processes a single query by performing a sequence of K-ary searches on B-tree nodes.

5 Experiments

We compared our open-source GPU language model **gLM** with the CPU language model KenLM (Heafield, 2011).[4][5] KenLM can use two quite different language model data structures: a fast probing hash table, and a more compact but slower trie, which inspired our own language model design. Except where noted, our B-tree

[3] Since the size of the array may not be exactly divisible by K, some subarrays may differ in length by one.

[4] https://github.com/XapaJIaMnu/gLM

[5] https://github.com/kpu/kenlm/commit/9495443

node size $K = 31$, and we measure throughput in terms of query speed, which does not include the cost of initializing or copying data structures, or the cost of moving data to or from the GPU.

We performed our GPU experiments on an Nvidia Geforce GTX, a state-of-the-art GPU, released in the first quarter of 2015 and costing 1000 USD. Our CPU experiments were performed on two different devices: one for single-threaded tests and one for multi-threaded tests. For the single-threaded CPU tests, we used an Intel Quad Core i7 4720HQ CPU released in the first quarter of 2015, costing 280 USD, and achieving 85% of the speed of a state-of-the-art consumer-grade CPU when single-threaded. For the multi-threaded CPU tests we used two Intel Xeon E5-2680 CPUs, offering a combined 16 cores and 32 threads, costing at the time of their release 3,500 USD together. Together, their performance specifications are similar to the recently released Intel Xeon E5-2698 v3 (16 cores, 32 threads, costing 3,500USD). The different CPU configurations are favorable to the CPU implementation in their tested condition: the consumer-grade CPU has higher clock speeds in single-threaded mode than the professional-grade CPU; while the professional-grade CPUs provide many more cores (though at lower clock speeds) when fully saturated. Except where noted, CPU throughput is reported for the single-threaded condition.

Except where noted, our language model is the Moses 3.0 release English 5-gram language model, containing 88 million n-grams.[6] Our benchmark task computes perplexity on data extracted from the Common Crawl dataset used for the 2013 Workshop on Machine Translation, which contains 74 million words across 3.2 million sentences.[7] Both gLM and KenLM produce identical perplexities, so we are certain that our implementation is correct. Except where noted, the faster KenLM Probing backend is used. The perplexity task has been used as a basic test of other language model implementations (Osborne et al., 2014; Heafield et al., 2015).

5.1 Query speed

When compared to single-threaded KenLM, our results (Table 3) show that gLM is just over six times faster than the fast probing hash table, and nearly fifteen times faster than the trie data structure, which is quite similar to our own, though slightly smaller due to the use of compression. The raw speed of the GPU is apparent, since we were able to obtain our results with a relatively short engineering effort when compared to that of KenLM, which has been optimized over several years.

LM (threads)	Throughput	Size (GB)
KenLM probing (1)	10.3M	1.8
KenLM probing (16)	49.8M	1.8
KenLM probing (32)	**120.4M**	1.8
KenLM trie (1)	4.5M	**0.8**
gLM	65.5M	1.2

Table 3: Comparison of gLM and KenLM on throughput (N-gram queries per second) and data structure size.

When we fully saturate our professional-grade CPU, using all sixteen cores and sixteen hyper-threads, KenLM is about twice as fast as gLM. However, our CPU costs nearly four times as much as our GPU, so economically, this comparison favors the GPU.

On first glance, the scaling from one to sixteen threads is surprisingly sublinear. This is not due to vastly different computational power of the individual cores, which are actually very similar. It is instead due to scheduling, cache contention, and—most importantly—the fact that our CPUs implement *dynamic overclocking*: the base clock rate of 2.7 GHz at full saturation increases to 3.5 GHz when the professional CPU is underutilized, as when single-threaded; the rates for the consumer-grade CPU similarly increase from 2.6 to 3.6 GHz.[8]

5.2 Effect of B-tree node size

What is the optimal K for our B-tree node size? We hypothesized that the optimal size would be one that approaches the size of a coalesced memory read, which should allow us to maximize parallelism while minimizing global memory accesses and B-tree depth. Since the size of a coalesced read is 128 bytes and keys are four bytes, we hypothesized that the optimal node size would be around $K = 32$, which is also the size of a warp. We tested this by running experiments

[6] http://www.statmt.org/moses/RELEASE-3.0/models/fr-en/lm/europarl.lm.1

[7] http://www.statmt.org/wmt13/training-parallel-commoncrawl.tgz

[8] Intel calls this Intel Turbo Boost.

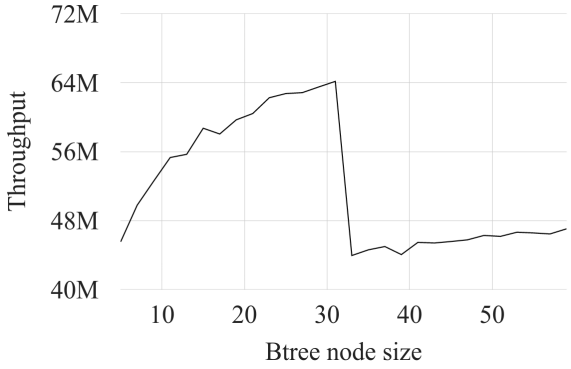


Figure 9: Effect of BTree node size on throughput (ngram queries per second)

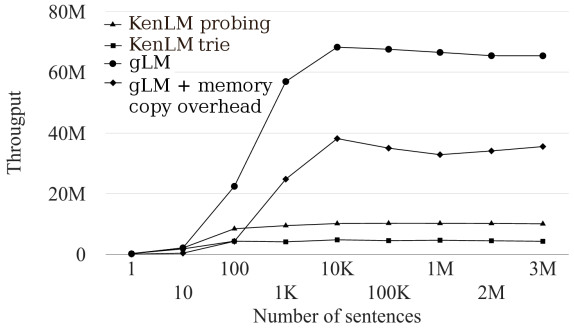


Figure 10: Throughput (N-gram queries per second) vs. batch size for gLM, KenLM probing, and KenLM trie.

	Regular LM	Big LM
KenLM	10.2M	8.2M
KenLM Trie	4.5M	3.0M
gLM	**65.5M**	**55M**

Table 4: Throughput comparison (ngram queries per second) between **gLM** and **KenLM** with a 5 times larger model and a regular language model.

that varied K from 5 to 59, and the results (Figure 9) confirmed our hypothesis. As the node size increases, throughput increases until we reach a node size of 33, where it steeply drops. This result highlights the importance of designing data structures that minimize global memory access and maximize parallelism.

We were curious about what effect this node size had on the depth of the B-trees representing each trie node. Measuring this, we discovered that for bigrams, 88% of the trie nodes have a depth of one—we call these *B-stumps*, and they can be exhaustively searched in a single parallel operation. For trigrams, 97% of trie nodes are B-stumps, and for higher order n-grams the percentage exceeds 99%.

5.3 Saturating the GPU

A limitation of our approach is that it is only effective in high-throughput situations that continually saturate the GPU. In situations where a language model is queried only intermittently or only in short bursts, a GPU implementation may not be useful. We wanted to understand the point at which this saturation occurs, so we ran experiments varying the batch size sent to our language model, comparing its behavior with that of KenLM. To understand situations in which the GPU hosts the language model for query by an external GPU, we measure query speed with and without the cost of copying queries to the device.

Our results (Figure 10) suggest that the device is nearly saturated once the batch size reaches a thousand queries, and fully saturated by ten thousand queries. Throughput remains steady as batch size increases beyond this point. Even with the cost of copying batch queries to GPU memory, throughput is more than three times higher than that of single threaded KenLM. We have not included results of multi-threaded KenLM scaling on Figure 10 but they are similar to the single-threaded case: throughput (as shown on Table 3) plateaus at around one hundred sentences per thread.

5.4 Effect of model size

To understand the effect of model size on query speed, we built a language model with 423 million n-grams, five times larger than our basic model. The results (Table 4) show an 18% slowdown for gLM and 20% slowdown for KenLM, showing that model size affects both implementations similarly.

5.5 Effect of N-gram order on performance

All experiments so far use an N-gram order of five. We hypothesized that lowering the ngram order of the model would lead to faster query time (Table 5). We observe that N-gram order affects throughput of the GPU language model much more than the CPU one. This is likely due to effects of backoff queries, which are more optimized in KenLM. At higher orders, more backoff queries occur, which reduces throughput for gLM.

	5-gram	4-gram	3-gram
KenLM	10.2M	9.8M	11.5M
KenLM Trie	4.5M	4.5M	5.2M
gLM	**65.5M**	**71.9M**	**93.7M**

Table 5: Throughput comparison (ngram queries per second) achieved using lower order ngram models.

5.6 Effect of templated code

Our implementation initially relied on hard-coded values for parameters such as B-tree node size and N-gram order, which we later replaced with parameters. Surprisingly, we observed that this led to a reduction in throughput from 65.6 million queries per second to 59.0 million, which we traced back to the use of dynamically allocated shared memory, as well as compiler optimizations that only apply to compile-time constants. To remove this effect, we heavily templated our code, using as many compile-time constants as possible, which improves throughput but enables us to change parameters through recompilation.

5.7 Bottlenecks: computation or memory?

On CPU, language models are typically memory-bound: most cycles are spent in random memory accesses, with little computation between accesses. To see if this is true in gLM we experimented with two variants of the benchmark in Figure 3: one in which the GPU core was underclocked, and one in which the memory was underclocked. This effectively simulates two variations in our hardware: A GPU with slower cores but identical memory, and one with slower memory, but identical processing speed. We found that throughput decreases by about 10% when underclocking the cores by 10%. On the other hand, underclocking memory by 25% reduced throughput by 1%. We therefore conclude that gLM is computation-bound. We expect that gLM will continue to improve on parallel devices offering higher theoretical floating point performance.

6 Conclusion

Our language model is implemented on a GPU, but its general design (and much of the actual code) is likely to be useful to other hardware that supports SIMD parallelism, such as the Xeon Phi. Because it uses batch processing, our on-chip language model could be integrated into a machine translation decoder using strategies similar to those used to integrate an on-network language model nearly a decade ago (Brants et al., 2007). An alternative method of integration would be to move the decoder itself to GPU. For phrase-based translation, this would require a translation model and dynamic programming search algorithm on GPU. Translation models have been implemented on GPU by He et al. (2015), while related search algorithms for (Chong et al., 2009; Chong et al., 2008) and parsing (Canny et al., 2013; Hall et al., 2014) have been developed for GPU. We intend to explore these possibilities in future work.

Acknowledgements


This work was conducted within the scope of the Horizon 2020 Innovation Action *Modern MT*, which has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 645487.

We thank Kenneth Heafield, Ulrich Germann, Rico Sennrich, Hieu Hoang, Federico Fancellu, Nathan Schneider, Naomi Saphra, Sorcha Gilroy, Clara Vania and the anonymous reviewers for productive discussion of this work and helpful comments on previous drafts of the paper. Any errors are our own.

References

R. Bayer and E. McCreight. 1970. Organization and maintenance of large ordered indices. In *Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*, SIGFIDET '70, pages 107–141.

T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean. 2007. Large language models in machine translation. In *In Proceedings of EMNLP-CoNLL*.

J. Canny, D. Hall, and D. Klein. 2013. A multi-teraflop constituency parser using GPUs. In *Proceedings of EMNLP*.

F. Cesarini and G. Soda. 1983. An algorithm to construct a compact B-tree in case of ordered keys. *Information Processing Letters*, 17(1):13–16.

S. F. Chen and J. Goodman. 1999. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 13(4):359–393.

J. Chong, Y. Yi, A. Faria, N. R. Satish, and K. Keutzer. 2008. Data-parallel large vocabulary continuous

speech recognition on graphics processors. Technical Report UCB/EECS-2008-69, EECS Department, University of California, Berkeley, May.

J. Chong, E. Gonina, Y. Yi, and K. Keutzer. 2009. A fully data parallel WFST-based large vocabulary continuous speech recognition on a graphics processing unit. In *Proceedings of Interspeech*.

T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2009. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

M. Federico, N. Bertoldi, and M. Cettolo. 2008. IRSTLM: an open source toolkit for handling large scale language models. In *Proceedings of Interspeech*, pages 1618–1621. ISCA.

S. Green, D. Cer, and C. Manning. 2014. Phrasal: A toolkit for new directions in statistical machine translation. In *Proceedings of WMT*.

D. Hall, T. Berg-Kirkpatrick, and D. Klein. 2014. Sparser, better, faster GPU parsing. In *Proceedings of ACL*.

H. He, J. Lin, and A. Lopez. 2015. Gappy pattern matching on GPUs for on-demand extraction of hierarchical translation grammars. *TACL*, 3:87–100.

K. Heafield, R. Kshirsagar, and S. Barona. 2015. Language identification and modeling in specialized hardware. In *Proceedings of ACL-IJCNLP*, July.

K. Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of WMT*, pages 187–197, July.

K. Heafield. 2013. *Efficient Language Modeling Algorithms with Applications to Statistical Machine Translation*. Ph.D. thesis, Carnegie Mellon University, September.

W.-m. W. Hwu. 2011. *GPU Computing Gems Emerald Edition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition.

Nvidia Corporation. 2015. *Nvidia CUDA Compute Unified Device Architecture Programming Guide*. Nvidia Corporation. https://docs.nvidia.com/cuda/cuda-c-programming-guide/.

M. Osborne, A. Lall, and B. V. Durme. 2014. Exponential reservoir sampling for streaming language models. In *Proceedings of ACL*, pages 687–692.

A. L. Rosenberg and L. Snyder. 1981. Time- and space-optimality in B-trees. *ACM Trans. Database Syst.*, 6(1):174–193, Mar.

D. Talbot and M. Osborne. 2007. Smoothed Bloom filter language models: Tera-scale LMs on the cheap. In *Proceedings of EMNLP-CoNLL*, pages 468–476.

T. Watanabe, H. Tsukada, and H. Isozaki. 2009. A succinct N-gram language model. In *Proc. of ACL-IJCNLP*.

M. Yasuhara, T. Tanaka, J. ya Norimatsu, and M. Yamamoto. 2013. An efficient language model using double-array structures. In *EMNLP*, pages 222–232.

Cross-Lingual Morphological Tagging for Low-Resource Languages

Jan Buys
Department of Computer Science
University of Oxford
jan.buys@cs.ox.ac.uk

Jan A. Botha
Google Inc.
London
jabot@google.com

Abstract

Morphologically rich languages often lack the annotated linguistic resources required to develop accurate natural language processing tools. We propose models suitable for training morphological taggers with rich tagsets for low-resource languages without using direct supervision. Our approach extends existing approaches of projecting part-of-speech tags across languages, using bitext to infer constraints on the possible tags for a given word type or token. We propose a tagging model using Wsabie, a discriminative embedding-based model with rank-based learning. In our evaluation on 11 languages, on average this model performs on par with a baseline weakly-supervised HMM, while being more scalable. Multilingual experiments show that the method performs best when projecting between related language pairs. Despite the inherently lossy projection, we show that the morphological tags predicted by our models improve the downstream performance of a parser by +0.6 LAS on average.

1 Introduction

Morphologically rich languages pose significant challenges for Natural Language Processing (NLP) due to data-sparseness caused by large vocabularies. Intermediate processing is often required to address the limitations of only using surface forms, especially for small datasets. Common morphological processing tasks include segmentation (Creutz and Lagus, 2007; Snyder and Barzilay, 2008), paradigm learning (Durrett and DeNero, 2013; Ahlberg et al., 2015) and morphological tagging (Müller and Schuetze, 2015). In this paper we focus on the latter.

Parts-of-speech (POS) tagging is the most common form of syntactic annotation. However, the granularity of POS varies across languages and annotation-schemas, and tagsets have often been extended to include tags for morphologically-marked properties such as number, case or degree. To enable cross-lingual learning, a small set of universal (coarse-grained) POS tags have been proposed (Petrov et al., 2012). For morphological processing this can be complemented with a set of attribute-feature values that makes the annotation more fine-grained (Zeman, 2008; Sylak-Glassman et al., 2015b).

Tagging text with morphologically-enriched labels has been shown to benefit downstream tasks such as parsing (Tsarfaty et al., 2010) and semantic role labelling (Hajič et al., 2009). In generation tasks such as machine translation these tags can help to generate the right form of a word and to model agreement (Toutanova et al., 2008). Morphological information can also benefit automatic speech recognition for low-resource languages (Besacier et al., 2014).

However, annotating sufficient data to learn accurate morphological taggers is expensive and relies on linguistic expertise, and is therefore currently only feasible for the world's most widely-used languages. In this paper we are interested in learning morphological taggers without the availability of supervised data. A successful paradigm for learning without direct supervision is to make use of word-aligned parallel text, with a resource-rich language on one side and a resource-poor language on the other side (Yarowsky et al., 2001; Fossum and Abney, 2005; Das and Petrov, 2011; Täckström et al., 2013).

In this paper we extend these methods, that have mostly been proposed for universal POS-taggers, to learn weakly-supervised morphological taggers.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1954–1964,
Berlin, Germany, August 7-12, 2016.

Our approach is based on projecting token and type constraints across parallel text, learning a tagger in a weakly-supervised manner from the projected constraints (Täckström et al., 2013). We propose an embedding-based model trained with the Wsabie algorithm (Weston et al., 2011), and compare this approach against a baseline HMM model.

We evaluate the projected tags for a set of languages for which morphological tags are available in the Universal Dependency corpora. To show the feasibility of our approach, and to compare the performance of different models, we use English as source language. Then we perform an evaluation on all language pairs in the set of target languages which shows that the best performance is obtained when projecting between genealogically related languages.

As an extrinsic evaluation of our approach, we show that NLP models can benefit from using these induced tags even if they are not as accurate as tags produced by supervised models, by evaluating the effect of features obtained from tags predicted by the induced morphological taggers in dependency parsing.

2 Universal Morphological Tags

In order to do cross-lingual learning we require a common morphological tagset. To evaluate these models we require datasets in multiple languages which have been annotated with such a consistent schema. The treebanks annotated in the Universal Dependencies (UD) project (de Marneffe et al., 2014) are suitable for this purpose.

All the data is annotated with universal POS tags, a set of 17 tags[1]. We use UD v1.2 (Nivre et al., 2015), which contain 25 languages annotated with morphological attributes (called features). In addition to POS, there are 17 universal *attributes*, which each takes one of a set of *values* when annotated. The morphological tag of a token denotes the union of its morphological attribute-value pairs, including its POS.

Although the schema is consistent across languages, there are language-specific phenomena and considerations that result in some mismatches for a given pair of languages. One source of this is that the UD treebanks were mostly constructed by fully or semi-automatic conversion of existing treebanks which had used different annotation schemes. Furthermore, not all the attributes and values appear in all languages (e.g. additional cases in morphologically-rich languages such as Finnish), and there are still a number of language-specific tags not in the universal schema. Finally, in some instances properties that are not realised in the surface word form are absent from the annotation (e.g. in English the person and number of verbs are only annotated for third-person singular, as there are no distinct morphological forms for their other values).

An example of the morphological annotation employed is given in Figure 1. Note that the annotations for aligned word-pairs are not fully consistent. Some attributes appear only in the English treebank (e.g. Voice), while others appear only in the Dutch treebank (e.g. Aspect, Subcat).

3 Tag Projection across Bitext

Our approach to train morphological taggers is based on the paradigm of projecting token and type constraints as proposed by Täckström et al. (2013). The training data consist of parallel text with the resource-rich language on the source-side and the low-resource language on the target side. The source-side text is tagged with a supervised morphological tagger. For every target-side sentence, the type and token constraints are used to construct a set of permitted tags for each token in the sentence. These constraints will then be used to train morphological taggers.

3.1 Type and token constraints

To extract constraints from the parallel text, we first obtain bidirectional word alignments. To ensure high quality alignments, alignment pairs with a confidence below a fixed threshold α are removed. The motivation for using only high-confidence alignments is that incorrect alignments will hurt the performance of the model, while it is easier to use more parallel text to obtain a sufficient number of alignments for training.

The first class of constraints that we extract from the parallel text is type constraints. For each word type, we construct a distribution over tags for the word by accumulating counts of the morphological tags of source-side tokens that are aligned to instances of the word type. The set of tags with probability above some threshold β is taken as the tag dictionary entry for that word type. To

[1]This extends, but is not fully consistent with, the set of 12 tags proposed by Petrov et al. (2012).

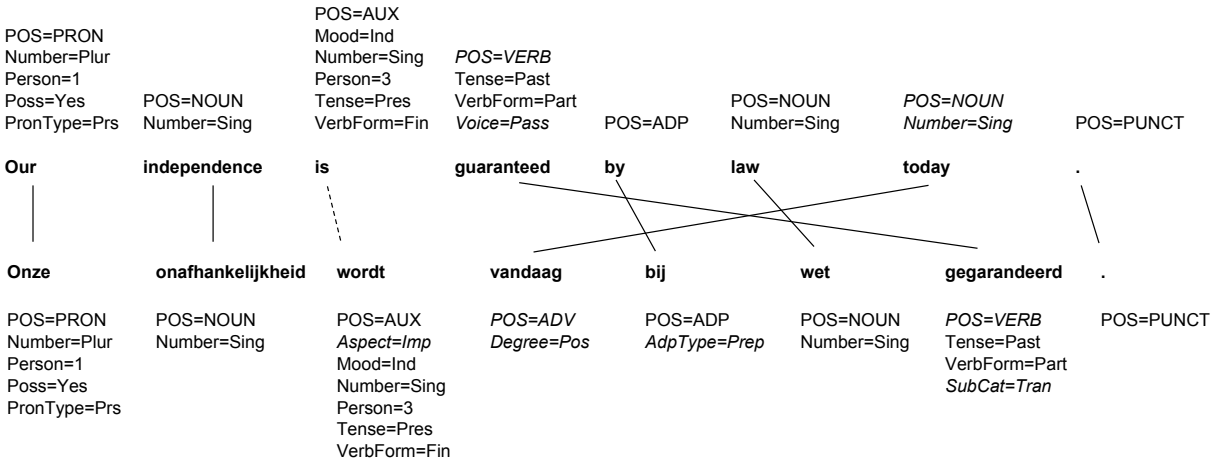


Figure 1: A parallel sentence in English and Dutch annotated with universal morphological tags, showing high-confidence automatic word-alignments. Attribute-value pairs that occur only on one side of an aligned pair of tokens are indicated in *italics*. The dashed line indicates a low-confidence alignment point, which is ignored in our projection method.

construct the training examples, each token whose type occurs in the tag dictionary is restricted to the set of tags in the dictionary entry. For tokens for which the dictionary entry is empty, all the tags are included in the set of permitted tags (this happens when the tag distribution is too flat and all the probabilities are below the threshold). In principle, type constraints can also be obtained from an external dictionary, but in this paper we assume we do not have such a resource.

The second class of constraints places restrictions on word tokens. Every target token is constrained to the tag of its aligned source token, while unaligned tokens can take any tag.

Token constraints are combined with type constraints as proposed by Täckström et al. (2013): If a token is unaligned, its type constraints are used. If the token is aligned, and there is no dictionary entry for the token type, the token constraint is used. If there is a dictionary entry for the token type, and the token constraint tag is in the dictionary, the token constraint is used. If the token constraint tag is not in the dictionary entry, the type constraints are used.

4 Learning from Projected Tags

Next we propose models to learn a morphological tagger from cross-lingually projected constraints.

4.1 Related work

HMMs have previously been used for weakly-supervised learning from token or type constraints (Das and Petrov, 2011; Li et al., 2012; Täckström et al., 2013). HMMs are generative models, and in this setting the words in the target sentence form the observed sequence and the morphological tags the hidden sequence. The projected constraints are used as partially observed training data for the hidden sequence.

Täckström et al. (2013) proposed a discriminative CRF model that relies on incorporating two sets of constraints, of which one is a subset of the other. Ganchev and Das (2013) used a similar CRF model, but instead of using the projected tags as hard constraints, they were employed as soft constraints with posterior regularization.

The model of Wisniewski et al. (2014) makes greedy predictions with a history-based model, that includes previously predicted tags in the sequence, during training and testing. The model is trained with a variant of the perceptron algorithm that allows a set of positive labels. When an incorrect prediction is made during training, the parameters are updated in the direction of all the positive labels.

4.2 HMM model

As a baseline model we use an HMM where the transition and emission distributions are parameterized by log-linear models (a feature-HMM). Training is performed with L-BFGS rather than with the EM algorithm. This parameterization was proposed by Berg-Kirkpatrick et al. (2010) and applied to cross-lingual POS induction by Das and

Petrov (2011) and Täckström et al. (2013).

Let $\mathbf{w}$ be the target sentence and $\mathbf{t}$ the sequence of tags for the sentence. The marginal probability of a sequence during training is

$$p(\mathbf{w}_{1:n}) = \sum_{\mathbf{t}_{1:n} \in \mathcal{T}} \prod_{i=1}^{n} p(t_i|t_{i-1})p(w_i|t_i),$$

where $\mathcal{T}$ is the set of tag sequences allowed by the type and token constraints. The probability of all other tag sequences are assumed to be 0.

The features in our model are similar to those used by Täckström et al. (2013), including features based on word and tag identity, suffixes up to length 3, punctuation and word clusters. Word clusters are obtained by clustering frequent words into 256 clusters with the Exchange algorithm (Uszkoreit and Brants, 2008), using the data and methodology detailed in Täckström et al. (2012).

4.3 Wsabie model

We propose a discriminative model based on Wsabie (Weston et al., 2011), a shallow neural network that learns to optimize precision at the top of a ranked list of labels. In our application, the goal is to learn to rank the set of tags allowed by the projected constraints in the training data above all other tags. In contrast to the HMM, which performs inference over the entire sequence, Wsabie makes the predictions at each token independently, based on a large context-size. Therefore, Wsabie inference is linear in the number of tags, while for an HMM it is quadratic, making the Wsabie model much faster during training and decoding.

Wsabie maps the input features and output labels into a low-dimensional joint space. The input vector x for a word w consists of the concatenation of word embeddings and sparse features extracted from w and the surrounding context. A mapping

$$\Theta_I(x) = Vx$$

maps $x \in \mathbb{R}^d$ into $\mathbb{R}^D$, with matrix $V \in \mathbb{R}^{D \times d}$ of parameters. The output tag t is mapped into the same space by

$$\Theta_O(t) = W_t,$$

where $W \in \mathbb{R}^{D \times L}$ is a matrix of output tag embeddings and W_t selects the column embedding of tag t. The model score for tag t given input token with feature vector x is the dot product

$$f_t(x) = \Theta_O(t)^T \Theta_I(x),$$

where the tags are ranked by the magnitude of $f_t(x)$. The norms of the columns of V and W are constrained, which acts as a regularizer.

The loss function is a margin-based hinge loss based on the rank of a tag given by $f_t(x)$. The rank is estimated by sampling an incorrect tag uniformly with replacement until the sampled tag violates the margin with a correct tag. Training is performed with stochastic gradient descent by performing a gradient step against the violating tag.

The word embedding features for the Wsabie models consist of 64-dimensional word vectors of the 5 words on either side of a token and of the token itself. The embeddings are trained with word2vec (Mikolov et al., 2013) on large corpora of newswire text.

Sparse features are based on prefixes and suffixes up to length 3 as well as word cluster features for a window size 3 around the token, using the clusters described in the previous section.

5 Experiments

We evaluate our model in two settings. The first evaluation measures the accuracy of the cross-lingual taggers on language pairs where annotated data is available for both languages. The annotated target language data is used only during evaluation and not for training. Second, we perform a downstream evaluation by including the morphological attributes predicted by the tagger as features in a dependency parser to guage the effectiveness of our approach in a setting where one does not have access to gold morphological annotations.

5.1 Experimental setup

As source of parallel training data we use Europarl[2] (Koehn, 2005) version 7. Sentences are tokenized but not lower-cased, and sentences longer than 80 words are excluded. In our experiments we learn taggers for a set of 11 European languages that have both UD training data with morphological features, and parallel data in Europarl: Bulgarian, Czech, Danish, Dutch, Finnish, Italian, Polish, Portuguese, Slovene, Spanish and Swedish. We train cross-lingual models in two setups: The first uses English as source language; in the second we train models with different source languages for each target language.

Word alignments over the parallel data are obtained using FastAlign (Dyer et al., 2013). High-

[2] http://www.statmt.org/europarl/

confidence bidirectional word alignments are constructed by intersecting the alignments in the two directions and including alignment points only if the posterior probabilities in both directions are above the alignment threshold α. For each language pair all the word-aligned parallel data available (between 10 and 50 million target-side tokens per language) are used to extract the type constraints, and the models are trained on a subset of 2 million target-side tokens (optionally with their token constraints).

The number of distinct attribute-value pairs appearing in the tagsets depends on the language pair and ranges between 35 and 79, with 54 on average (including POS tags). The number of distinct composite morphological tags is 423 on average, with a much larger range, between 81 and 1483. The English UD data has 116 tags composed out of 51 distinct attribute-value pairs. Therefore, we can project a reasonable number of morpho-syntactic attributes from English, although the number of attribute combinations that occur in the data is less than for morphologically richer languages.

The source text is tagged with supervised taggers, trained with Wsabie on the UD training data for each of the source languages used. For each language pair, we train a distinct source-side model covering only the attribute types appearing in both languages. This is meant to obtain a maximally accurate source-side tagger, while accepting that our approach cannot predict target-side attributes that are absent from the source language. The average accuracy of the English taggers on the UD test data is 94.96%. The source-side taggers over all the language pairs we experiment on have an average accuracy of 95.75%, with a minimum of 89.14% and a maximum of 98.59%.

5.2 Tuning

The hyperparameters of the Wsabie taggers are tuned on the English development set, and the same parameters are used for the Wsabie target-side models trained on the projected tags. The optimal setting is a learning rate of 0.01, embedding dimension size $D = 50$, margin 0.1, and 25 training iterations.

Hyperparameters for the projection models are set by tuning on the UD dev set accuracy for English to Danish. English was chosen as it is the language with the most available data and the most likely to be used when projecting to other languages; Danish simply because its corpus size is typical of the larger languages in Europarl. Using a small grid search, we choose the parameters that give the best average accuracy across all four projection model instances we consider. This allows using the same hyperparameters for all these models, an important factor in making them comparable in the evaluation, since the hyperparameters determine the effective training data. The parameters tuned in this manner are the alignment threshold α, which is set to 0.8, and the type distribution threshold β, set to 0.3.

5.3 Tagging evaluation setup

In order to evaluate the induced taggers on the annotated UD data for the target languages, we define two settings that circumvent mismatches between source and target language annotations to different degrees.

The STANDARD setting involves first making minor corrections to certain predicted POS values to account for inconsistencies in the original annotated data. When predicted by the model, the POS tag values absent from the target language training corpus are deterministically mapped to the most-related value present in the target language in the following way: `PROPN` to `NOUN`; `SYM` and `INTJ` to `X`; `SYM` and `X` to `PUNCT`. Besides POS, the evaluation considers only those attribute *types* that appear in both languages' training corpora, i.e., the set of attributes for which the model was trained. Note that this leaves cases intact where the model predicts certain attribute *values* that appear only in one of the two languages; it is thus penalised for making mistakes on values that it cannot learn under our projection approach.

The second evaluation setting, INTERSECTED, relaxes the latter aspect: it only considers attribute-value pairs appearing in the training corpora of both languages. The motivation for this is to get a better measurement of the accuracy of our method, assuming that the tagsets are consistent.

In both settings we report macro-averaged F1 scores over all the considered attribute types. Results for Wsabie are averaged over 3 random restarts because it uses stochastic optimization during training.

5.4 Tagging results projecting from English

Following previous work on projecting POS tags and the assumption that it is easier to obtain paral-

Model	STANDARD		INTERSECTED		POS	
HMM projected type	53.86	(-)	58.67	(-)	79.45	(-)
HMM projected type and token	48.49	(-)	52.40	(-)	73.61	(-)
unambiguous type	51.72	(0.33)	56.22	(0.36)	79.58	(0.22)
projected type	53.60	(0.16)	58.11	(0.18)	80.09	(0.12)
projected type and token	53.36	(0.19)	57.77	(0.21)	79.94	(0.11)
supervised 1K	62.44	(1.52)	61.74	(1.55)	72.51	(0.82)
supervised type	75.55	(1.88)	74.72	(1.95)	75.91	(1.16)

Table 1: Cross-lingual morphological tagging from English: Macro F1 scores averaged across 11 languages. All the results except for the first two rows are for Wsabie models. The standard deviation over 3 runs is given in brackets.

lel data between a low-resource language and English than with another language, we start by training cross-lingual taggers using English as source language.

The overall tagging results are given in Table 1. In addition to evaluating the morphological tags in the two settings described above, we also report accuracies for POS tags only, projected jointly with the morphological attributes.

We find that for both the HMM and Wsabie models the performance with type and token constraints is worse than when only using type constraints. Täckström et al. (2013) similarly found that for HMMs for POS projection, models with joint constraints do not perform better than those using only type constraints. They postulated that this is due to the type dictionaries having the same biases as token projections, and therefore the model with joint constraints not being able to filter out systematic errors in the projections.

For both sets of constraints the performance of the Wsabie model is close to that of the corresponding HMM, despite the Wsabie model having a linear runtime against the quadratic runtime of the HMM.

As another baseline we train a Wsabie model on unambiguous type constraints, i.e., we only extract training examples for words which only have a single tag in the tag dictionary. Including ambiguous type constraints gives an average improvement of 2.2%.

As a target ceiling on performance we train a Wsabie model with supervised type constraints. This model uses type constraints based on an oracle morphological tag dictionary extracted from the gold training data of the target language. It is trained on the same training data as the projected models (without token constraints). The model scores higher on STANDARD than on INTERSECTED, as it has access to annotations for the full set of tags used in the target language, not just the restricted set that can be projected. This oracle performs on average 17% better than the projected type constraints model on INTERSECTED. Therefore, despite the promising results of our approach, there is still a considerable amount of noise in the type constraints extracted from the aligned data.

We also compare the performance of the model to that of a supervised model trained on a small annotated corpus. Average performance when training on 1000 annotated tokens is only a few points higher than that of the best projected model for INTERSECTED. Given that is it expensive to let annotators learn to annotate a large set of attributes, even for a small corpus, it shows that our model can bring considerable benefits in practice to the development of NLP models for low-resource languages. It is possible to obtain further improvements in performance by learning jointly from a small annotated dataset and parallel data (Duong et al., 2014), but we leave that for future work.

The results when evaluating only the POS tags follow the same pattern, except that the overall level of accuracy is much higher than when considering all morphological attributes. For POS, the models with projected constraints actually perform better than those with supervised type constraints. In this case the benefits from learning constraints from a larger set of word types seem to outweigh the noise in the projections. The projected models are also more accurate than the supervised model trained on 1000 tokens.

5.5 Multilingual tagging results

Results for cross-lingual experiments on all pairs of the target languages under consideration are

	bg	cs	da	es	fi	it	nl	pl	pt	sl	sv	**Avg.**
en	46.7	49.7	58.0	55.7	54.0	59.6	**64.1**	45.0	57.8	51.0	47.9	53.6
bg	-	58.3	59.2	51.2	52.6	43.2	38.7	52.8	41.1	49.2	53.6	50.0
cs	55.2	-	54.5	42.3	48.4	51.3	45.0	**56.8**	33.6	**67.5**	53.2	50.8
da	**61.9**	**61.6**	-	41.8	49.1	45.5	49.6	53.7	44.0	49.3	**72.1**	52.9
es	54.3	58.8	41.3	-	53.0	**74.4**	52.1	52.2	**69.2**	53.8	46.9	55.6
fi	46.6	48.7	45.3	39.5	-	50.9	36.8	37.4	30.1	55.5	57.8	44.9
it	43.6	59.4	44.0	**74.0**	53.3	-	54.3	46.5	69.2	55.9	47.0	54.7
nl	44.7	59.5	56.2	54.8	54.0	60.3	-	55.9	58.6	48.6	51.6	54.4
pl	52.7	58.6	46.3	37.5	42.1	47.9	42.1	-	40.7	56.0	42.6	46.6
pt	45.4	45.0	49.6	66.2	42.6	69.5	50.1	43.5	-	47.8	43.9	50.3
sl	46.6	60.7	35.2	40.9	49.2	49.8	36.0	54.1	35.0	-	40.4	44.8
sv	50.1	54.6	**70.7**	47.7	**57.2**	49.7	46.9	41.6	46.3	43.5	-	50.8
Avg	49.8	55.9	50.9	50.1	50.5	54.7	46.9	49.0	47.8	52.6	50.6	

Table 2: Cross-lingual morphological tagging results (STANDARD F1 scores) per source and target language, Wsabie projected model with type constraints. Rows indicate source language and columns target language.

given in Table 2, using the STANDARD evaluation setup. We make use of Wsabie for these experiments, as it is a more efficient model, which is especially significant when training models with large tagsets.

We see that there is large variance in the morphological tagging accuracies across language pairs. In most cases the source language for which we learn the most accurate model for morphological tagging on the target language is a related language. The Romance languages we consider (Spanish, Italian and Portuguese) seem to transfer particularly well across each other. Swedish and Danish also transfer well to each other, while English transfers best to Dutch, which the former is most closely related to among the languages compared here. However, there are also some cases of unrelated source languages performing best: Using Danish as source language gives the highest performing models for both Bulgarian and Czech. When comparing these results, however, one should keep in mind that the attribute type sets used to train taggers from different source languages for the same target language is not always the same (due to our definition of the STANDARD evaluation), therefore these results should not be interpreted directly as indicating which source language gives the best target language performance on a particular tagset.

We compare the results of the STANDARD and INTERSECTED evaluations, both when using English as source language, and when using the source language which gives the highest accuracy on STANDARD for each target language (Table 3). We see that the gap in performance between the two evaluations tends to be larger when projecting from English than when projecting from the source language which performs best for each target language.

	STANDARD		INTERSECTED	
	en-	best-	en-	best-
bg	46.7	61.88	51.6	64.97
cs	49.7	61.57	55.7	63.97
da	58.0	70.74	65.4	73.14
es	55.7	74.01	60.7	74.62
fi	54.0	57.23	59.1	59.11
it	59.6	74.42	66.1	75.32
nl	64.1	64.12	64.7	64.66
pl	45.0	56.83	47.3	60.39
pt	57.8	69.22	60.2	73.10
sl	51.0	67.48	53.4	69.86
sv	47.9	72.07	55.1	74.60

Table 3: Comparison of the performance of the most accurate cross-lingual taggers for each target language, compared to having English as source language.

One of the main causes of variation in performance is annotation differences. Languages that are morphologically rich tend to have lower performance, but we also see variation between similar languages: There is a 10% performance gap between Danish and Swedish when projecting from English, even though they are closely related.

We also investigate the effect of the choice of source language on the accuracy of the projected POS tags (Table 4). Again, we compare the performance with English as source (which is standard for previous work on POS projection) to that of the best source language for each target. Although the gap in performance is smaller than for

Target	en-	best-
bg	81.84	81.84 (en)
cs	80.41	86.29 (sl)
da	80.69	84.85 (sv)
es	86.02	89.04 (it)
fi	77.07	77.48 (cs)
it	83.46	86.91 (es)
nl	73.05	76.02 (da)
pl	79.38	82.66 (cs)
pt	84.30	87.98 (es)
sl	74.71	83.21 (cs)
sv	80.37	86.47 (da)

Table 4: Wsabie projected model with type constraints, POS accuracy with English and the best language for each target as source.

	no morph	projected type	supervised
bg	79.14	78.99	79.62
cs	76.88	77.25	79.03
da	69.73	70.04	71.51
es	77.66	78.08	78.64
fi	61.78	62.68	70.42
it	81.51	81.49	82.24
nl	64.76	65.80	65.92
pl	70.83	71.89	74.03
pt	75.92	76.71	77.98
sl	77.17	77.46	79.25
sv	72.92	74.09	74.58
Avg.	73.48	74.04	75.75

Table 6: Dependency parsing results (LAS) with no, projected and supervised morphological tags.

the full evaluation, we see that for most target languages we can still do better by projecting from a language other than English.

Detailed per attribute results for the STANDARD evaluation are given in Table 5, again comparing the results of projecting from English to that of the most accurate model for each target language. We see that there are large differences in accuracy across attributes and across languages. In some cases, the transfer is unsuccessful. For example, degree accuracy in Italian is 2% F1 when projecting from English and 14% F1 projecting from Portuguese. Some of the cases can be explained by differences in where an attribute is marked: For example, for definiteness the performance is 1% from English to Bulgarian, as Bulgarian marks definiteness on nouns and adjectives rather than on determiners. Other attributes are very language-dependent. Gender transfers well between Romance languages, but poorly when transferring from English.

5.6 Parsing evaluation

To evaluate the effect of our models on a downstream task, we apply the cross-lingual taggers induced using English as source language to dependency parsing. This is applicable to a scenario where a language might have a corpus annotated with dependency trees and universal POS, but not morphological attributes. We want to determine how much of the performance gain from features based on supervised morphological tags we can recover with the tags predicted by our model.

As baseline we use a reimplementation of Zhang and Nivre (2011), an arc-eager transition-based dependency parser with a rich feature-set, with beam-size 8, trained for 10 epochs with a structured perceptron. We assume that universal POS tags are available, using a supervised SVM POS tagger for training and evaluation.

To include the morphology, we add features based on the predicted tags of the word on top of the stack and the first two words on the buffer.

Parsing results are given in Table 6. We report labelled attachment scores (LAS) for the baseline with no morphological tags, the model with features predicted by Wsabie with projected type constraints, and the model with features predicted by the supervised morphological tagger.

We obtain improvements in parsing accuracies for all languages except Bulgarian when adding the induced morphological tags. Using the projected tags as features recovers 24.67% (0.6 LAS absolute) of the average gain that supervised morphology features delivers over the baseline parser. The parser with features from the supervised tagger trained on 1000 tokens obtains 73.63 LAS on average. This improvement of $+0.15$ LAS over the baseline versus the $+0.6$ of our method shows that the tags predicted by our projected models are more useful as features than those predicted by a small supervised model.

To investigate the effect of source language choice for the projected models in this evaluation, we trained a model for Swedish using Danish as source language. The parsing performance is insignificantly different from using English as source, despite the accuracy of the tags projected

Target	bg		cs		da		es		fi		it		nl		pl		pt		sl		sv	
Source	en	da	en	it	en	sv	en	it	en	sv	en	pt	en	en	en	nl	en	it	en	cs	en	da
Case	40	62	2	-	62	18	4	-	5	26	-	-	16	16	4	4	50	-	2	68	10	14
Definite	1	68	-	-	0	64	97	97	-	-	89	91	89	89	-	-	93	93	2	-	19	66
Degree	67	63	69	2	72	77	5	26	50	47	2	14	56	56	57	47	2	18	63	74	70	81
Gender	1	6	2	46	7	78	0	85	-	-	2	80	0	0	3	0	1	77	2	61	7	81
Mood	61	66	55	83	81	94	72	80	69	79	76	83	69	69	58	63	74	75	68	91	73	94
Number	69	71	67	75	60	82	54	92	67	68	57	90	78	78	69	63	62	75	68	91	64	94
NumType	64	62	91	86	84	-	82	85	86	-	89	66	78	78	-	-	63	65	86	73	-	-
Person	54	28	63	68	56	-	58	79	51	-	57	80	82	82	55	59	61	74	67	91	-	-
Poss	76	77	90	84	97	98	94	93	-	-	87	88	64	64	-	-	96	98	67	62	99	97
PronType	72	71	41	38	46	2	82	74	42	0	76	71	81	81	38	50	81	79	43	73	0	3
Reflex	0	85	0	0	62	-	0	0	61	-	0	0	60	60	0	97	0	0	-	-	-	-
Tense	60	63	68	70	81	85	69	81	67	77	75	75	74	74	66	66	64	72	62	74	65	86
VerbForm	43	49	59	78	75	79	79	81	64	65	82	81	78	78	56	66	79	72	59	74	73	86
Voice	9	75	9	-	6	89	-	-	10	76	-	-	-	-	55	-	-	-	-	-	15	90

Table 5: Cross-lingual tagging results (F1 scores) per language and per attribute (not showing POS and a small number of attribute types that only appear with 1 or 2 language pairs), for Wsabie projected with type constraints. English and best source language.

from Danish being higher.

Faruqui et al. (2016) show that features from induced morpho-syntactic lexicons can also improve dependency parsing accuracy. However, their method relies on having a seed lexicon of 1000 annotated word types, while our method does not require any morphological annotations in the target language.

6 Future Work

A big challenge in cross-lingual morphology is that of relatedness between source and target languages. Although we evaluate our models on multiple source-target language pairs, more work is required to investigate strategies for choosing which source language to use for a low-resource target language. A related direction is to constructing models from multiple source languages, as our results show that the overall best-performing source language for a given target language may not always have the best performance on all attributes.

Another direction is to make use of dictionaries such as Wiktionary to obtain type constraints, similar to previous work on weakly-supervised POS tagging (Li et al., 2012; Täckström et al., 2013). Sylak-Glassman et al. (2015b) and Sylak-Glassman et al. (2015a) proposed a morphological schema and method to extract annotations in that schema from Wiktionary. Although different from the schema used in this paper, their method can be used to extract type dictionaries for morphological tags that can be used to complement constraints extracted from parallel data.

Finally, greater use can be made of syntactic information: There is a close relation between the syntactic structure expressed in dependency parses and inflections in morphologically rich languages; by including this syntactic structure in our models we can induce morphological tags, e.g. related to case, that is also expressed in dependency parses.

7 Conclusion

In this paper we proposed a method that can successfully induce morphological taggers for resource-scarce languages using tags projected across bitext. It relies on access to a morphological tagger for a source-language and a moderate amount of bitext. The method obtains strong performance on a range of language pairs. We showed that downstream tasks such as dependency parsing can be improved by using the predictions from the tagger as features. Our results provide a strong baseline for future work in weakly-supervised morphological tagging.

Acknowledgments

This research was primarily performed while the first author was an intern at Google Inc. We thank Oscar Täckström, Kuzman Ganchev, Bernd Bohnet and Ryan McDonald for valuable assistance and discussions about this work.

References

Malin Ahlberg, Markus Forsberg, and Mans Hulden. 2015. Paradigm classification in supervised learning of morphology. In *Proceedings of NAACL*, pages 1024–1029.

Taylor Berg-Kirkpatrick, Alexandre Bouchard-Côté, John DeNero, and Dan Klein. 2010. Painless unsupervised learning with features. In *Proceedings of NAACL*, pages 582–590.

Lauent Besacier, Ettiene Barnard, Alexey Karpov, and Tanja Schultz. 2014. Automatic speech recognition for under-resourced languages: A survey. *Speech Communication*, 56:85–100.

Mathias Creutz and Krista Lagus. 2007. Unsupervised models for morpheme segmentation and morphology learning. *ACM Transactions on Speech and Language Processing*, 4(1).

Dipanjan Das and Slav Petrov. 2011. Unsupervised part-of-speech tagging with bilingual graph-based projections. In *Proceedings of ACL*, pages 600–609.

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal dependencies: A cross-linguistic typology. In *Proceedings of LREC*.

Long Duong, Trevor Cohn, Karin Verspoor, Steven Bird, and Paul Cook. 2014. What can we get from 1000 tokens? A case study of multilingual pos tagging for resource-poor languages. In *Proceedings of EMNLP*, pages 886–897.

Greg Durrett and John DeNero. 2013. Supervised learning of complete morphological paradigms. In *Proceedings of NAACL*, pages 1185–1195.

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of IBM Model 2. In *Proceeding of NAACL*, pages 682–686.

Manaal Faruqui, Ryan McDonald, and Radu Soricut. 2016. Morpho-syntactic lexicon generation using graph-based semi-supervised learning. *Transactions of the Association for Computational Linguistics*, 4:1–16.

Victoria Fossum and Steven Abney. 2005. Automatically inducing a part-of-speech tagger by projecting from multiple source languages across aligned corpora. In *Proceedings of IJCNLP*, pages 862–873.

Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proceedings of EMNLP*, pages 1996–2006.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, et al. 2009. The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL: Shared Task*, pages 1–18.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Shen Li, João Graça, and Ben Taskar. 2012. Wiki-ly supervised part-of-speech tagging. In *Proceedings of EMNLP-CoNLL*, pages 1389–1398, July.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Thomas Müller and Hinrich Schuetze. 2015. Robust morphological tagging with word representations. In *Proceedings of NAACL*, pages 526–536, Denver, Colorado, May–June.

Joakim Nivre, Željko Agić, Maria Jesus Aranzabe, Masayuki Asahara, Aitziber Atutxa, Miguel Ballesteros, John Bauer, Kepa Bengoetxea, Riyaz Ahmad Bhat, Cristina Bosco, Sam Bowman, Giuseppe G. A. Celano, Miriam Connor, Marie-Catherine de Marneffe, Arantza Diaz de Ilarraza, Kaja Dobrovoljc, Timothy Dozat, Tomaž Erjavec, Richárd Farkas, Jennifer Foster, Daniel Galbraith, Filip Ginter, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Berta Gonzales, Bruno Guillaume, Jan Hajič, Dag Haug, Radu Ion, Elena Irimia, Anders Johannsen, Hiroshi Kanayama, Jenna Kanerva, Simon Krek, Veronika Laippala, Alessandro Lenci, Nikola Ljubešić, Teresa Lynn, Christopher Manning, Cătălina Mărănduc, David Mareček, Héctor Martínez Alonso, Jan Mašek, Yuji Matsumoto, Ryan McDonald, Anna Missilä, Verginica Mititelu, Yusuke Miyao, Simonetta Montemagni, Shunsuke Mori, Hanna Nurmi, Petya Osenova, Lilja Øvrelid, Elena Pascual, Marco Passarotti, Cenel-Augusto Perez, Slav Petrov, Jussi Piitulainen, Barbara Plank, Martin Popel, Prokopis Prokopidis, Sampo Pyysalo, Loganathan Ramasamy, Rudolf Rosa, Shadi Saleh, Sebastian Schuster, Wolfgang Seeker, Mojgan Seraji, Natalia Silveira, Maria Simi, Radu Simionescu, Katalin Simkó, Kiril Simov, Aaron Smith, Jan Štěpánek, Alane Suhr, Zsolt Szántó, Takaaki Tanaka, Reut Tsarfaty, Sumire Uematsu, Larraitz Uria, Viktor Varga, Veronika Vincze, Zdeněk Žabokrtský, Daniel Zeman, and Hanzhi Zhu. 2015. Universal dependencies 1.2. LINDAT/CLARIN digital library at Institute of Formal and Applied Linguistics, Charles University in Prague.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proceedings of LREC*.

Benjamin Snyder and Regina Barzilay. 2008. Unsupervised multilingual learning for morphological segmentation. In *Proceedings of ACL*, pages 737–745.

John Sylak-Glassman, Christo Kirov, Matt Post, Roger Que, and David Yarowsky. 2015a. A universal feature schema for rich morphological annotation and fine-grained cross-lingual part-of-speech tagging. In

Proceedings of Systems and Frameworks for Computational Morphology: Fourth International Workshop, pages 72–93. Springer International Publishing, Cham.

John Sylak-Glassman, Christo Kirov, David Yarowsky, and Roger Que. 2015b. A language-independent feature schema for inflectional morphology. In *Proceedings of ACL-IJCNLP (short papers)*, pages 674–680.

Oscar Täckström, Ryan McDonald, and Jakob Uszkoreit. 2012. Cross-lingual word clusters for direct transfer of linguistic structure. In *Proceedings of NAACL*, pages 477–487.

Oscar Täckström, Dipanjan Das, Slav Petrov, Ryan McDonald, and Joakim Nivre. 2013. Token and type constraints for cross-lingual part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 1:1–12.

Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. 2008. Applying morphology generation models to machine translation. In *Proceedings of ACL-HLT*, pages 558–566.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Marie Candito, Jennifer Foster, Yannick Versley, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (SPMRL): what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12.

Jakob Uszkoreit and Thorsten Brants. 2008. Distributed word clustering for large scale class-based language modeling in machine translation. In *Proceedings of ACL-HLT*, pages 755–762.

Jason Weston, Samy Bengio, and Nicolas Usunier. 2011. Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Guillaume Wisniewski, Nicolas Pcheux, Souhir Gahbiche-Braham, and Franois Yvon. 2014. Cross-lingual part-of-speech tagging through ambiguous learning. In *Proceedings of EMNLP*, pages 1779–1785.

David Yarowsky, Grace Ngai, and Richard Wicentowski. 2001. Incuding multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of HLT*.

Daniel Zeman. 2008. Reusable tagset conversion using tagset drivers. In *Proceedings of LREC*.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *Proceedings of ACL-HLT*, pages 188–193.

Semi-Supervised Learning for Neural Machine Translation

Yong Cheng[#], Wei Xu[#], Zhongjun He[+], Wei He[+], Hua Wu[+], Maosong Sun[†] and Yang Liu[†] *
[#]Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
[†]State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing, China
[+]Baidu Inc., Beijing, China
chengyong3001@gmail.com weixu@tsinghua.edu.cn
{hezhongjun,hewei06,wu_hua}@baidu.com
{sms,liuyang2011}@tsinghua.edu.cn

Abstract

While end-to-end neural machine translation (NMT) has made remarkable progress recently, NMT systems only rely on parallel corpora for parameter estimation. Since parallel corpora are usually limited in quantity, quality, and coverage, especially for low-resource languages, it is appealing to exploit monolingual corpora to improve NMT. We propose a semi-supervised approach for training NMT models on the concatenation of labeled (parallel corpora) and unlabeled (monolingual corpora) data. The central idea is to reconstruct the monolingual corpora using an autoencoder, in which the source-to-target and target-to-source translation models serve as the encoder and decoder, respectively. Our approach can not only exploit the monolingual corpora of the target language, but also of the source language. Experiments on the Chinese-English dataset show that our approach achieves significant improvements over state-of-the-art SMT and NMT systems.

1 Introduction

End-to-end neural machine translation (NMT), which leverages a single, large neural network to directly transform a source-language sentence into a target-language sentence, has attracted increasing attention in recent several years (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). Free of latent structure design and feature engineering that are critical in conventional statistical machine translation (SMT) (Brown et al., 1993; Koehn et al., 2003; Chiang, 2005), NMT has proven to excel in modeling long-distance dependencies by enhancing recurrent neural networks (RNNs) with the gating (Hochreiter and Schmidhuber, 1993; Cho et al., 2014; Sutskever et al., 2014) and attention mechanisms (Bahdanau et al., 2015).

However, most existing NMT approaches suffer from a major drawback: they heavily rely on parallel corpora for training translation models. This is because NMT directly models the probability of a target-language sentence given a source-language sentence and does not have a separate language model like SMT (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015). Unfortunately, parallel corpora are usually only available for a handful of research-rich languages and restricted to limited domains such as government documents and news reports. In contrast, SMT is capable of exploiting abundant target-side monolingual corpora to boost fluency of translations. Therefore, the unavailability of large-scale, high-quality, and wide-coverage parallel corpora hinders the applicability of NMT.

As a result, several authors have tried to use abundant monolingual corpora to improve NMT. Gulccehre et al. (2015) propose two methods, which are referred to as shallow fusion and deep fusion, to integrate a language model into NMT. The basic idea is to use the language model to score the candidate words proposed by the translation model at each time step or concatenating the hidden states of the language model and the decoder. Although their approach leads to significant improvements, one possible downside is that the network architecture has to be modified to integrate the language model.

Alternatively, Sennrich et al. (2015) propose two approaches to exploiting monolingual corpora that is transparent to network architectures. The first approach pairs monolingual sentences with dummy input. Then, the parameters of encoder

* Yang Liu is the corresponding author.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1965–1974,
Berlin, Germany, August 7-12, 2016.

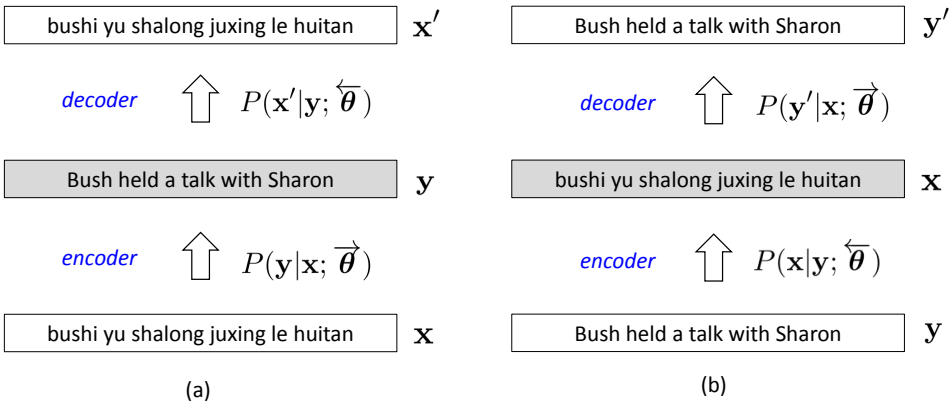


Figure 1: Examples of (a) source autoencoder and (b) target autoencoder on monolingual corpora. Our idea is to leverage autoencoders to exploit monolingual corpora for NMT. In a source autoencoder, the source-to-target model $P(\mathbf{y}|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})$ serves as an encoder to transform the observed source sentence $\mathbf{x}$ into a latent target sentence $\mathbf{y}$ (highlighted in grey), from which the target-to-source model $P(\mathbf{x}'|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})$ reconstructs a copy of the observed source sentence $\mathbf{x}'$ from the latent target sentence. As a result, monolingual corpora can be combined with parallel corpora to train bidirectional NMT models in a semi-supervised setting.

and attention model are fixed when training on these pseudo parallel sentence pairs. In the second approach, they first train a nerual translation model on the parallel corpus and then use the learned model to translate a monolingual corpus. The monolingual corpus and its translations constitute an additional pseudo parallel corpus. Similar ideas have also been suggested in conventional SMT (Ueffing et al., 2007; Bertoldi and Federico, 2009). Sennrich et al. (2015) report that their approach significantly improves translation quality across a variety of language pairs.

In this paper, we propose semi-supervised learning for neural machine translation. Given labeled (i.e., parallel corpora) and unlabeled (i.e., monolingual corpora) data, our approach jointly trains source-to-target and target-to-source translation models. The key idea is to append a reconstruction term to the training objective, which aims to reconstruct the observed monolingual corpora using an autoencoder. In the autoencoder, the source-to-target and target-to-source models serve as the encoder and decoder, respectively. As the inference is intractable, we propose to sample the full search space to improve the efficiency. Specifically, our approach has the following advantages:

1. *Transparent to network architectures*: our approach does not depend on specific architectures and can be easily applied to arbitrary end-to-end NMT systems.
2. *Both the source and target monolingual corpora can be used*: our approach can benefit NMT not only using target monolingual corpora in a conventional way, but also the monolingual corpora of the source language.

Experiments on Chinese-English NIST datasets show that our approach results in significant improvements in both directions over state-of-the-art SMT and NMT systems.

2 Semi-Supervised Learning for Neural Machine Translation

2.1 Supervised Learning

Given a parallel corpus $\mathcal{D} = \{\langle \mathbf{x}^{(n)}, \mathbf{y}^{(n)} \rangle\}_{n=1}^{N}$, the standard training objective in NMT is to maximize the likelihood of the training data:

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)};\boldsymbol{\theta}), \quad (1)$$

where $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ is a neural translation model and $\boldsymbol{\theta}$ is a set of model parameters. $\mathcal{D}$ can be seen as *labeled* data for the task of predicting a target sentence $\mathbf{y}$ given a source sentence $\mathbf{x}$.

As $P(\mathbf{y}|\mathbf{x};\boldsymbol{\theta})$ is modeled by a single, large neural network, there does not exist a separate target language model $P(\mathbf{y};\boldsymbol{\theta})$ in NMT. Therefore, parallel corpora have been the only resource for parameter estimation in most existing NMT systems. Unfortunately, even for a handful of resource-rich

languages, the available domains are unbalanced and restricted to government documents and news reports. Therefore, the availability of large-scale, high-quality, and wide-coverage parallel corpora becomes a major obstacle for NMT.

2.2 Autoencoders on Monolingual Corpora

It is appealing to explore the more readily available, abundant monolingual corpora to improve NMT. Let us first consider an *unsupervised* setting: how to train NMT models on a monolingual corpus $\mathcal{T} = \{\mathbf{y}^{(t)}\}_{t=1}^{T}$?

Our idea is to leverage *autoencoders* (Vincent et al., 2010; Socher et al., 2011): (1) *encoding* an observed target sentence into a latent source sentence using a target-to-source translation model and (2) *decoding* the source sentence to reconstruct the observed target sentence using a source-to-target model. For example, as shown in Figure 1(b), given an observed English sentence "Bush held a talk with Sharon", a target-to-source translation model (i.e., encoder) transforms it into a Chinese translation "bushi yu shalong juxing le huitan" that is unobserved on the training data (highlighted in grey). Then, a source-to-target translation model (i.e., decoder) reconstructs the observed English sentence from the Chinese translation.

More formally, let $P(\mathbf{y}|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})$ and $P(\mathbf{x}|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})$ be *source-to-target* and *target-to-source* translation models respectively, where $\overrightarrow{\boldsymbol{\theta}}$ and $\overleftarrow{\boldsymbol{\theta}}$ are corresponding model parameters. An autoencoder aims to reconstruct the observed target sentence via a latent source sentence:

$$\begin{aligned} & P(\mathbf{y}'|\mathbf{y};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}}) \\ = & \sum_{\mathbf{x}} P(\mathbf{y}',\mathbf{x}|\mathbf{y};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}}) \\ = & \sum_{\mathbf{x}} \underbrace{P(\mathbf{x}|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})}_{encoder} \underbrace{P(\mathbf{y}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})}_{decoder}, \end{aligned} \quad (2)$$

where $\mathbf{y}$ is an observed target sentence, $\mathbf{y}'$ is a copy of $\mathbf{y}$ to be reconstructed, and $\mathbf{x}$ is a latent source sentence.

We refer to Eq. (2) as a *target autoencoder*. [1] Likewise, given a monolingual corpus of source language $\mathcal{S} = \{\mathbf{x}^{(s)}\}_{s=1}^{S}$, it is natural to introduce a *source autoencoder* that aims at reconstructing the observed source sentence via a latent target sentence:

$$\begin{aligned} & P(\mathbf{x}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}}) \\ = & \sum_{\mathbf{y}} P(\mathbf{x}',\mathbf{y}|\mathbf{x};\overleftarrow{\boldsymbol{\theta}}) \\ = & \sum_{\mathbf{y}} \underbrace{P(\mathbf{y}|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})}_{encoder} \underbrace{P(\mathbf{x}'|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})}_{decoder}. \end{aligned} \quad (3)$$

Please see Figure 1(a) for illustration.

2.3 Semi-Supervised Learning

As the autoencoders involve both source-to-target and target-to-source models, it is natural to combine parallel corpora and monolingual corpora to learn birectional NMT translation models in a semi-supervised setting.

Formally, given a parallel corpus $\mathcal{D} = \{\langle \mathbf{x}^{(n)}, \mathbf{y}^{(n)} \rangle\}_{n=1}^{N}$, a monolingual corpus of target language $\mathcal{T} = \{\mathbf{y}^{(t)}\}_{t=1}^{T}$, and a monolingual corpus of source language $\mathcal{S} = \{\mathbf{x}^{(s)}\}_{s=1}^{S}$, we introduce our new semi-supervised training objective as follows:

$$\begin{aligned} & J(\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}}) \\ = & \underbrace{\sum_{n=1}^{N} \log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)};\overrightarrow{\boldsymbol{\theta}})}_{source\text{-}to\text{-}target\ likelihood} \\ & + \underbrace{\sum_{n=1}^{N} \log P(\mathbf{x}^{(n)}|\mathbf{y}^{(n)};\overleftarrow{\boldsymbol{\theta}})}_{target\text{-}to\text{-}source\ likelihood} \\ & + \lambda_1 \underbrace{\sum_{t=1}^{T} \log P(\mathbf{y}'|\mathbf{y}^{(t)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})}_{target\ autoencoder} \\ & + \lambda_2 \underbrace{\sum_{s=1}^{S} \log P(\mathbf{x}'|\mathbf{x}^{(s)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})}_{source\ autoencoder}, \end{aligned} \quad (4)$$

where λ_1 and λ_2 are hyper-parameters for balancing the preference between likelihood and autoencoders.

Note that the objective consists of four parts: source-to-target likelihood, target-to-source likelihood, target autoencoder, and source autoencoder. In this way, our approach is capable of exploiting abundant monolingual corpora of both source and target languages.

[1] Our definition of auotoencoders is inspired by Ammar et al. (2014). Note that our autoencoders inherit the same spirit from conventional autoencoders (Vincent et al., 2010; Socher et al., 2011) except that the hidden layer is denoted by a latent sentence instead of real-valued vectors.

The optimal model parameters are given by

$$\overrightarrow{\boldsymbol{\theta}}^* = \text{argmax}\left\{\sum_{n=1}^{N}\log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)};\overrightarrow{\boldsymbol{\theta}}) + \lambda_1\sum_{t=1}^{T}\log P(\mathbf{y}'|\mathbf{y}^{(t)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}}) + \lambda_2\sum_{s=1}^{S}\log P(\mathbf{x}'|\mathbf{x}^{(s)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})\right\} \quad (5)$$

$$\overleftarrow{\boldsymbol{\theta}}^* = \text{argmax}\left\{\sum_{n=1}^{N}\log P(\mathbf{x}^{(n)}|\mathbf{y}^{(n)};\overleftarrow{\boldsymbol{\theta}}) + \lambda_1\sum_{t=1}^{T}\log P(\mathbf{y}'|\mathbf{y}^{(t)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}}) + \lambda_2\sum_{s=1}^{S}\log P(\mathbf{x}'|\mathbf{x}^{(s)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})\right\} \quad (6)$$

It is clear that the source-to-target and target-to-source models are connected via the autoencoder and can hopefully benefit each other in joint training.

2.4 Training

We use mini-batch stochastic gradient descent to train our joint model. For each iteration, besides the mini-batch from the parallel corpus, we also construct two additional mini-batches by randomly selecting sentences from the source and target monolingual corpora. Then, gradients are collected from these mini-batches to update model parameters.

The partial derivative of $J(\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})$ with respect to the source-to-target model $\overrightarrow{\boldsymbol{\theta}}$ is given by

$$\begin{aligned}&\frac{\partial J(\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})}{\partial\overrightarrow{\boldsymbol{\theta}}}\\ =&\sum_{n=1}^{N}\frac{\partial\log P(\mathbf{y}^{(n)}|\mathbf{x}^{(n)};\overrightarrow{\boldsymbol{\theta}})}{\partial\overrightarrow{\boldsymbol{\theta}}}\\ &+\lambda_1\sum_{t=1}^{T}\frac{\partial\log P(\mathbf{y}'|\mathbf{y}^{(t)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})}{\partial\overrightarrow{\boldsymbol{\theta}}}\\ &+\lambda_2\sum_{s=1}^{S}\frac{\partial\log P(\mathbf{x}'|\mathbf{x}^{(s)};\overrightarrow{\boldsymbol{\theta}},\overleftarrow{\boldsymbol{\theta}})}{\partial\overrightarrow{\boldsymbol{\theta}}}. \quad (7)\end{aligned}$$

The partial derivative with respect to $\overleftarrow{\boldsymbol{\theta}}$ can be calculated similarly.

Unfortunately, the second and third terms in Eq. (7) are intractable to calculate due to the exponential search space. For example, the derivative in the third term in Eq. (7) is given by

$$\frac{\sum_{\mathbf{x}\in\mathcal{X}(\mathbf{y})}P(\mathbf{x}|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})P(\mathbf{y}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})\frac{\partial\log P(\mathbf{y}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})}{\partial\overrightarrow{\boldsymbol{\theta}}}}{\sum_{\mathbf{x}\in\mathcal{X}(\mathbf{y})}P(\mathbf{x}|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})P(\mathbf{y}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})}. \quad (8)$$

It is prohibitively expensive to compute the sums due to the exponential search space of $\mathcal{X}(\mathbf{y})$.

Alternatively, we propose to use a subset of the full space $\tilde{\mathcal{X}}(\mathbf{y}) \subset \mathcal{X}(\mathbf{y})$ to approximate Eq. (8):

$$\frac{\sum_{\mathbf{x}\in\tilde{\mathcal{X}}(\mathbf{y})}P(\mathbf{x}|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})P(\mathbf{y}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})\frac{\partial\log P(\mathbf{y}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})}{\partial\overrightarrow{\boldsymbol{\theta}}}}{\sum_{\mathbf{x}\in\tilde{\mathcal{X}}(\mathbf{y})}P(\mathbf{x}|\mathbf{y};\overleftarrow{\boldsymbol{\theta}})P(\mathbf{y}'|\mathbf{x};\overrightarrow{\boldsymbol{\theta}})}. \quad (9)$$

In practice, we use the top-k list of candidate translations of $\mathbf{y}$ as $\tilde{\mathcal{X}}(\mathbf{y})$. As $|\tilde{\mathcal{X}}(\mathbf{y})| \ll \mathcal{X}|(\mathbf{y})|$, it is possible to calculate Eq. (9) efficiently by enumerating all candidates in $\tilde{\mathcal{X}}(\mathbf{y})$. In practice, we find this approximation results in significant improvements and $k = 10$ seems to suffice to keep the balance between efficiency and translation quality.

		Chinese	English
Parallel	# Sent.	2.56M	
	# Word	67.54M	74.82M
	Vocab.	0.21M	0.16M
Monolingual	# Sent.	18.75M	22.32M
	# Word	451.94M	399.83M
	Vocab.	0.97M	1.34M

Table 1: Characteristics of parallel and monolingual corpora.

3 Experiments

3.1 Setup

We evaluated our approach on the Chinese-English dataset.

As shown in Table 1, we use both a parallel corpus and two monolingual corpora as the training set. The parallel corpus from LDC consists of 2.56M sentence pairs with 67.53M Chinese words and 74.81M English words. The vocabulary sizes of Chinese and English are 0.21M and 0.16M, respectively. We use the Chinese and English parts of the Xinhua portion of the GIGAWORD corpus as the monolingual corpora. The Chinese monolingual corpus contains 18.75M sentences with 451.94M words. The English corpus contains 22.32M sentences with 399.83M words. The vocabulary sizes of Chinese and English are 0.97M and 1.34M, respectively.

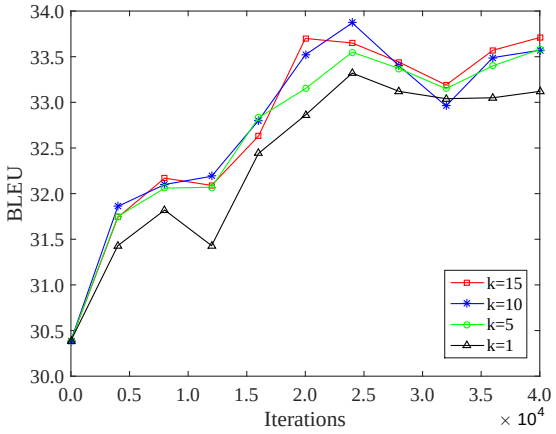


Figure 2: Effect of sample size k on the Chinese-to-English validation set.

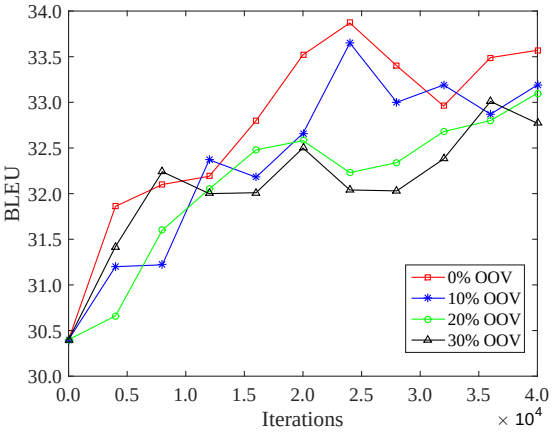


Figure 4: Effect of OOV ratio on the Chinese-to-English validation set.

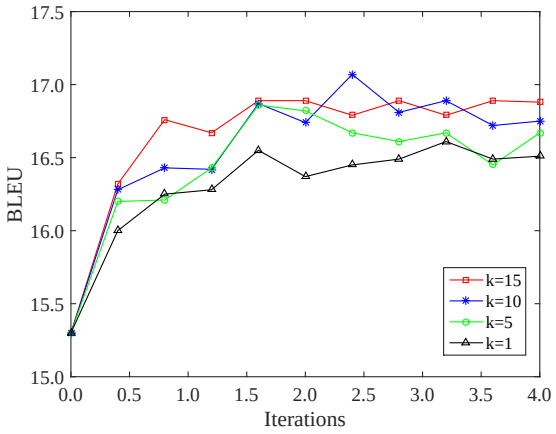


Figure 3: Effect of sample size k on the English-to-Chinese validation set.

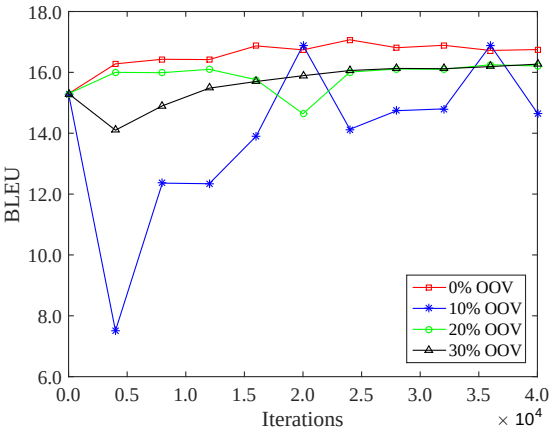


Figure 5: Effect of OOV ratio on the English-to-Chinese validation set.

For Chinese-to-English translation, we use the NIST 2006 Chinese-English dataset as the validation set for hyper-parameter optimization and model selection. The NIST 2002, 2003, 2004, and 2005 datasets serve as test sets. Each Chinese sentence has four reference translations. For English-to-Chinese translation, we use the NIST datasets in a reverse direction: treating the first English sentence in the four reference translations as a source sentence and the original input Chinese sentence as the single reference translation. The evaluation metric is case-insensitive BLEU (Papineni et al., 2002) as calculated by the `multi-bleu.perl` script.

We compared our approach with two state-of-the-art SMT and NMT systems:

1. MOSES (Koehn et al., 2007): a phrase-based SMT system;
2. RNNSEARCH (Bahdanau et al., 2015): an attention-based NMT system.

For MOSES, we use the default setting to train the phrase-based translation on the parallel corpus and optimize the parameters of log-linear models using the minimum error rate training algorithm (Och, 2003). We use the SRILM toolkit (Stolcke, 2002) to train 4-gram language models.

For RNNSEARCH, we use the parallel corpus to train the attention-based neural translation models. We set the vocabulary size of word embeddings to 30K for both Chinese and English. We follow Luong et al. (2015) to address rare words.

On top of RNNSEARCH, our approach is capable of training bidirectional attention-based neural translation models on the concatenation of parallel and monolingual corpora. The sample size k is set to 10. We set the hyper-parameter $\lambda_1 = 0.1$ and

$\lambda_2 = 0$ when we add the target monolingual corpus, and $\lambda_1 = 0$ and $\lambda_2 = 0.1$ for source monolingual corpus incorporation. The threshold of gradient clipping is set to 0.05. The parameters of our model are initialized by the model trained on parallel corpus.

3.2 Effect of Sample Size k

As the inference of our approach is intractable, we propose to approximate the full search space with the top-k list of candidate translations to improve efficiency (see Eq. (9)).

Figure 2 shows the BLEU scores of various settings of k over time. Only the English monolingual corpus is appended to the training data. We observe that increasing the size of the approximate search space generally leads to improved BLEU scores. There are significant gaps between $k = 1$ and $k = 5$. However, keeping increasing k does not result in significant improvements and decreases the training efficiency. We find that $k = 10$ achieves a balance between training efficiency and translation quality. As shown in Figure 3, similar findings are also observed on the English-to-Chinese validation set. Therefore, we set $k = 10$ in the following experiments.

3.3 Effect of OOV Ratio

Given a parallel corpus, what kind of monolingual corpus is most beneficial for improving translation quality? To answer this question, we investigate the effect of *OOV ratio* on translation quality, which is defined as

$$\text{ratio} = \frac{\sum_{y \in \mathbf{y}} [\![y \notin \mathcal{V}_{D_t}]\!]}{|\mathbf{y}|}, \quad (10)$$

where $\mathbf{y}$ is a target-language sentence in the monolingual corpus $\mathcal{T}$, y is a target-language word in $\mathbf{y}$, $\mathcal{V}_{D_t}$ is the vocabulary of the target side of the parallel corpus D.

Intuitively, the OOV ratio indicates how a sentence in the monolingual resembles the parallel corpus. If the ratio is 0, all words in the monolingual sentence also occur in the parallel corpus.

Figure 4 shows the effect of OOV ratio on the Chinese-to-English validation set. Only English monolingual corpus is appended to the parallel corpus during training. We constructed four monolingual corpora of the same size in terms of sentence pairs. "0% OOV" means the OOV ratio is 0% for all sentences in the monolingual corpus. "10% OOV" suggests that the OOV ratio is no greater 10% for each sentence in the monolingual corpus. We find that using a monolingual corpus with a lower OOV ratio generally leads to higher BLEU scores. One possible reason is that low-OOV monolingual corpus is relatively easier to reconstruct than its high-OOV counterpart and results in better estimation of model parameters.

Figure 5 shows the effect of OOV ratio on the English-to-Chinese validation set. Only English monolingual corpus is appended to the parallel corpus during training. We find that "0% OOV" still achieves the highest BLEU scores.

3.4 Comparison with SMT

Table 2 shows the comparison between MOSES and our work. MOSES used the monolingual corpora as shown in Table 1: 18.75M Chinese sentences and 22.32M English sentences. We find that exploiting monolingual corpora dramatically improves translation performance in both Chinese-to-English and English-to-Chinese directions.

Relying only on parallel corpus, RNNSEARCH outperforms MOSES trained also only on parallel corpus. But the capability of making use of abundant monolingual corpora enables MOSES to achieve much higher BLEU scores than RNNSEARCH only using parallel corpus.

Instead of using all sentences in the monolingual corpora, we constructed smaller monolingual corpora with zero OOV ratio: 2.56M Chinese sentences with 47.51M words and 2.56M English English sentences with 37.47M words. In other words, the monolingual corpora we used in the experiments are much smaller than those used by MOSES.

By adding English monolingual corpus, our approach achieves substantial improvements over RNNSEARCH using only parallel corpus (up to +4.7 BLEU points). In addition, significant improvements are also obtained over MOSES using both parallel and monolingual corpora (up to +3.5 BLEU points).

An interesting finding is that adding English monolingual corpora helps to improve English-to-Chinese translation over RNNSEARCH using only parallel corpus (up to +3.2 BLEU points), suggesting that our approach is capable of improving NMT using source-side monolingual corpora.

In the English-to-Chinese direction, we obtain similar findings. In particular, adding Chi-

System	Training Data			Direction	NIST06	NIST02	NIST03	NIST04	NIST05
	CE	C	E						
MOSES	√	×	×	C → E	32.48	32.69	32.39	33.62	30.23
				E → C	14.27	18.28	15.36	13.96	14.11
	√	×	√	C → E	34.59	35.21	35.71	35.56	33.74
	√	√	×	E → C	20.69	25.85	19.76	18.77	19.74
RNNSEARCH	√	×	×	C →E	30.74	35.16	33.75	34.63	31.74
				E →C	15.71	20.76	16.56	16.85	15.14
	√	×	√	C → E	35.61^{**++}	38.78^{**++}	38.32^{**++}	38.49^{**++}	36.45^{**++}
				E →C	17.59^{++}	23.99^{++}	18.95^{++}	18.85^{++}	17.91^{++}
	√	√	×	C →E	35.01^{++}	38.20^{**++}	37.99^{**++}	38.16^{**++}	36.07^{**++}
				E →C	21.12^{*++}	29.52^{*++}	20.49^{*++}	21.59^{*++}	19.97^{++}

Table 2: Comparison with MOSES and RNNSEARCH. MOSES is a phrase-based statistical machine translation system (Koehn et al., 2007). RNNSEARCH is an attention-based neural machine translation system (Bahdanau et al., 2015). "CE" donates Chinese-English parallel corpus, "C" donates Chinese monolingual corpus, and "E" donates English monolingual corpus. "√" means the corpus is included in the training data and × means not included. "NIST06" is the validation set and "NIST02-05" are test sets. The BLEU scores are case-insensitive. "*": significantly better than MOSES ($p < 0.05$); "**": significantly better than MOSES ($p < 0.01$);"+": significantly better than RNNSEARCH ($p < 0.05$); "++": significantly better than RNNSEARCH ($p < 0.01$).

Method	Training Data			Direction	NIST06	NIST02	NIST03	NIST04	NIST05
	CE	C	E						
Sennrich et al. (2015)	√	×	√	C →E	34.10	36.95	36.80	37.99	35.33
	√	√	×	E →C	19.85	28.83	20.61	20.54	19.17
this work	√	×	√	C →E	35.61^{**}	38.78^{**}	38.32^{**}	38.49^{*}	36.45^{**}
				E →C	17.59	23.99	18.95	18.85	17.91
	√	√	×	C →E	35.01^{**}	38.20^{**}	37.99^{**}	38.16	36.07^{**}
				E →C	21.12^{**}	29.52^{**}	20.49	21.59^{**}	19.97^{**}

Table 3: Comparison with Sennrich et al. (2015). Both Sennrich et al. (2015) and our approach build on top of RNNSEARCH to exploit monolingual corpora. The BLEU scores are case-insensitive. "*": significantly better than Sennrich et al. (2015) ($p < 0.05$); "**": significantly better than Sennrich et al. (2015) ($p < 0.01$).

nese monolingual corpus leads to more benefits to English-to-Chinese translation than adding English monolingual corpus. We also tried to use both Chinese and English monolingual corpora through simply setting all the λ to 0.1 but failed to obtain further significant improvements.

Therefore, our findings can be summarized as follows:

1. Adding target monolingual corpus improves over using only parallel corpus for source-to-target translation;
2. Adding source monolingual corpus also improves over using only parallel corpus for source-to-target translation, but the improvements are smaller than adding target monolingual corpus;
3. Adding both source and target monolingual corpora does not lead to further significant improvements.

3.5 Comparison with Previous Work

We re-implemented Sennrich et al. (2015)'s method on top of RNNSEARCH as follows:

1. Train the target-to-source neural translation model $P(\mathbf{x}|\mathbf{y}; \overleftarrow{\boldsymbol{\theta}})$ on the parallel corpus $D = \{\langle \mathbf{x}^{(n)}, \mathbf{y}^{(n)} \rangle\}_{n=1}^{N}$.
2. The trained target-to-source model $\overleftarrow{\boldsymbol{\theta}}^{*}$ is used to translate a target monolingual corpus $\mathcal{T} = \{\mathbf{y}^{(t)}\}_{t=1}^{T}$ into a source monolingual corpus $\tilde{\mathcal{S}} = \{\tilde{\mathbf{x}}^{(t)}\}_{t=1}^{T}$.
3. The target monolingual corpus is paired with its translations to form a pseudo parallel corpus, which is then appended to the original parallel corpus to obtain a larger parallel corpus: $\tilde{\mathcal{D}} = \mathcal{D} \cup \langle \tilde{\mathcal{S}}, \mathcal{T} \rangle$.
4. Re-train the the source-to-target neural translation model on $\tilde{\mathcal{D}}$ to obtain the final model parameters $\overrightarrow{\boldsymbol{\theta}}^{*}$.

Monolingual	hongsen shuo , ruguo you na jia famu gongsi dangan yishenshifa , name tamen jiang zihui qiancheng .
Reference	hongsen said, if any *logging companies* dare to defy the law, then they will *destroy their own future* .
Translation	hun sen said , if any of *those companies* dare defy the law , then they will *have their own fate* . [*iteration 0*]
	hun sen said if any *tree felling company* dared to break the law , then they would *kill themselves* . [*iteration 40K*]
	hun sen said if any *logging companies* dare to defy the law , they would *destroy the future themselves* . [*iteration 240K*]

Monolingual	dan yidan panjue jieguo zuizhong queding , ze bixu zai 30 tian nei zhixing .
Reference	But once *the final verdict is confirmed* , it must be executed within 30 days .
Translation	however , *in the final analysis* , it must be carried out within 30 days . [*iteration 0*]
	however , *in the final analysis* , the final decision will be carried out within 30 days . [*iteration 40K*]
	however , once *the verdict is finally confirmed* , it must be carried out within 30 days . [*iteration 240K*]

Table 4: Example translations of sentences in the monolingual corpus during semi-supervised learning. We find our approach is capable of generating better translations of the monolingual corpus over time.

Table 3 shows the comparison results. Both the two approaches use the same parallel and monolingual corpora. Our approach achieves significant improvements over Sennrich et al. (2015) in both Chinese-to-English and English-to-Chinese directions (up to +1.8 and +1.0 BLEU points). One possible reason is that Sennrich et al. (2015) only use the pesudo parallel corpus for parameter estimation for once (see Step 4 above) while our approach enables source-to-target and target-to-source models to interact with each other iteratively on both parallel and monolingual corpora.

To some extent, our approach can be seen as an iterative extension of Sennrich et al. (2015)'s approach: after estimating model parameters on the pseudo parallel corpus, the learned model parameters are used to produce a better pseudo parallel corpus. Table 4 shows example Viterbi translations on the Chinese monolingual corpus over iterations:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \left\{ P(\mathbf{y}'|\mathbf{x}; \overrightarrow{\boldsymbol{\theta}}) P(\mathbf{x}|\mathbf{y}; \overleftarrow{\boldsymbol{\theta}}) \right\}. \quad (11)$$

We observe that the quality of Viterbi translations generally improves over time.

4 Related Work

Our work is inspired by two lines of research: (1) exploiting monolingual corpora for machine translation and (2) autoencoders in unsupervised and semi-supervised learning.

4.1 Exploiting Monolingual Corpora for Machine Translation

Exploiting monolingual corpora for conventional SMT has attracted intensive attention in recent years. Several authors have introduced transductive learning to make full use of monolingual corpora (Ueffing et al., 2007; Bertoldi and Federico, 2009). They use an existing translation model to translate unseen source text, which can be paired with its translations to form a pseudo parallel corpus. This process iterates until convergence. While Klementiev et al. (2012) propose an approach to estimating phrase translation probabilities from monolingual corpora, Zhang and Zong (2013) directly extract parallel phrases from monolingual corpora using retrieval techniques. Another important line of research is to treat translation on monolingual corpora as a decipherment problem (Ravi and Knight, 2011; Dou et al., 2014).

Closely related to Gulccehre et al. (2015) and Sennrich et al. (2015), our approach focuses on learning birectional NMT models via autoencoders on monolingual corpora. The major advantages of our approach are the transparency to network architectures and the capability to exploit both source and target monolingual corpora.

4.2 Autoencoders in Unsupervised and Semi-Supervised Learning

Autoencoders and their variants have been widely used in unsupervised deep learning ((Vincent et al., 2010; Socher et al., 2011; Ammar et al., 2014), just to name a few). Among them, Socher et al. (2011)'s approach bears close resemblance to our approach as they introduce semi-supervised recursive autoencoders for sentiment analysis. The difference is that we are interested in making a better use of parallel and monolingual corpora while they concentrate on injecting partial supervision to conventional unsupervised autoencoders. Dai and Le (2015) introduce a sequence autoencoder to reconstruct an observed sequence via RNNs. Our approach differs from sequence autoencoders in that we use bidirectional translation models as encoders and decoders to enable them to interact within the autoencoders.

5 Conclusion

We have presented a semi-supervised approach to training bidirectional neural machine translation models. The central idea is to introduce autoencoders on the monolingual corpora with source-to-target and target-to-source translation models as encoders and decoders. Experiments on Chinese-English NIST datasets show that our approach leads to significant improvements.

As our method is sensitive to the OOVs present in monolingual corpora, we plan to integrate Jean et al. (2015)'s technique on using very large vocabulary into our approach. It is also necessary to further validate the effectiveness of our approach on more language pairs and NMT architectures. Another interesting direction is to enhance the connection between source-to-target and target-to-source models (e.g., letting the two models share the same word embeddings) to help them benefit more from interacting with each other.

Acknowledgements

This work was done while Yong Cheng was visiting Baidu. This research is supported by the 973 Program (2014CB340501, 2014CB340505), the National Natural Science Foundation of China (No. 61522204, 61331013, 61361136003), 1000 Talent Plan grant, Tsinghua Initiative Research Program grants 20151080475 and a Google Faculty Research Award. We sincerely thank the viewers for their valuable suggestions.

References

Waleed Ammar, Chris Dyer, and Noah Smith. 2014. Conditional random field autoencoders for unsupervised structred prediction. In *Proceedings of NIPS 2014*.

Dzmitry Bahdanau, KyungHyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of ICLR*.

Nicola Bertoldi and Marcello Federico. 2009. Domain adaptation for statistical machine translation. In *Proceedings of WMT*.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguisitics*.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of ACL*.

Kyunhyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In *Proceedings of SSST-8*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proceedings of NIPS*.

Qing Dou, Ashish Vaswani, and Kevin Knight. 2014. Beyond parallel data: Joint word alignment and decipherment improves machine translation. In *Proceedings of EMNLP*.

Caglar Gulccehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. arXiv:1503.03535 [cs.CL].

Sepp Hochreiter and Jürgen Schmidhuber. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguisitics*.

Sebastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of ACL*.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of EMNLP*.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*.

Alexandre Klementiev, Ann Irvine, Chris Callison-Burch, and David Yarowsky. 2012. Toward statistical machine translation without paralel corpora. In *Proceedings of EACL*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL (demo session)*.

Minh-Thang Luong, Ilya Sutskever, Quoc V. Le, Oriol Vinyals, and Wojciech Zaremba. 2015. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*.

Franz Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of ACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a methof for automatic evaluation of machine translation. In *Proceedings of ACL*.

Sujith Ravi and Kevin Knight. 2011. Deciphering foreign language. In *Proceedings of ACL*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving nerual machine translation models with monolingual data. arXiv:1511.06709 [cs.CL].

Richard Socher, Jeffrey Pennington, Eric Huang, Andrew Ng, and Christopher Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*.

Andreas Stolcke. 2002. Srilm - am extensible language modeling toolkit. In *Proceedings of ICSLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*.

Nicola Ueffing, Gholamreza Haffari, and Anoop Sarkar. 2007. Trasductive learning for statistical machine translation. In *Proceedings of ACL*.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Autoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*.

Jiajun Zhang and Chengqing Zong. 2013. Learning a phrase-based translation model from monolingual data with application to domain adaptation. In *Proceedings of ACL*.

Strategies for Training Large Vocabulary Neural Language Models

Wenlin Chen **David Grangier** **Michael Auli**
Facebook, Menlo Park, CA

Abstract

Training neural network language models over large vocabularies is computationally costly compared to count-based models such as Kneser-Ney. We present a systematic comparison of neural strategies to represent and train large vocabularies, including softmax, hierarchical softmax, target sampling, noise contrastive estimation and self normalization. We extend self normalization to be a proper estimator of likelihood and introduce an efficient variant of softmax. We evaluate each method on three popular benchmarks, examining performance on rare words, the speed/accuracy trade-off and complementarity to Kneser-Ney.

1 Introduction

Neural network language models (Bengio et al., 2003; Mikolov et al., 2010) have gained popularity for tasks such as automatic speech recognition (Arisoy et al., 2012) and statistical machine translation (Schwenk et al., 2012; Vaswani et al., 2013; Baltescu and Blunsom, 2014). Similar models are also developed for translation (Le et al., 2012; Devlin et al., 2014; Bahdanau et al., 2015), summarization (Chopra et al., 2015) and language generation (Sordoni et al., 2015).

Language models assign a probability to a word given a context of preceding, and possibly subsequent, words. The model architecture determines how the context is represented and there are several choices including recurrent neural networks (Mikolov et al., 2010; Jozefowicz et al., 2016), or log-bilinear models (Mnih and Hinton, 2010). This paper does not focus on architecture or context representation but rather on how to efficiently deal with large output vocabularies, a problem common to all approaches to neural language modeling and related tasks (machine translation, language generation). We therefore experiment with a classical feed-forward neural network model similar to Bengio et al. (2003).

Practical training speed for these models quickly decreases as the vocabulary grows. This is due to three combined factors: (i) model evaluation and gradient computation become more time consuming, mainly due to the need of computing normalized probabilities over a large vocabulary; (ii) large vocabularies require more training data in order to observe enough instances of infrequent words which increases training times; (iii) a larger training set often allows for larger models which requires more training iterations.

This paper provides an overview of popular strategies to model large vocabularies for language modeling. This includes the classical *softmax* over all output classes, *hierarchical softmax* which introduces latent variables, or clusters, to simplify normalization, target sampling which only considers a random subset of classes for normalization, *noise contrastive estimation* which discriminates between genuine data points and samples from a noise distribution, and *infrequent normalization*, also referred as self-normalization, which computes the partition function at an infrequent rate. We also extend self-normalization to be a proper estimator of likelihood. Furthermore, we introduce *differentiated softmax*, a novel variation of softmax which assigns more parameters, or capacity, to frequent words and which we show to be faster and more accurate than softmax (§2).

Our comparison assumes a reasonable budget of one week for training models on a high end GPU (Nvidia K40). We evaluate on three benchmarks differing in the amount of training data and vocabulary size, that is Penn Treebank, Gigaword and the Billion Word benchmark (§3).

Our results show that conclusions drawn from small datasets do not always generalize to larger settings. For instance, hierarchical softmax is less accurate than softmax on the small vocabulary Penn Treebank task but performs best on the very large vocabulary Billion Word benchmark. This is because hierarchical softmax is the fastest method for training and can perform more training updates in the same period of time. Furthermore, our re-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1975–1985,
Berlin, Germany, August 7-12, 2016.

sults with differentiated softmax demonstrate that assigning capacity where it has the most impact allows to train better models in our time budget (§4). Our analysis also shows clearly that traditional Kneser-Ney models are competitive on rare words, contrary to the common belief that neural models are better on infrequent words (§5).

2 Modeling Large Vocabularies

We first introduce our model architecture with a classical softmax and then describe various other methods including a novel variation of softmax.

2.1 Softmax Neural Language Model

Our feed-forward neural network implements an n-gram language model, i.e., it is a parametric function estimating the probability of the next word w^t given $n-1$ previous context words, $w^{t-1}, \ldots, w^{t-n+1}$. Formally, we take as input a sequence of discrete indexes representing the $n-1$ previous words and output a vocabulary-sized vector of probability estimates, i.e.,

$$f : \{1, \ldots, V\}^{n-1} \rightarrow [0,1]^V,$$

where V is the vocabulary size. This function results from the composition of simple differentiable functions or *layers*.

Specifically, f composes an input mapping from discrete word indexes to continuous vectors, a succession of linear operations followed by hyperbolic tangent non-linearities, plus one final linear operation, followed by a softmax normalization.

The input layer maps each context word index to a continuous d'_0-dimensional vector. It relies on a matrix $W^0 \in \mathbb{R}^{V \times d'_0}$ to convert the input

$$x = [w^{t-1}, \ldots, w^{t-n+1}] \in \{1, \ldots, V\}^{n-1}$$

to $n-1$ vectors of dimension d'_0. These vectors are concatenated into a single $(n-1) \times d'_0$ matrix,

$$h^0 = [W^0_{w^{t-1}}; \ldots; W^0_{w^{t-n+1}}] \in \mathbb{R}^{n-1 \times d'_0}.$$

This state h^0 is considered as a $d_0 = (n-1) \times d'_0$ vector by the next layer. The subsequent states are computed through k layers of linear mappings followed by hyperbolic tangents, i.e.

$$\forall i = 1, \ldots, k, \quad h^i = \tanh(W^i h^{i-1} + b^i) \in \mathbb{R}^{d_i}$$

where $W^i \in \mathbb{R}^{d_i \times d_{i-1}}, b \in \mathbb{R}^{d_i}$ are learnable weights and biases and $\tanh$ denotes the component-wise hyperbolic tangent.

Finally, the last layer performs a linear operation followed by a softmax normalization, i.e.,

$$\begin{aligned} & h^{k+1} = W^{k+1} h^k + b^{k+1} \in \mathbb{R}^V \\ \text{and} \quad & y = \tfrac{1}{Z} \exp(h^{k+1}) \in [0,1]^V \end{aligned} \tag{1}$$

where $Z = \sum_{j=1}^{V} \exp(h^{k+1}_j)$ and $\exp$ denotes the component-wise exponential. The network output y is therefore a vocabulary-sized vector of probability estimates. We use the standard cross-entropy loss with respect to the computed log probabilities

$$\frac{\partial \log y_i}{\partial h^{k+1}_j} = \delta_{ij} - y_j$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise The gradient update therefore increases the score of the correct output h^{k+1}_i and decreases the score of all other outputs h^{k+1}_j for $j \neq i$.

A downside of the classical softmax formulation is that it requires computation of the activations for *all output words*, Eq. (1). The output layer with V activations is much larger than any other layer in the network and its matrix multiplication dominates the complexity of the entire network.

2.2 Hierarchical Softmax

Hierarchical Softmax (HSM) organizes the output vocabulary into a tree where the leaves are the words and the intermediate nodes are latent variables, or *classes* (Morin and Bengio, 2005). The tree has potentially many levels and there is a unique path from the root to each word. The probability of a word is the product of the probabilities of the latent variables along the path from the root to the leaf, including the probability of the leaf.

We follow Goodman (2001) and Mikolov et al. (2011b) and model a two-level tree. Given context x, HSM predicts the *class* of the next word c^t and the actual word w^t

$$p(w^t|x) = p(c^t|x)\, p(w^t|c^t, x) \tag{2}$$

If the number of classes is $\mathcal{O}(\sqrt{V})$ and classes are balanced, then we only need to compute $\mathcal{O}(2\sqrt{V})$ outputs. In practice, this strategy results in weight matrices whose largest dimension is $< 1,000$, a setting for which GPU hardware is fast.

A popular strategy is *frequency* clustering. It sorts the vocabulary by frequency and then forms clusters of words with similar frequency. Each cluster contains an equal share of the total unigram probability. We compare this strategy to random class assignment and to clustering based on word

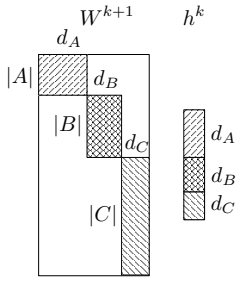


Figure 1: Output weight matrix W^{k+1} and hidden layer h^k for differentiated softmax for vocabulary partitions A, B, C with embedding dimensions d_A, d_B, d_C; non-shaded areas are zero.

contexts, relying on PCA (Lebret and Collobert, 2014). A full comparison of context-based clustering is beyond the scope of this work (Brown et al., 1992; Mikolov et al., 2013).

2.3 Differentiated Softmax

This section introduces a novel variation of softmax that assigns a variable number of parameters to each word in the output layer. The weight matrix of the final layer $W^{k+1} \in \mathbb{R}^{d_k \times V}$ stores *output embeddings* of size d_k for the V words the language model may predict: $W_1^{k+1}; \ldots; W_V^{k+1}$. Differentiated softmax (D-Softmax) varies the dimension of the output embeddings d_k across words depending on how much model capacity, or parameters, are deemed suitable for a given word. We assign more parameters to frequent words than to rare words since more training occurrences allow for fitting more parameters.

We partition the output vocabulary based on word frequency and the words in each partition share the same embedding size. Partitioning the vocabulary in this way results in a sparse final weight matrix W^{k+1} which arranges the embeddings of the output words in blocks, each block corresponding to a separate partition (Figure 1). The size of the final hidden layer h^k is the sum of the embedding sizes of the partitions. The final hidden layer is effectively a concatenation of separate features for each partition which are used to compute the dot product with the corresponding embedding type in W^{k+1}. In practice, we efficiently compute separate matrix-vector products, or in batched form, matrix-matrix products, for each partition in W^{k+1} and h^k.

Overall, differentiated softmax can lead to large speed-ups as well as accuracy gains since we can greatly reduce the complexity of computing the output layer. Most significantly, this strategy speeds up *both* training and inference. This is in contrast to hierarchical softmax which is fast during training but requires even more effort than softmax for computing the most likely next word.

2.4 Target Sampling

Sampling-based methods approximate the softmax normalization, Eq. (1), by summing over a sub-sample of impostor classes. This can significantly speed-up each training iteration, depending on the size of the impostor set. Target sampling builds upon the importance sampling work of Bengio and Senécal (2008). We follow Jean et al. (2014) who choose as impostors all positive examples in a mini-batch as well as a subset of the remaining words. This subset is sampled uniformly and its size is chosen by validation.

2.5 Noise Contrastive Estimation

Noise contrastive estimation (NCE) is another sampling-based technique (Hyvärinen, 2010; Mnih and Teh, 2012; Chen et al., 2015). Contrary to target sampling, it does not maximize the training data likelihood directly. Instead, it solves a two-class problem of distinguishing genuine data from noise samples. The training algorithm samples a word w given the preceding context x from a mixture

$$p(w|x) = \frac{1}{k+1} p_{\text{train}}(w|x) + \frac{k}{k+1} p_{\text{noise}}(w|x)$$

where p_{train} is the empirical distribution of the training set and p_{noise} is a known noise distribution which is typically a context-independent unigram distribution. The training algorithm fits the model $\hat{p}(w|x)$ to recover whether a mixture sample came from the data or the noise distribution, this amounts to minimizing the binary cross-entropy $-y \log \hat{p}(y=1|w,x) - (1-y) \log \hat{p}(y=0|w,x)$ where y is a binary variable indicating where the current sample originates from

$$\begin{cases} \hat{p}(y=1|w,x) = \frac{\hat{p}(w|x)}{\hat{p}(w|x) + k p_{\text{noise}}(w|x)} & \text{(data)} \\ \hat{p}(y=0|w,x) = 1 - \hat{p}(y=1|w,x) & \text{(noise).} \end{cases}$$

This formulation still involves a softmax over the vocabulary to compute $\hat{p}(w|x)$. However, Mnih and Teh (2012) suggest to forego normalization and replace $\hat{p}(w|x)$ with unnormalized exponentiated scores. This makes the training complexity independent of the vocabulary size. At test time, softmax normalization is reintroduced to get a proper distribution. We also follow Mnih and Teh (2012) recommendations for p_{noise} and rely on a unigram distribution of the training set.

2.6 Infrequent Normalization

Devlin et al. (2014), followed by Andreas and Klein (2015), proposed to relax score normalization. Their strategy (here referred to as WeaknormSQ) associates unnormalized likelihood maximization with a penalty term that favors normalized predictions. This yields the following loss over the training set T

$$L_{\alpha}^{(2)} = -\sum_{(w,x)\in T} s(w|x) + \alpha \sum_{(w,x)\in T} (\log Z(x))^2$$

where $s(w|x)$ refers to the unnormalized score of word w given context x and $Z(x) = \sum_w \exp(s(w|x))$ refers to the partition function for context x. This strategy therefore pushes the log partition towards zero. For efficient training, the second term can be down-sampled

$$L_{\alpha,\gamma}^{(2)} = -\sum_{(w,x)\in T} s(w|x) + \frac{\alpha}{\gamma} \sum_{(w,x)\in T_\gamma} (\log Z(x))^2$$

where T_γ is the training set sampled at rate γ. A small rate implies computing the partition function only for a small fraction of the training data.

We extend this strategy to the case where the log partition term is not squared (Weaknorm), i.e.,

$$L_{\alpha,\gamma}^{(1)} = -\sum_{(w,x)\in T} s(w|x) + \frac{\alpha}{\gamma} \sum_{(w,x)\in T_\gamma} \log Z(x)$$

For $\alpha = 1$, this loss is an unbiased estimator of the negative log-likelihood of the training data $L_1^{(2)} = -\sum_{(w,x)\in T} s(w|x) + \log Z(x)$.

3 Experimental Setup

Datasets We run experiments over three news datasets of different sizes: Penn Treebank (PTB), WMT11-lm (billionW) and English Gigaword, version 5 (gigaword). Penn Treebank (Marcus et al., 1993) is the smallest corpus with 1M tokens and we use a vocabulary size of 10k (Mikolov et al., 2011a). The billion word benchmark (Chelba et al., 2013) comprises almost one billion tokens and a vocabulary of about 800k words[1]. Gigaword (Parker et al., 2011) is even larger with 5 billion tokens and was previously used for language modeling (Heafield, 2011) but there is no standard train/test split or vocabulary for this set. We split according to time: training covers 1994–2009 and test covers 2010. The vocabulary comprises the 100k most frequent words in train. Table 1 summarizes the data statistics.

[1]T. Robinson version `http://tiny.cc/1billionLM`.

Dataset	Train	Test	Vocab	OOV
PTB	1M	0.08M	10k	5.8%
gigaword	4,631M	279M	100k	5.6%
billionW	799M	8.1M	793k	0.3%

Table 1: Dataset statistics. Number of tokens for train and test, vocabulary size, fraction of OOV.

Evaluation We measure perplexity on the test set. For PTB and billionW, we report results on a per sentence basis, i.e., models do not use context words across sentence boundaries and we score end-of-sentence markers. This is the standard setting for these benchmarks and allows comparison with other work. On gigaword, we use contexts across sentence boundaries and evaluation does not include end-of-sentence markers.

Our baseline is an interpolated Kneser-Ney (KN) model. We use KenLM (Heafield, 2011) to train 5-gram models without pruning. For neural models, we train 11-gram models for gigaword and billionW; for PTB we train a 6-gram model. The model parameters (weights W^i and biases b^i for $i = 0, \ldots, k+1$) are learned to maximize the training log-likelihood relying on stochastic gradient descent (SGD; LeCun et al.. 1998).

Validation Hyper-parameters are the number of layers k and the dimension of each layer $d_i, \forall i = 0, \ldots, k$. We tune the following settings for each technique on the validation set: the number of clusters, the clustering technique for hierarchical softmax, the number of frequency bands and their allocated capacity for differentiated softmax, the number of distractors for target sampling, the noise/data ratio for NCE, as well as the regularization rate and strength for infrequent normalization. Similarly, SGD parameters (learning rate and mini-batch size) are set to maximize validation likelihood. We also tune the dropout rate (Srivastava et al., 2014); dropout is employed after each tanh non-linearity.[2]

Training Time We train for 168 hours (one week) on the large datasets (billionW, gigaword) and 24 hours (one day) for Penn Treebank. All experiments are performed on the same hardware, a single K40 GPU. We select the hyper-parameters which yield the best validation perplexity after the allocated time and report the perplexity of the resulting model on the test set. This training time

[2]More parameter settings are available in an extended version of the paper at `http://arxiv.org/abs/1512.04906`.

is a trade-off between being able to do a comprehensive exploration of the various settings for each method and good accuracy. The chosen training times are not long enough to observe over-fitting, i.e. validation performance is still improving – albeit very slowly – at the end of the training session. As a general observation, even on the small PTB where 24 hours is rather long, we always found better results using the full training time, possibly increasing the dropout rate.

A concern may be that a fixing the training time favors models with better implementations. However, all models are very similar and their core computations are always matrix/matrix products. Training differs mostly in the size and frequency of large matrix/matrix products. Matrix products rely on CuBLAS[3], using torch[4]. For the matrix sizes involved ($> 500 \times 1,000$), the time complexity of matrix product is linear in each dimension, both on CPU (Intel MKL[5]) and GPU (CuBLAS), with a $10X$ speedup for GPU (Nvidia K40) compared to CPU (Intel Xeon E5-2680). Therefore, the speed trade-off applies to both CPU and GPU hardware, albeit with a different time scale.

4 Results

The test perplexities (Table 2) and validation learning curves (Figures 2, 3, and 4) show that the competitiveness of softmax diminishes with larger vocabularies. Softmax does well on the small vocabulary PTB but poorly on the large vocabulary billionW corpus. Faster methods such as sampling, hierarchical softmax, and infrequent normalization (Weaknorm, WeaknormSQ) are much better in the large vocabulary setting of billionW.

D-Softmax is performing well on all sets and shows that assigning higher capacity where it benefits most results in better models. Target sampling performs worse than softmax on gigaword but better on billionW. Hierarchical softmax performs poorly on Penn Treebank which is in stark contrast to billionW where it does well. Noise contrastive estimation has good accuracy on billionW, where speed is essential to achieving good accuracy.

Of all the methods, hierarchical softmax processes most training examples in a given time frame (Table 3). Our test time speed comparison assumes that we would like to find the highest scoring next word rather than rescoring an existing string. This scenario requires scoring all output words and D-Softmax can process nearly twice as many tokens per second than the other methods whose complexity is similar to softmax.

	PTB	gigaW	billionW
KN	141.2	57.1	70.2[6]
Softmax	123.8	56.5	108.3
D-Softmax	**121.1**	**52.0**	91.2
Sampling	124.2	57.6	101.0
HSM	138.2	57.1	**85.2**
NCE	143.1	78.4	104.7
Weaknorm	124.4	56.9	98.7
WeaknormSQ	122.1	56.1	94.9
KN+Softmax	108.5	43.6	59.4
KN+D-Softmax	**107.0**	**42.0**	56.3
KN+Sampling	109.4	43.8	58.1
KN+HSM	115.0	43.9	**55.6**
KN+NCE	114.6	49.0	58.8
KN+Weaknorm	109.2	43.8	58.1
KN+WeaknormSQ	108.8	43.8	57.7

Table 2: Test perplexity of individual models and interpolation with Kneser-Ney.

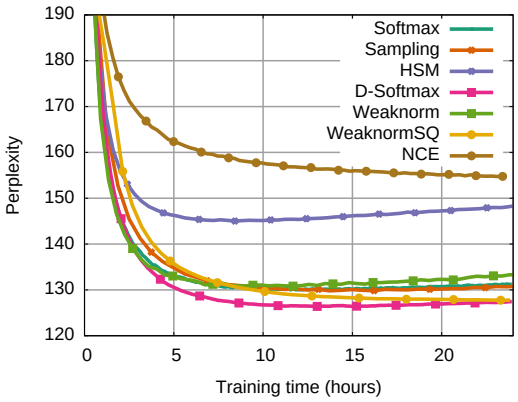


Figure 2: PTB validation learning curve.

4.1 Softmax

Despite being our baseline, softmax ranks among the most accurate methods on PTB and it is second best on gigaword after D-Softmax (with WeaknormSQ performing similarly). For billionW, the extremely large vocabulary makes softmax training too slow to compete with faster alterna-

[3]http://docs.nvidia.com/cuda/cublas/
[4]http://torch.ch
[5]https://software.intel.com/en-us/intel-mkl

[6]This perplexity is higher than reported in (Chelba et al., 2013), in which Kneser Ney is not trained on the 800m token training set, but on a larger corpus of 1.1B tokens.

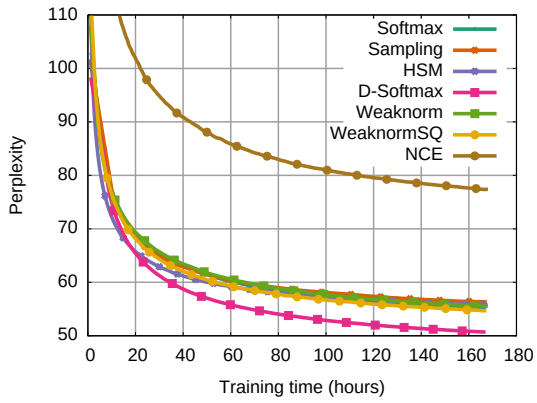


Figure 3: Gigaword validation learning curve.

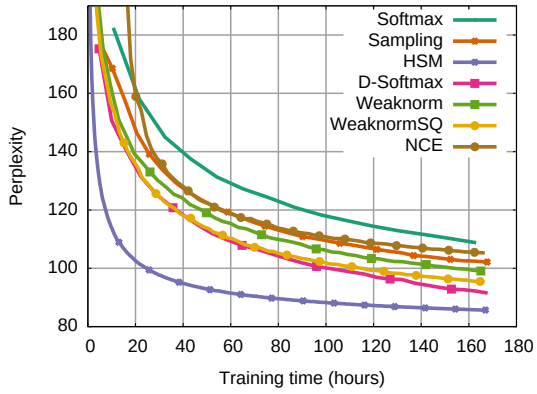


Figure 4: Billion Word validation learning curve.

	train	test
Softmax	510	510
D-Softmax	960	**960**
Sampling	1,060	510
HSM	**12,650**	510
NCE	4,520	510
Weaknorm	1,680	510
WeaknormSQ	2,870	510

Table 3: Training and test speed on billionW in tokens per second for generation of the next word. Most techniques are identical to softmax at test time. HSM can be faster for rescoring.

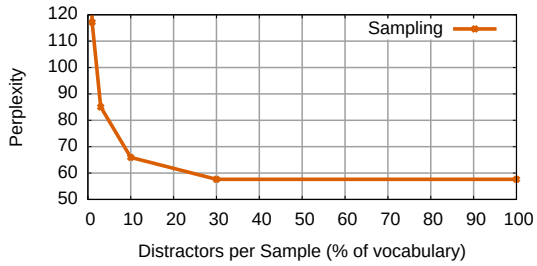


Figure 5: Number of Distractors versus Perplexity for Target Sampling over Gigaword

tives. However, of all the methods softmax has the simplest implementation and it has no additional hyper-parameters compared to other methods.

4.2 Target Sampling

Figure 5 shows that target sampling is most accurate for distractor sets that amount to a large fraction of the vocabulary, i.e. $> 30\%$ on gigaword (billionW best setting $> 50\%$ is even higher). Target sampling is faster and performs more iterations than softmax in the same time. However, its perplexity reduction per iteration is less than softmax. Overall, it is not much better than softmax. A reason might be that sampling chooses distractors independently from context and current model performance. This does not favor distractors the model incorrectly considers likely for the current context. These distractors would yield higher gradients that could update the model faster.

4.3 Hierarchical Softmax

Hierarchical softmax is very efficient for large vocabularies and it is the best method on billionW. On the other hand, HSM does poorly on small vocabularies as seen on PTB. We found that a good word clustering structure is crucial: when clusters gather words occurring in similar contexts, cluster likelihoods are easier to learn; when the cluster structure is uninformative, cluster likelihoods converge to the uniform distribution. This affects accuracy since words cannot have higher probability than their clusters, Eq. (2).

Our experiments organize words into a two level hierarchy and compare four clustering strategies on billionW and gigaword (§2.2). Random clustering shuffles the vocabulary and splits it into equally sized partitions. Frequency-based clustering first orders words based on their frequency and assigns words to clusters such that each cluster represents an equal share of the total frequency (Mikolov et al., 2011b). K-means runs the well-known clustering algorithm on Hellinger PCA word embeddings. Weighted k-means weights each word by its frequency.[7]

Random clusters perform worst (Table 4) followed by frequency-based clustering but k-means does best; weighted k-means performs similarly to its unweighted version. In earlier experiments, plain k-means performed very poorly since the most frequent cluster captured up to 40% of the

[7]The time to compute the clustering (multi-threaded word co-occurrence counts, PCA and k-means) is under one hour, which is negligible given a one week training budget.

	billionW	gigaword
random	98.51	62,27
frequency-based	92.02	59.47
k-means	85.70	57.52
weighted k-means	85.24	57.09

Table 4: HSM with different clustering.

token occurrences. We then explicitly capped the frequency budget of each cluster to 10% which brought k-means on par with weighted k-means.

4.4 Differentiated Softmax

D-Softmax is the best technique on gigaword and second best on billionW after HSM. On PTB it ranks among the best techniques whose perplexities cannot be reliably distinguished. The variable-capacity scheme of D-Softmax can assign large embeddings to frequent words, while keeping computational complexity manageable through small embeddings for rare words.

Unlike for hierarchical softmax, NCE or Weaknorm, the computational advantage of D-Softmax is preserved at test time (Table 3). D-Softmax is the fastest technique at test time, while ranking among the most accurate methods. This speed advantage is due to the low dimensional representation of rare words which negatively affects the model accuracy on these words (Table 5).

4.5 Noise Contrastive Estimation

Although we report better perplexities than the original NCE paper on PTB (Mnih and Teh, 2012), we found NCE difficult to use for large vocabularies. In order to work in this setting where models are larger, we had to dissociate the number of noise samples from the data to noise ratio in the modeled mixture. For instance, a data/noise ratio of $1/50$ gives good performance in our experiments but estimating only 50 noise sample posteriors per data point is wasteful given the cost of network evaluation. Moreover, 50 samples do not allow frequent sampling of every word in a large vocabulary. Our setting considers more noise samples and up-weights the data sample. This allows to set the data/noise ratio independently from the number of noise samples.

Overall, NCE results are better than softmax only for billionW, a setting for which softmax is very slow due to the very large vocabulary. Why does NCE perform so poorly? Figure 6 shows entropy on the validation set versus the NCE loss for several models. The results clearly show that similar NCE loss values can result in very different validation entropy. Although NCE might make sense for other metrics such as BLEU (Baltescu and Blunsom, 2014), it is not among the best techniques for minimizing perplexity. Jozefowicz et al. (2016) recently drew similar conclusions.

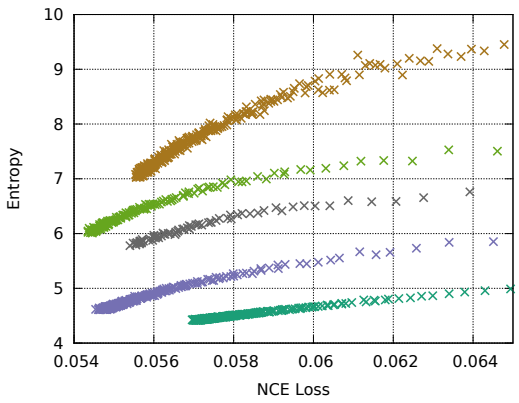


Figure 6: Validation entropy versus NCE loss on gigaword for experiments differing only in learning rates and initial weights. Each color corresponds to one experiment, with one point per hour.

4.6 Infrequent Normalization

Infrequent normalization (Weaknorm and WeaknormSQ) performs better than softmax on billionW and comparably to softmax on Penn Treebank and gigaword (Table 2). The speedup from skipping partition function computations is substantial. For instance, WeaknormSQ on billionW evaluates the partition only on 10% of the examples. In one week, the model is evaluated and updated on 868M tokens (with 86.8M partition evaluations) compared to 156M tokens for softmax.

Although referred to as self-normalizing (Andreas and Klein, 2015), the trained models still need normalization after training. The partition varies greatly between data samples. On billionW, the partition ranges between 9.4 to 10.3 in log scale for 10th to 90th percentile, i.e. a ratio of 2.5.

We observed the squared version (WeaknormSQ) to be unstable at times. Regularization strength could be found too low (collapse) or too high (blow-up) after a few days of training. We added an extra unit to bound unnormalized predictions $x \rightarrow 10 \tanh(x/5)$, which yields stable training and better generalization. For the non-squared Weaknorm, stability was not an issue. A regularization strength of 1 was the best setting for Weaknorm. This choice makes the loss an unbiased estimator of the data likelihood.

	1-4K	4-20K	20-40K	40-70K	70-100K
Kneser-Ney	3.48	7.85	9.76	10.76	11.57
Softmax	3.46	7.87	9.76	11.09	12.39
D-Softmax	**3.35**	7.79	10.13	12.22	12.69
Target sampling	3.51	**7.62**	9.51	10.81	12.06
HSM	3.49	7.86	**9.38**	**10.30**	**11.24**
NCE	3.74	8.48	10.60	12.06	13.37
Weaknorm	3.46	7.86	9.77	11.12	12.40
WeaknormSQ	3.46	7.79	9.67	10.98	12.32

Table 5: Test entropy on gigaword over subsets of the frequency ranked vocabulary; rank 1 is the most frequent word.

5 Analysis

5.1 Model Capacity

Training neural language models over large corpora highlights that training time, not training data, is the main factor limiting performance. The learning curves on gigaword and billionW indicate that most models are still making progress after one week. Training time has therefore to be taken into account when considering increasing capacity. Figure 7 shows validation perplexity versus the number of iterations for a week of training. This figure shows that a softmax model with 1024 hidden units in the last layer could perform better than the 512-hidden unit model with a longer training horizon. However, in the allocated time, 512 hidden units yield the best validation performance. D-softmax shows that it is possible to *selectively* increase capacity, i.e., to allocate more hidden units to the most frequent words at the expense of rarer words. This captures most of the benefit of a larger softmax model while staying within a reasonable training budget.

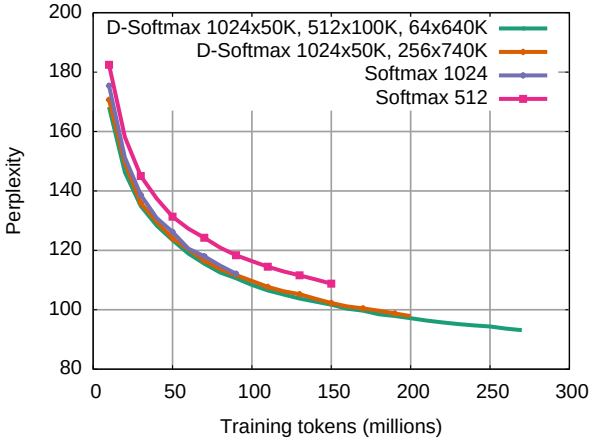


Figure 7: Validation perplexity per iteration on billionW for softmax and D-softmax. Softmax uses the same number of units for all words. The first D-Softmax experiment uses 1024 units for the 50K most frequent words, 512 for the next 100K, and 64 units for the rest; similarly for the second experiment. All experiments end after one week.

5.2 Effect of Initialization

We consider initializing both the input word embeddings and the output matrix from Hellinger PCA embeddings. Several alternative techniques for pre-training embeddings have been proposed (Mikolov et al., 2013; Lebret and Collobert, 2014; Pennington et al., 2014). Our experiment highlights the advantage of initialization and do not aim to compare embedding techniques.

Figure 8 shows that PCA is better than random for initializing both input and output word representations; initializing both from PCA is even better. We see that even after long training sessions, the initial conditions still impact the validation perplexity. We observed this trend also with other strategies than softmax. After one week of training, HSM is the only method which can reach comparable accuracy to PCA initialization when the output matrix is randomly initialized.

5.3 Training Set Size

Large training sets and a fixed training time introduce competition between slower models with more capacity and observing more training data. This trade-off only applies to iterative SGD optimization and does not apply to classical count-based models, which visit the training set once and then solve training in closed form.

We compare Kneser-Ney and softmax, trained for one week, with gigaword on differently sized subsets of the training data. For each setting we take care to include all data from the smaller subsets. Figure 9 shows that the performance of the neural model improves very little on more than

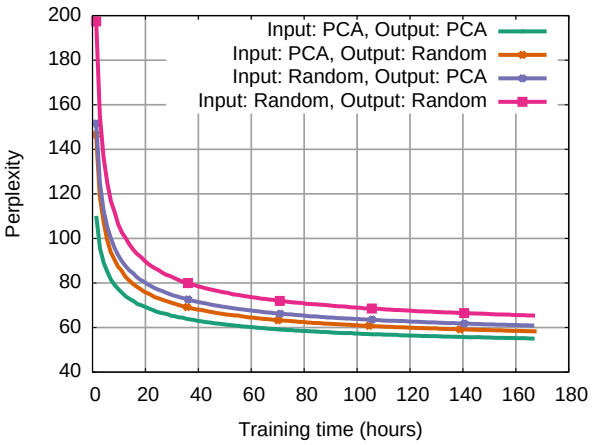


Figure 8: Effect of random initialization and with Hellinger PCA on gigaword for softmax.

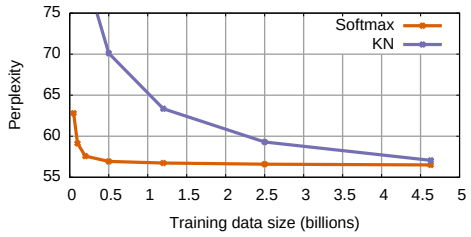


Figure 9: Effect of training set size measured on test of gigaword for Softmax and Kneser-Ney.

500M tokens. In order to benefit from the full training set we would require a much higher training budget, faster hardware, or parallelization.

Scaling training to large datasets can have a significant impact on perplexity, even when data from the distribution of interest is limited. As an illustration, we adapted a softmax model trained on billionW to Penn Treebank and achieved a perplexity of 96 - a far better result than with any model we trained from scratch on PTB (cf. Table 2).

5.4 Rare Words

How well do neural models perform on rare words? To answer this question, we computed entropy across word frequency bands for Kneser-Ney and neural models. Table 5 reports entropy for the $4,000$ most frequent words, then the next most frequent $16,000$ words, etc. For frequent words, neural models are on par or better than Kneser-Ney. For rare words, Kneser-Ney is very competitive. Although neural models might eventually close this gap with much longer training, one should consider that Kneser-Ney trains on gigaword in only 8 hours on CPU which contrasts with 168 hours of training for neural models on high end GPUs. This result highlights the complementarity of both approaches, as observed in our interpolation experiments (Table 2).

For neural models, D-Softmax excels on frequent words but performs poorly on rare ones. This is because D-Softmax assigns more capacity to frequent words at the expense of rare words. Overall, hierarchical softmax is the best neural technique for rare words. HSM does more iterations than any other technique and so it can observe every rare word more often.

6 Conclusions

This paper presents a comprehensive analysis of strategies to train neural language models with large vocabularies. This setting is very challenging for neural networks as they need to compute the partition function over the entire vocabulary at each evaluation.

We compared classical softmax to hierarchical softmax, target sampling, noise contrastive estimation and infrequent normalization, commonly referred to as self-normalization. Furthermore, we extend infrequent normalization to be a proper estimator of likelihood and we introduce differentiated softmax, a novel variant of softmax assigning less capacity to rare words to reduce computation.

Our results show that methods which are effective on small vocabularies are not necessarily equally so on large vocabularies. In our setting, target sampling and noise contrastive estimation failed to outperform the softmax baseline. Overall, differentiated softmax and hierarchical softmax are the best strategies for large vocabularies. Compared to classical Kneser-Ney models, neural models are better at modeling frequent words, but are less effective for rare words. A combination of the two is therefore very effective.

We conclude that there is a lot to explore in training from a combination of normalized and unnormalized objectives. An interesting future direction is to combine complementary approaches, either through combined parameterization (e.g. hierarchical softmax with differentiated capacity per word) or through a curriculum (e.g. transitioning from target sampling to regular softmax as training progresses). Further promising areas are parallel training as well as better rare word modeling.

References

Jacob Andreas and Dan Klein. 2015. When and why are log-linear models self-normalizing? In *Proc. of NAACL*.

Ebru Arisoy, Tara N. Sainath, Brian Kingsbury, and Bhuvana Ramabhadran. 2012. Deep Neural Network Language Models. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 20–28, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. of ICLR*. Association for Computational Linguistics, May.

Paul Baltescu and Phil Blunsom. 2014. Pragmatic neural language modelling in machine translation. Technical Report arXiv 1412.7119.

Yoshua Bengio and Jean-Sébastien Senécal. 2008. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research*, 3:1137–1155.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, Dec.

Ciprian Chelba, Tomáš Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. Technical report, Google.

Xie Chen, Xunying Liu, MJF Gales, and PC Woodland. 2015. Recurrent neural network language model training with noise contrastive estimation for speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP)*.

Sumit Chopra, Jason Weston, and Alexander M. Rush. 2015. Tuning as ranking. In *Proc. of EMNLP*. Association for Computational Linguistics, Sep.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, , and John Makhoul. 2014. Fast and Robust Neural Network Joint Models for Statistical Machine Translation. In *Proc. of ACL*. Association for Computational Linguistics, June.

Joshua Goodman. 2001. Classes for Fast Maximum Entropy Training. In *Proc. of ICASSP*.

Kenneth Heafield. 2011. KenLM: Faster and Smaller Language Model Queries. In *Workshop on Statistical Machine Translation*, pages 187–197.

Michael Gutmann Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. of AISTATS*.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2014. On Using Very Large Target Vocabulary for Neural Machine Translation. *CoRR*, abs/1412.2007.

Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. Technical Report arXiv 1602.02410.

Hai-Son Le, Alexandre Allauzen, and François Yvon. 2012. Continuous Space Translation Models with Neural Networks. In *Proc. of HLT-NAACL*, pages 39–48, Montréal, Canada. Association for Computational Linguistics.

Remi Lebret and Ronan Collobert. 2014. Word Embeddings through Hellinger PCA. In *Proc. of EACL*.

Yann LeCun, Leon Bottou, Genevieve Orr, and Klaus-Robert Mueller. 1998. Efficient BackProp. In Genevieve Orr and Klaus-Robert Muller, editors, *Neural Networks: Tricks of the trade*. Springer.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):314–330, Jun.

Tomáš Mikolov, Karafiát Martin, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent Neural Network based Language Model. In *Proc. of INTERSPEECH*, pages 1045–1048.

Tomáš Mikolov, Anoop Deoras, Stefan Kombrink, Lukas Burget, and Jan Honza Cernocky. 2011a. Empirical Evaluation and Combination of Advanced Language Modeling Techniques. In *Interspeech*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Cernocký, and Sanjeev Khudanpur. 2011b. Extensions of Recurrent Neural Network Language Model. In *Proc. of ICASSP*, pages 5528–5531.

Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *CoRR*, abs/1301.3781.

Andriy Mnih and Geoffrey E. Hinton. 2010. A Scalable Hierarchical Distributed Language Model. In *Proc. of NIPS*.

Andriy Mnih and Yee Whye Teh. 2012. A fast and simple algorithm for training neural probabilistic language models. In *Proc. of ICML*.

Frederic Morin and Yoshua Bengio. 2005. Hierarchical Probabilistic Neural Network Language Model. In *Proc. of AISTATS*.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition. Technical report, Linguistic Data Consortium.

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

Holger Schwenk, Anthony Rousseau, and Mohammed Attik. 2012. Large, Pruned or Continuous Space Language Models on a GPU for Statistical Machine Translation. In *NAACL-HLT Workshop on the Future of Language Modeling for HLT*, pages 11–19. Association for Computational Linguistics.

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie1, Jianfeng Gao, and Bill Dolan. 2015. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *Proc. of NAACL*. Association for Computational Linguistics, May.

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

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-scale Neural Language Models improves Translation. In *Proc. of EMNLP*. Association for Computational Linguistics, October.

Predicting the Compositionality of Nominal Compounds: Giving Word Embeddings a Hard Time

Silvio Cordeiro[1,2], **Carlos Ramisch**[1], **Marco Idiart**[3], **Aline Villavicencio**[2]
[1] Aix Marseille Université, CNRS, LIF UMR 7279 (France)
[2] Institute of Informatics, Federal University of Rio Grande do Sul (Brazil)
[3] Institute of Physics, Federal University of Rio Grande do Sul (Brazil)
silvioricardoc@gmail.com carlos.ramisch@lif.univ-mrs.fr
marco.idiart@gmail.com avillavicencio@inf.ufrgs.br

Abstract

Distributional semantic models (DSMs) are often evaluated on artificial similarity datasets containing single words or fully compositional phrases. We present a large-scale multilingual evaluation of DSMs for predicting the degree of semantic compositionality of nominal compounds on 4 datasets for English and French. We build a total of 816 DSMs and perform 2,856 evaluations using *word2vec*, *GloVe*, and PPMI-based models. In addition to the DSMs, we compare the impact of different parameters, such as level of corpus preprocessing, context window size and number of dimensions. The results obtained have a high correlation with human judgments, being comparable to or outperforming the state of the art for some datasets (Spearman's ρ=.82 for the *Reddy* dataset).

1 Introduction

Distributional semantic models (DSMs) use context information to represent the meaning of lexical units as vectors. They normally focus on the accurate semantic representation of single words. It is based on single words that many optimizations for these models have been proposed (Lin, 1999; Erk and Padó, 2010; Baroni and Lenci, 2010). This is particularly true for *word embeddings*, that is, a type of DSM where distributional vectors are obtained as a by-product of training a neural network to learn a function between words and their contexts (Mikolov et al., 2013a).

Simultaneously, there has been intensive research on models to compose individual word vectors in order to create representations for larger units such as phrases, sentences and even whole documents (Mitchell and Lapata, 2010; Mikolov et al., 2013a). Larger units can often be assumed to have their meanings derived from their parts according to the language's grammar, but this is not always the case (Sag et al., 2002). Many multiword units are associated with idiomatic interpretations, unrelated to the meaning of the component words (e.g. *silver bullet, eager beaver*).

Precision-oriented NLP applications need to be able to identify partly-compositional and idiomatic cases and ensure meaning preservation during processing. Compositionality identification is a first step towards complete semantic interpretation in tasks such as machine translation (to translate non-compositional compounds as a unit), word sense disambiguation (to avoid assigning a sense to parts of non-compositional compounds), and semantic parsing (to identify complex predicates and their arguments).

Even when larger units are explicitly represented in DSMs (McCarthy et al., 2003; Reddy et al., 2011; Mikolov et al., 2013c; Ferret, 2014), it is not clear whether the quality of these representations is comparable to the representations of single words. In particular, when building vectors for larger units, their generally lower frequencies in corpora (Kim and Baldwin, 2006) may combine with morphosyntactic phenomena to increase sparsity even further, often requiring non-trivial preprocessing (lemmatization and word reordering) to conflate variants.

This paper presents a large-scale multilingual evaluation of DSMs and their parameters for the task of compositionality prediction of nominal compounds in French and English. We examine parameters like the level of corpus preprocessing, the size of the context window and the number of dimensions for context representation. Additionally, we compare standard DSMs based on positive pointwise mutual information (PPMI)

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1986–1997,
Berlin, Germany, August 7-12, 2016.

against widely used word embedding tools such as *word2vec*, henceforth *w2v* (Mikolov et al., 2013c), and *GloVe* (Pennington et al., 2014). We start with a discussion of related work (§2) and the materials and methods used (§3). We report on the evaluations performed (§4) and finish with conclusions and future work (§5).

2 Related Work

We define *nominal compounds* as conventional noun phrases composed by two or more words, such as *science fiction* (Nakov, 2013). In English, they are often expressed as *noun compounds* but their syntactic realization may vary for different languages. For instance, one of the equivalent forms in French involves a denominal adjective used as modifier (e.g. *cell death* and the corresponding *mort cellulaire*).[1] In this paper, we focus on 2-word nominal compounds involving modifiers that are nouns (e.g. *word embedding*) or adjectives (e.g. *hard time*).

Semantically, nominal compounds may display a wide range of idiomaticity, from compositional cases like *access road* to idiomatic or non-compositional cases like *gravy train*, whose meaning is unrelated to its parts.[2] Even when there is a level of compositionality in the compound, the contribution of each word may vary considerably, independently from its status as a syntactic head or modifier, as *cash* in *cash cow* versus *tears* in *crocodile tears*. Indeed, various annotation scales have been proposed as means to collect human judgments about compositionality. Particularly for nominal compounds, Reddy et al. (2011) used a 6-point scale to collect judgments on the literal or figurative use of nominal compounds and its components in English. Similar judgments have also been collected for 244 German compounds, for which an average of 30 judgments on a scale from 1 to 7 were gathered through crowdsourcing (Roller et al., 2013). An alternative to multi-point scales is the binary judgment adopted by Farahmand et al. (2015), for a dataset of English nominal compounds.

There has been much interest in creating semantic representations of larger units, such as phrases (Mikolov et al., 2013b), sentences and documents (Le and Mikolov, 2014), and in examining whether it is possible to accurately derive the semantics of a compound or multiword expression from its parts (McCarthy et al., 2003; Baldwin et al., 2003; Tratz and Hovy, 2010; Reddy et al., 2011). For the latter, proposals include using additive and multiplicative functions to combine vector representations of component words (Mitchell and Lapata, 2008; Reddy et al., 2011), calculating the overlap between the components and the expression (McCarthy et al., 2003) and looking at the literality of translations into multiple languages (Salehi et al., 2014). Other proposals to explicitly represent the semantics of nominal compounds include the use of paraphrases (Lauer, 1995; Nakov, 2008; Hendrickx et al., 2013), and inventories of semantic relations (Girju et al., 2005).

The ability of DSMs for accurately capturing semantic information may be affected by a number of factors involved in constructing the models, such as the source *corpus*, *context* representation, and parameters of the *model*. Relevant corpus parameters include size (Ferret, 2013; Mikolov et al., 2013c) and quality (Lapesa and Evert, 2014). Factors related to context representation include the context window size and the number of context dimensions adopted for a model (Lapesa and Evert, 2014); the choice of contexts to be used with targets (syntactic dependencies vs. bag-of-words) (Agirre et al., 2009); the use of morphosyntactic information (Padó and Lapata, 2003; Padó and Lapata, 2007); context filtering (Riedl and Biemann, 2012; Padró et al., 2014a); and dimensionality reduction methods (van de Cruys et al., 2012). Important model parameters that have been studied include the choice of association and similarity measures (Curran and Moens, 2002) and the use of subsampling and negative sampling techniques (Mikolov et al., 2013c). However, the particular effects may be heterogeneous and depend on the task and model (Lapesa and Evert, 2014). In this paper, we examine the impact of both corpus and context parameters for a variety of models, for the task of nominal compound compositionality prediction in English and French.

For the choice of particular DSM, contradictory results have been published showing the superiority of neural models (Baroni et al., 2014) and of more traditional but carefully designed models (Levy et al., 2015). The former were also reported as a better fit to behavioral data on semantic prim-

[1] In French, one can also use a preposition and optional determiner, like *cancer du poumon* (*lung cancer*).

[2] It refers to an initiative that provides money to many people without much effort.

ing tasks (Mandera et al., 2016). Moreover, these evaluations are often performed on single-word similarity tasks (Freitag et al., 2005; Camacho-Collados et al., 2015) and little has been said about the use of word embeddings for the compositionality prediction of multiword expressions. Two notable exceptions are the recent works of Salehi et al. (2015) and Yazdani et al. (2015). Salehi et al. (2015) show that word embeddings are more accurate in predicting compositionality than a simplistic count-based DSM. Yazdani et al. (2015) focus on the composition function, using a lightly supervised neural network to learn the best combination strategy for individual word vectors. In order to consolidate previous punctual results, we present a large-scale and systematic evaluation, comparing DSMs and their parameters, on several compositionality datasets.

3 Materials and Methods

We examine the impact of corpus parameters related to the target language and the degree of corpus preprocessing adopted. We also investigate context parameters related to the size of the context window and the number of dimensions used to represent context.

3.1 Corpora Preprocessing

We use the lemmatized and POS-tagged versions of the ukWaC for English (∼2 billion tokens) and frWaC (∼1.6 billion tokens) for French (Baroni et al., 2009) to train the models and build vector representations of words and compounds. For each corpus, we re-tokenize all target compounds as a single word with a separator (e.g. *monkey business* → *monkey_business*) and re-tag them using a single manually selected tag per compound to handle POS-tagging errors.[3] All forms are then lowercased (surface forms, lemmas and POS-tags); and noisy tokens, with special characters, numbers or punctuation, are removed. Additionally, ligatures are normalized for French (e.g. *œ* → *oe*) and a spellchecker[4] is applied to normalize words across English spelling variants (e.g. *color* → *colour*).

To test the influence of preprocessing in model accuracy, for each corpus, we generate four variants with different degrees of abstraction:

1. *surface*$^+$: the original corpus with no preprocessing, containing surface forms.
2. *surface*: stopword removal; generating a corpus of surface forms of content words.
3. *lemma*: stopword removal and lemmatization; generating a corpus of lemmas of content words.
4. *lemmaPOS*: stopword removal, lemmatization and POS-tagging; generating a corpus of content words, represented as lemma/tag.

The operation of stopword removal eliminates from the corpus all function words, leaving only nouns, adjectives, adverbs and verbs. In lemmatized corpora, the lemmas of proper names are replaced by placeholders.

3.2 Compositionality Datasets

For evaluation, we use nominal compound compositionality datasets for English (*Reddy*, *Reddy++* and *Farahmand*) and for French (*FR-comp*). They provide annotations as to whether a given compound is more idiomatic or more compositional.

Reddy contains compositionality judgments for 90 compounds and their individual word components, in a scale of literality from 0 (idiomatic) to 5 (literal), collected with Mechanical Turk (Reddy et al., 2011). For each compound, compositionality scores are averaged over its annotators. Compounds included in the dataset were selected to balance frequency range and degree of compositionality (low, middle and high). We use only the global compositionality score, ignoring individual word judgments. With a few exceptions (e.g. *sacred cow*), most compounds are formed exclusively by nouns.

Reddy++ is a new resource created for this evaluation (Ramisch et al., 2016). It extends the *Reddy* set with an additional 90 English nominal compounds, in a total of 180 entries. Scores also range from 0 to 5 and were collected through Mechanical Turk and averaged over the annotators. The extra 90 entries include some adjective-noun compounds and are balanced with respect to frequency and compositionality. We focus our evaluation on this combined dataset, since it includes *Reddy*. However, to allow comparison with state of the art, we also report results individually for *Reddy*.

Farahmand contains 1042 English compounds extracted from Wikipedia with binary non-compositionality judgments by four experts (Farahmand et al., 2015). We consider a compound as non-compositional if at least two judges agree that it is non-compositional, following Yaz-

[3] We use a simplified tag set (e.g. `v` instead of `vvz`).

[4] `https://hunspell.github.io`

dani et al. (2015). In our evaluations, we use the sum of all judgments in order to have a single numeral compositionality score, ranging from 0 (compositional) to 4 (idiomatic).

FR-comp is also a new resource created for this evaluation (Ramisch et al., 2016). It contains 180 adjective-noun and noun-adjective compounds in French, such as *belle-mère* (*mother-in-law*, lit. *beautiful-mother*) and *carte bleue* (*credit card*, lit. *blue card*). This dataset was constructed in the same manner as the extension to *Reddy*, that is, using crowdsourcing and average numerical scores. Special care was taken to guarantee that annotators were native speakers by asking them to provide paraphrases along with compositionality scores.

The new datasets *Reddy++* and *FR-comp* are similar to *Reddy*. For instance, the average standard deviation of compound scores given by different annotators is $\sigma = 1.17$ for the new compounds in *Reddy++*, $\sigma = 1.15$ for *FR-comp* and $\sigma = 0.99$ for *Reddy*. Their detailed evaluation is presented by Ramisch et al. (2016).

3.3 DSM Models

We build three types of DSMs: models based on sparse PPMI cooccurrence vectors, as well as those constructed with *word2vec* and *GloVe*.

PPMI For each target word or compound, we extract from the corpus its neighboring nouns and verbs in a symmetric sliding window of w words to the left/right[5], using a linear decay weighting scheme with respect to its distance d to the target (Levy et al., 2015). In other words, each cooccurrence count of target-context pairs is incremented by $w + 1 - d$ instead of 1. The representation of a target is a vector containing the positive pointwise mutual information (PPMI) association scores between the target and its contexts.[6]

In *PPMI-thresh*, we follow Padró et al. (2014b) to select the top k most relevant contexts (highest PPMI) for each target. No further dimensionality reduction is applied.

In *PPMI-TopK*, we use a fixed global list of 1000 contexts, built by looking at the most frequent words in the corpus: the top 50 are skipped, and the next 1000 are taken (Salehi et al., 2015). No further dimensionality reduction is applied.

In *PPMI-SVD*, for each target, contexts that appear less than 1000 times are discarded.[7] We then use the Dissect toolkit[8] (Dinu et al., 2013) in order to build a PPMI matrix and reduce its dimensionality using singular value decomposition (SVD) to factorize the matrix.

w2v Uses the *word2vec* toolkit based on neural networks to predict target/context cooccurrence (Mikolov et al., 2013a). We build models from two variants of word2vec: CBOW (*w2v-cbow*) and skipgram (*w2v-sg*). In both cases, the configurations are the default ones, except for the following: no hierarchical softmax; negative sampling of 25; frequent-word downsampling weight of 10^{-6}; runs 15 training iterations. We use the default minimum word count threshold of 5.

glove We use the count-based DSM of Pennington et al. (2014), which implements a factorization of the co-occurrence count matrix. The configurations are the default ones, except for the following: internal cutoff parameter $x_{max} = 75$; builds co-occurrence matrix in 15 iterations. Due to the large vocabulary size, we use a minimum word count threshold of 5 for lemma-based models, 15 for *surface* and 20 for *surface*$^+$.

For each DSM, we evaluate the influence of a set of parameters. By varying the values of these parameters, we build a total of 408 models per language. The parameters are:

- WORDFORM: Refers to one of the four variants of each corpus: *surface*$^+$, *surface*, *lemma*, and *lemmaPOS*.
- WINDOWSIZE: Indicates within how many words to the left/right we are searching for target-context co-occurrence pairs. In this work we explore windows of sizes of 1, 4 and 8.
- DIMENSION: Each model is constructed to have a maximum number of final dimensions for each vector. We generate models with 250, 500 and 750 dimensions.

3.4 Compositionality Prediction

To predict the compositionality of a nominal compound w_1w_2 using the DSMs, we use as a measure the cosine similarity between the compound

[5] Syntactic context definition is planned as future work.

[6] *PPMI* vectors are built using minimantics `https://github.com/ceramisch/minimantics`.

[7] Aggressive filtering was required because SVD seems quite sensitive to low-frequency contexts.

[8] `http://clic.cimec.unitn.it/composes/toolkit/index.html`

vector representation $v(w_1w_2)$ and the sum of the vector representations of the component words:

$$\cos(\ v(w_1w_2),\ v(w_1 + w_2)\ )$$

where for $v(w_1 + w_2)$ we use the normalized sum

$$v(w_1 + w_2) = \frac{v(w_1)}{||v(w_1)||} + \frac{v(w_2)}{||v(w_2)||}.$$

In this framework, a compound is compositional if the compound representation is close to the sum of its components representations (cosine is close to 1), and it is idiomatic otherwise.

One possible improvement of the predictive model would consist in using more sophisticated composition functions instead of sum, such as the multiplicative model of Mitchell and Lapata (2008). However, we want to first assess the performance of a simple additive function. Other optimized functions like the ones proposed by Yazdani et al. (2015) could also be verified, but are out of the scope of this paper, since they are based on supervised learning.

3.5 Evaluation Setup

We evaluate the compositionality models and their parameters on the datasets described in Section 3.2. For *Reddy*, *Reddy++* and *FR-comp*, we report Spearman's ρ correlation between the ranking provided by humans and those calculated from the models. We follow Yazdani et al. (2015) and report the best F1 score (BF1) obtained for the *Farahmand* dataset, by calculating the F1 score for the top k compounds classified as positive (non-compositional), for all possible values of k.

Given the high number of experiments we performed, we report the best performance of each model type. For instance, the performances reported for *w2v-cbow* using different values of WINDOWSIZE are the best configurations across all possible values of other parameters such as DIMENSION and WORDFORM. This avoids reporting local maxima that can arise if one fixes all other parameters when evaluating a given one (Lapesa and Evert, 2014).

For *Reddy++* and *Farahmand*, we distinguish between *strict* evaluation, reported in the form of wider bars in the figures, and *loose* evaluation, shown as narrow blue bars in the figures. Strict evaluation corresponds to the performance of the model only on those compounds that have a vector representation in all underlying DSMs, 175 out of 180 for *Reddy++* and 913 out of 1042 for *Farahmand*. Loose evaluation considers the full dataset, using a fallback strategy for the imputation of missing values, assigning the average compositionality score to absent compounds (Salehi et al., 2015). This is particularly important for *Farahmand*, which contains more rare compounds such as *universe human* and *mankind instruction* so that 129 compounds are missing in the corpus. Only strict evaluation is reported for *FR-comp*, as all compounds are frequent enough in FRWaC.

The vectors generated by *w2v* and *glove* have some non-determinism due to random initialization. To assess its impact on results, we report the average of 3 runs using identical configurations and use error bars in the graphics.[9]

4 Results

We report results on each dataset separately and then discuss findings that hold for all datasets.

4.1 *Reddy++* and *Reddy* Datasets

Figure 1 summarizes the results for *Reddy++* dataset.[10] Overall, *w2v-cbow* ($\rho = 0.73$), *w2v-sg* ($\rho = 0.73$), *PPMI-SVD* ($\rho = 0.72$) and *PPMI-thresh* ($\rho = 0.71$) obtain similar results. In spite of this, except for the two best *w2v* models, all differences were deemed statistically significant (Wilcoxon rank correlation test, $p < 0.05$).

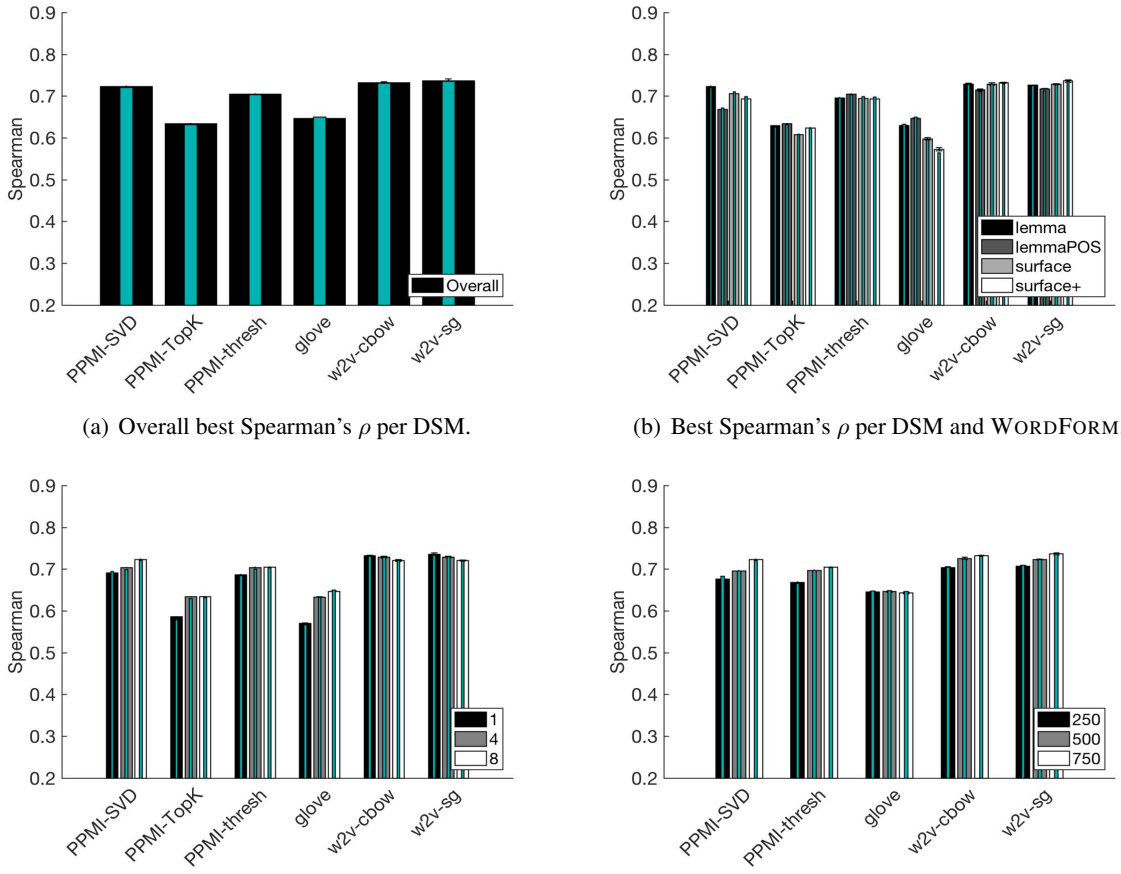
Figure 1(b) shows the influence of the degree of corpus preprocessing (shown as WORDFORM in these figures). The results are heterogeneous, as the best *w2v* models seem to profit from the presence of stopwords, unlike the other models for which more preprocessing (*lemma* and *lemma-POS*) leads to better results. One exception is *PPMI-SVD* for which the use of *lemmaPOS* drastically reduces performance.[11]

For WINDOWSIZE, Figure 1(c), although increasing context size seems to help DSMs (at least up to 4), for the best *w2v* models, a better result is obtained with limited context of 1 word left/right. Probably the interaction between the subsampling strategy and randomized window size explains why increasing this value does not improve the

[9] Error bars are barely visible because results are stable.

[10] In the remainder of this section, we will discuss strict evaluation results (outer bars).

[11] Further investigation must be done to determine the cause of this reduction as an increase in vocabulary size alone is insufficient to explain the effect, given that both surface forms outperform it.



(a) Overall best Spearman's ρ per DSM.

(b) Best Spearman's ρ per DSM and WORDFORM.

(c) Best Spearman's ρ per DSM and WINDOWSIZE.

(d) Best Spearman's ρ per DSM and DIMENSION.

Figure 1: Spearman's ρ for different DSM parameters on *Reddy++* dataset.

results. *PPMI-SVD* can use extra information from larger window sizes (WINDOWSIZE=8) better than models based on context filtering. This is probably related to the aggressive context filter, which keeps only very salient cooccurrences even in large windows.

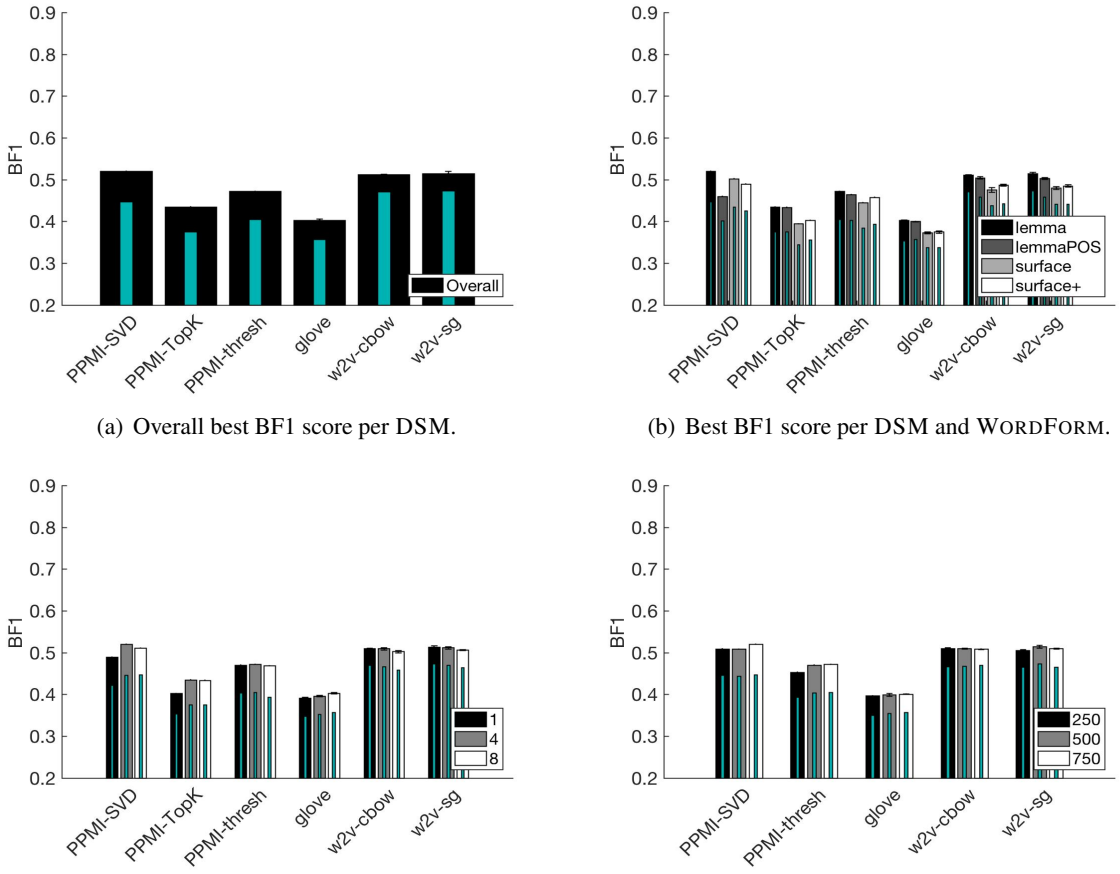
The results for context vector dimensionality, Figure 1(d), show, as expected, that the best results are obtained with larger dimensions (DIMENSION=750) for all models, except for *glove*, which displays very similar results independently of the number of dimensions.

Examining the *Reddy* dataset alone, the same trends for all parameters were found, but with higher results. The overall best performances on *Reddy* were quite similar: *w2v-cbow* ($\rho = 0.82$), *w2v-sg* ($\rho = 0.81$), *PPMI-SVD* ($\rho = 0.80$) and *PPMI-thresh* ($\rho = 0.79$), and the differences are significant except for the two best *w2v* models. The 90 compounds added to *Reddy++* seem to be more difficult to assess than the original ones, probably because they include many adjectives, which have been found harder to judge for compositionality than nouns (Ramisch et al., 2016).

4.2 *Farahmand* Dataset

Figure 2(a) shows the overall best model for the *Farahmand* dataset. *PPMI-SVD* reached a BF1 score of 0.52, with DIMENSION=750, WINDOWSIZE=4, using *lemma*, and both *w2v* (BF1=0.51) obtain comparable results with similar configurations.

These results show a marked difference between the loose (the narrower bars in the figures) and the strict evaluation (wider bars). The former uses a fallback strategy for the imputation of missing values that does not accurately reflect how the compositionality scores vary. Indeed, we observed that compounds that do not appear very often in our corpora tend to be non-compositional, whereas most of the compound occurrences are compositional, increasing average compositionality. For instance, the 10 most compositional compounds in *Reddy++* occur an average of 26551



(a) Overall best BF1 score per DSM.

(b) Best BF1 score per DSM and WORDFORM.

(c) Best BF1 score per DSM and WINDOWSIZE.

(d) Best BF1 score per DSM and DIMENSION.

Figure 2: BF1 scores for different DSM parameters on *Farahmand* dataset.

times in the UKWaC vs 1096 times for the 10 least compositional ones. Spearman rank correlation between frequency and compositionality in *Reddy++* is $\rho = 0.43$.[12] In short, even if a fallback strategy is adopted as the means to obtain a lower-bound for performance, it may be unrelated to the real performance for the missing compounds.

For most models, corpus preprocessing resulted in better scores, with WORDFORM=*lemma* outperforming all other forms of preprocessing, especially for French. Concatenating lemmas and POS tags does not seem to help, probably due to decreasing word frequencies without substantial gain in informativeness (Figure 2(b)).

The impact of WINDOWSIZE has a similar trend to the one found for the *Reddy++* and *Reddy* datasets (Figure 2(c)). That is, the larger window was preferred by most models, but the average difference between the best and the worst size for each DSM is only 0.01. For DIMENSION, a larger number resulted in better scores, as expected, with 750 being the best for all models in Figure 2(d). Nonetheless, here too the average difference in scores between DIMENSION=750 and 250 is 0.01.

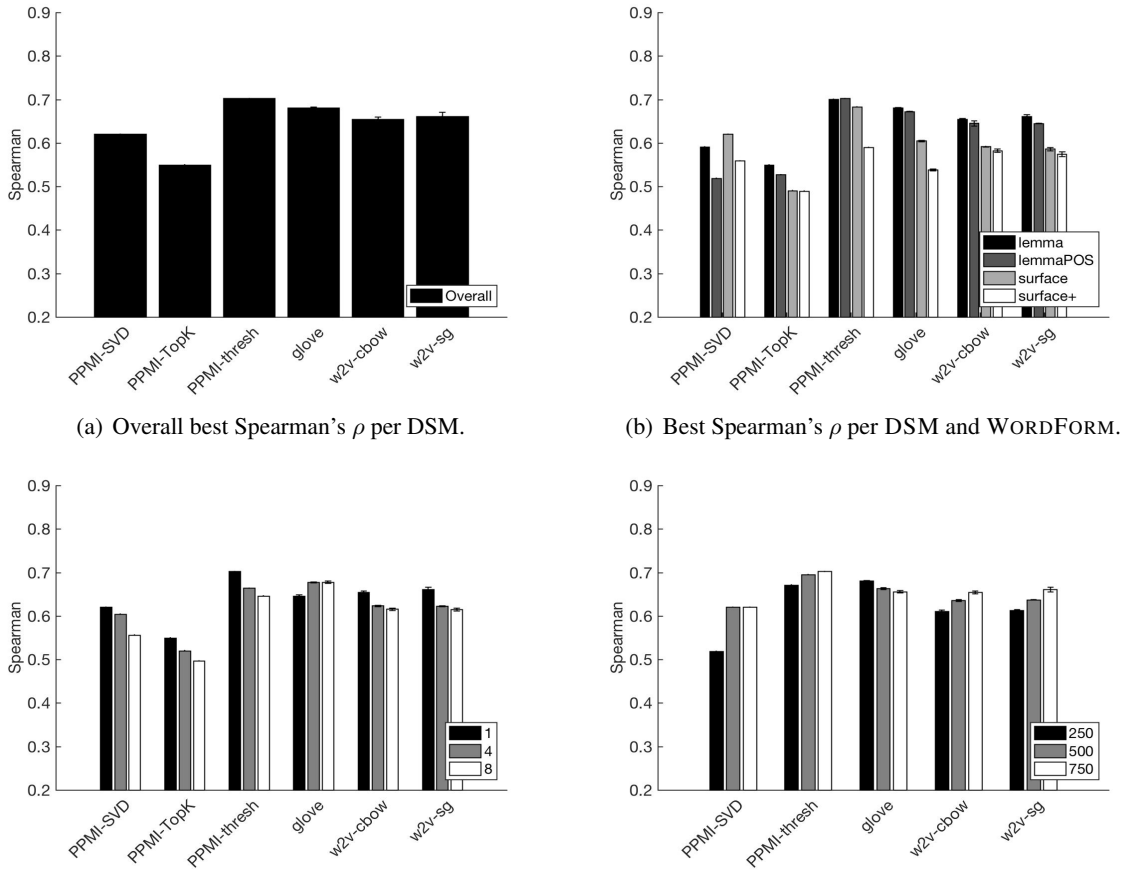
4.3 *FR-comp* Dataset

Globally, for the *FR-comp* dataset, *PPMI-thresh* ($\rho = 0.70$) outperforms *glove* ($\rho = 0.68$) and *w2v* ($\rho = 0.66$), as can be seen in Figure 3(a).[13]

For morphologically rich languages like French, Figure 3(b) indicates that working on lemmatized data often yields better results than working on surface forms. Lemmas conflate the frequencies for all the many morphologically inflected variants which would otherwise be dispersed in different surface forms. Therefore, it is not surprising that the best results concerning WORDFORM are achieved by *lemma*. These results differ from English, where a corpus without any preprocess-

[12] We report these figures for *Reddy++* because *Farahmand* has many ties, given the binary nature of compositionality annotations.

[13] As all compounds in the dataset occur in the corpus, only strict evaluation results are reported.



(a) Overall best Spearman's ρ per DSM.

(b) Best Spearman's ρ per DSM and WORDFORM.

(c) Best Spearman's ρ per DSM and WINDOWSIZE.

(d) Best Spearman's ρ per DSM and DIMENSION.

Figure 3: Spearman's ρ for different DSM parameters on *FR-comp* dataset.

ing yields more accurate results. Moreover, a smaller WINDOWSIZE leads to better results for most models, as shown in Figure 3(c). But just as in English, all models except *glove* benefit from an increase in dimension, as shown in Figure 3(d).[14]

4.4 Discussion

When comparing DIMENSION across languages and datasets, larger values often bring better performance. Likewise, the *lemma* is usually the better WORDFORM. The recommended WINDOWSIZE depends on the model and language, but for the best models in all datasets, a window of 1 outperforms the others. This may be a consequence of the linear decay context weighting process, which assigns higher weights to closer words as window size increases. As an overall conclusion, in combination with a large dimension and a small window size, investing in preprocessing provides a good balance of a small vocabulary (of lemmas) and good accuracy. This is especially clear for a morphologically richer language like French, where lemmatization is homogeneously better for all models, even in *w2v*, for which surface forms were better for English.

In terms of models, the *w2v* models performed better than *PPMI* for *Reddy++*, both were in a tie for *Farahmand*, and *w2v* was outperformed by *PPMI-thresh* for French. The performance of *glove* for English was underwhelming, probably because we did not perform parameter tuning. As shown by (Salehi et al., 2015), *PPMI-TopK* is not an appropriate DSM for this task, as it does not model relevant cooccurrence very well.

The average Spearman's ρ for *Reddy* over all tested parameter configurations was 0.71 for both *w2v* models and 0.67 for *PPMI-SVD* and *PPMI-*

[14]For *w2v*, the same parameters used for English were adopted also for French. As a sanity check, we tested a range of negative sampling values [5, 15, 25, 35, 50], as well as subsampling rates for powers of 10 in [10^{-3} to 10^{-7}]. Variations in ρ are minor and do not show any clear trend.

[14]DSM parameters: WF: WORDFORM, D: DIMENSION, W: WINDOWSIZE. Results in parentheses for loose evaluation, using fallback.

Model & Parameters	Result
Reddy et al. (2011)	.71
Salehi et al. (2014)	.74
Salehi et al. (2015)	.80
Best *w2v* (*sg, WF=surface, D=750, W=1*)	.82 (.80)
Best *PPMI* (*thresh, WF=surface, D=750, W=8*)	.80 (.80)
Best *glove* (*WF=lemmapos, D=250, W=8*)	.76 (.76)

Table 1: Comparison of our best models with state-of-the-art ρ for *Reddy*.[14]

Model & Parameters	Result
Yazdani et al. (2015)	.49
Best *w2v* (*sg, WF=lemma, D=500, W=1*)	.51 (.47)
Best *PPMI* (*svd, WF=lemma, D=750, W=4*)	.52 (.45)
Best *glove* (*WF=lemma, D=500, W=8*)	.40 (.36)

Table 2: Comparison of our best models with state-of-the-art BF1 for *Farahmand*.[14]

thresh, and this was also observed for the other datasets. In short, both types of models can obtain good results. While *PPMI-thresh* is a simple, fast and inexpensive model to build, *w2v* has a free and push-button implementation, and requires less hyper-parameter tuning, as is it seems more robust to parameter variation. More generally, the best results obtained for *Reddy* and *Farahmand* are comparable and even outperform the state of the art, as shown in Tables 1 and 2, when strict evaluation is adopted (that is, when not using a fallback strategy for missing compounds).

5 Conclusions

In this paper we presented a multilingual, large-scale evaluation of DSMs for compound compositionality prediction. We have built 816 DSMs and performed 2,856 evaluations, examining the impact of corpus and context parameters, namely the level of corpus preprocessing, the context window size and the number of dimensions. Evaluation on 3 English datasets and a French one revealed that a large dimension is consistently better, and corpus preprocessing is usually beneficial. The choice of window size varies according to language and dataset, but a small window can often provide a good performance. The DSMs *w2v* and *PPMI* alternated in providing the best results. Moreover, the results obtained were comparable and even outperformed the state-of-the-art.

As future work, we plan to examine the use of a voting scheme for combining the output of complementary DSMs. Moreover, we also plan to combine additional sources of information for building the models, such as multilingual resources or translation data, to improve even further the compositionality prediction. We would also like to propose and evaluate more sophisticated compositionality functions that take into account the unbalanced contribution of individual words to the global meaning of a compound.

Acknowledgments

This work has been partly funded by projects PARSEME (Cost Action IC1207), PARSEME-FR (ANR-14-CERA-0001), AIM-WEST (FAPERGS-INRIA 1706-2551/13-7), CNPq 482520/2012-4, 312114/2015-0, "Simplificação Textual de Expressões Complexas", sponsored by Samsung Eletrônica da Amazônia Ltda. under the terms of Brazilian federal law No. 8.248/91.

References

Eneko Agirre, Enrique Alfonseca, Keith B. Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, May 31 - June 5, 2009, Boulder, Colorado, USA*, pages 19–27. The Association for Computational Linguistics.

Timothy Baldwin, Colin Bannard, Takaaki Tanaka, and Dominic Widdows. 2003. An empirical model of multiword expression decomposability. In Francis Bond, Anna Korhonen, Diana McCarthy, and Aline Villavicencio, editors, *Proc. of the ACL Workshop on MWEs: Analysis, Acquisition and Treatment (MWE 2003)*, pages 89–96, Sapporo, Japan, Jul. ACL.

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

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The wacky wide web: a collection of very large linguistically processed web-crawled corpora. *Lang. Res. & Eval.*, 43(3):209–226, Sep.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long*

Papers), pages 238–247, Baltimore, Maryland, June. Association for Computational Linguistics.

José Camacho-Collados, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. A framework for the construction of monolingual and cross-lingual word similarity datasets. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1–7, Beijing, China, July. Association for Computational Linguistics.

James Curran and Marc Moens. 2002. Scaling context space. In *Proc. of the 40th ACL (ACL 2002)*, pages 231–238, Philadelphia, PA, USA, Jul. ACL.

Georgiana Dinu, Nghia The Pham, and Marco Baroni. 2013. DISSECT - DIStributional SEmantics composition toolkit. In *Proc. of the ACL 2013 System Demonstrations*, pages 31–36, Sofia, Bulgaria, Aug. ACL.

Katrin Erk and Sebastian Padó. 2010. Exemplar-based models for word meaning in context. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden, Short Papers*, pages 92–97. The Association for Computer Linguistics.

Meghdad Farahmand, Aaron Smith, and Joakim Nivre. 2015. A multiword expression data set: Annotating non-compositionality and conventionalization for english noun compounds. In *Proceedings of the 11th Workshop on Multiword Expressions*, pages 29–33, Denver, Colorado, June. Association for Computational Linguistics.

Olivier Ferret. 2013. Identifying bad semantic neighbors for improving distributional thesauri. In *Proc. of the 51st ACL (Volume 1: Long Papers)*, pages 561–571, Sofia, Bulgaria, Aug. ACL.

Olivier Ferret. 2014. Compounds and distributional thesauri. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014), Reykjavik, Iceland, May 26-31, 2014.*, pages 2979–2984. European Language Resources Association (ELRA).

Dayne Freitag, Matthias Blume, John Byrnes, Edmond Chow, Sadik Kapadia, Richard Rohwer, and Zhiqiang Wang. 2005. New experiments in distributional representations of synonymy. In Ido Dagan and Dan Gildea, editors, *Proc. of the Ninth CoNLL (CoNLL-2005)*, pages 25–32, University of Michigan, MI, USA, Jun. ACL.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer speech & language*, 19(4):479–496.

Iris Hendrickx, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Stan Szpakowicz, and Tony Veale. 2013. Semeval-2013 task 4: Free paraphrases of noun compounds. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 138–143, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Su Nam Kim and Timothy Baldwin. 2006. Interpreting semantic relations in noun compounds via verb semantics. In James Curran, editor, *Proc. of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 491–498, Sidney, Australia, Jul. ACL.

Gabriella Lapesa and Stefan Evert. 2014. A large scale evaluation of distributional semantic models: Parameters, interactions and model selection. *Transactions of the Association for Computational Linguistics*, 2:531–545.

Mark Lauer. 1995. How much is enough?: Data requirements for statistical NLP. *CoRR*, abs/cmp-lg/9509001.

Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Proceedings*, pages 1188–1196. JMLR.org.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Dekang Lin. 1999. Automatic identification of non-compositional phrases. In *Proc. of the 37th ACL (ACL 1999)*, pages 317–324, College Park, MD, USA, Jun. ACL.

Paweł Mandera, Emmanuel Keuleers, and Marc Brysbaert. 2016. Explaining human performance in psycholinguistic tasks with models of semantic similarity based on prediction and counting: A review and empirical validation. *Journal of Memory and Language*.

Diana McCarthy, Bill Keller, and John Carroll. 2003. Detecting a continuum of compositionality in phrasal verbs. In Francis Bond, Anna Korhonen, Diana McCarthy, and Aline Villavicencio, editors, *Proc. of the ACL Workshop on MWEs: Analysis, Acquisition and Treatment (MWE 2003)*, pages 73–80, Sapporo, Japan, Jul. ACL.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, pages 3111–3119.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751. The Association for Computational Linguistics.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proc. of the 46th ACL: HLT (ACL-08: HLT)*, pages 236–244, Columbus, OH, USA, Jun. ACL.

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

Preslav Nakov. 2008. Paraphrasing verbs for noun compound interpretation. In *Proc. of the LREC Workshop Towards a Shared Task for MWEs (MWE 2008)*, pages 46–49.

Preslav Nakov. 2013. On the interpretation of noun compounds: Syntax, semantics, and entailment. *Nat. Lang. Eng. Special Issue on Noun Compounds*, 19(3):291–330.

Sebastian Padó and Mirella Lapata. 2003. Constructing semantic space models from parsed corpora. In *Proc. of the 41st ACL (ACL 2003)*, pages 128–135, Sapporo, Japan, Jul. ACL.

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

Muntsa Padró, Marco Idiart, Aline Villavicencio, and Carlos Ramisch. 2014a. Comparing similarity measures for distributional thesauri. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, Reykjavik, Iceland, May. European Language Resources Association.

Muntsa Padró, Marco Idiart, Aline Villavicencio, and Carlos Ramisch. 2014b. Nothing like good old frequency: Studying context filters for distributional thesauri. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2014) - short papers*, Doha, Qatar, Oct.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October. Association for Computational Linguistics.

Carlos Ramisch, Silvio Cordeiro, Leonardo Zilio, Marco Idiart, Aline Villavicencio, and Rodrigo Wilkens. 2016. How naked is the naked truth? a multilingual lexicon of nominal compound compositionality. In *Proc. of the 55th ACL (Volume 2: Short Papers)*, Berlin, Germany, Aug. ACL.

Siva Reddy, Diana McCarthy, and Suresh Manandhar. 2011. An empirical study on compositionality in compound nouns. In *Proceedings of The 5th International Joint Conference on Natural Language Processing 2011 (IJCNLP 2011)*, Chiang Mai, Thailand, November.

Martin Riedl and Chris Biemann. 2012. Topictiling: A text segmentation algorithm based on LDA. In *Proc. of the ACL 2012 SRW*, pages 37–42, Jeju, Republic of Korea, Jul. ACL.

Stephen Roller, Sabine Schulte im Walde, and Silke Scheible. 2013. The (un)expected effects of applying standard cleansing models to human ratings on compositionality. In Valia Kordoni, Carlos Ramisch, and Aline Villavicencio, editors, *Proc. of the 9th Workshop on MWEs (MWE 2013)*, pages 32–41, Atlanta, GA, USA, Jun. ACL.

Ivan Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. of the 3rd CICLing (CICLing-2002)*, volume 2276/2010 of *LNCS*, pages 1–15, Mexico City, Mexico, Feb. Springer.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2014. Using distributional similarity of multi-way translations to predict multiword expression compositionality. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 472–481, Gothenburg, Sweden, April. Association for Computational Linguistics.

Bahar Salehi, Paul Cook, and Timothy Baldwin. 2015. A word embedding approach to predicting the compositionality of multiword expressions. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 977–983, Denver, Colorado, May–June. Association for Computational Linguistics.

Stephen Tratz and Eduard Hovy. 2010. ISI: Automatic classification of relations between nominals using a maximum entropy classifier. In Katrin Erk and Carlo Strapparava, editors, *Proc. of the 5th SemEval (SemEval 2010)*, pages 222–225, Uppsala, Sweden, Jul. ACL.

Tim van de Cruys, Laura Rimell, Thierry Poibeau, and Anna Korhonen. 2012. Multi-way tensor factorization for unsupervised lexical acquisition. In *Proc. of the 24th COLING (COLING 2012)*, pages 2703–2720, Mumbai, India, Dec. The Coling 2012 Organizing Committee.

Majid Yazdani, Meghdad Farahmand, and James Henderson. 2015. Learning semantic composition to detect non-compositionality of multiword expressions. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1733–1742, Lisbon, Portugal, September. Association for Computational Linguistics.

Learning-Based Single-Document Summarization with Compression and Anaphoricity Constraints

Greg Durrett
Computer Science Division
UC Berkeley
gdurrett@cs.berkeley.edu

Taylor Berg-Kirkpatrick
School of Computer Science
Carnegie Mellon University
tberg@cs.cmu.edu

Dan Klein
Computer Science Division
UC Berkeley
klein@cs.berkeley.edu

Abstract

We present a discriminative model for single-document summarization that integrally combines compression and anaphoricity constraints. Our model selects textual units to include in the summary based on a rich set of sparse features whose weights are learned on a large corpus. We allow for the deletion of content within a sentence when that deletion is licensed by compression rules; in our framework, these are implemented as dependencies between subsentential units of text. Anaphoricity constraints then improve cross-sentence coherence by guaranteeing that, for each pronoun included in the summary, the pronoun's antecedent is included as well or the pronoun is rewritten as a full mention. When trained end-to-end, our final system[1] outperforms prior work on both ROUGE as well as on human judgments of linguistic quality.

1 Introduction

While multi-document summarization is well-studied in the NLP literature (Carbonell and Goldstein, 1998; Gillick and Favre, 2009; Lin and Bilmes, 2011; Nenkova and McKeown, 2011), single-document summarization (McKeown et al., 1995; Marcu, 1998; Mani, 2001; Hirao et al., 2013) has received less attention in recent years and is generally viewed as more difficult. Content selection is tricky without redundancy across multiple input documents as a guide and simple positional information is often hard to beat (Penn and Zhu, 2008). In this work, we tackle the single-document problem by training an expressive summarization model on a large naturally occurring corpus—the New York Times Annotated Corpus (Sandhaus, 2008) which contains around 100,000 news articles with abstractive summaries—learning to select important content with lexical features. This corpus has been explored in related contexts (Dunietz and Gillick, 2014; Hong and Nenkova, 2014), but to our knowledge it has not been directly used for single-document summarization.

To increase the expressive capacity of our model we allow more aggressive compression of individual sentences by combining two different formalisms—one syntactic and the other discursive. Additionally, we incorporate a model of anaphora resolution and give our system the ability rewrite pronominal mentions, further increasing expressivity. In order to guide the model, we incorporate (1) constraints from coreference ensuring that critical pronoun references are clear in the final summary and (2) constraints from syntactic and discourse parsers ensuring that sentence realizations are well-formed. Despite the complexity of these additional constraints, we demonstrate an efficient inference procedure using an ILP-based approach. By training our full system end-to-end on a large-scale dataset, we are able to learn a high-capacity structured model of the summarization process, contrasting with past approaches to the single-document task which have typically been heuristic in nature (Daumé and Marcu, 2002; Hirao et al., 2013).

We focus our evaluation on the New York Times Annotated corpus (Sandhaus, 2008). According to ROUGE, our system outperforms a document prefix baseline, a bigram coverage baseline adapted from a strong multi-document system (Gillick and Favre, 2009), and a discourse-informed method from prior work (Yoshida et al., 2014). Imposing discursive and referential constraints improves human judgments of linguistic clarity and referential structure—outperforming the method of

[1] Available at http://nlp.cs.berkeley.edu

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 1998–2008,
Berlin, Germany, August 7-12, 2016.

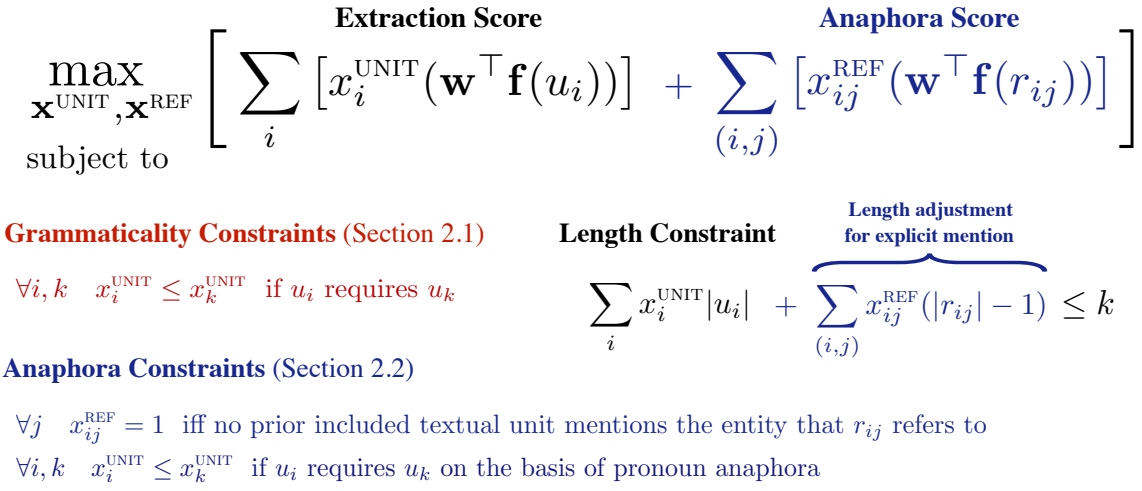


Figure 1: ILP formulation of our single-document summarization model. The basic model extracts a set of textual units with binary variables $\mathbf{x}^{\text{UNIT}}$ subject to a length constraint. These textual units $\mathbf{u}$ are scored with weights $\mathbf{w}$ and features $\mathbf{f}$. Next, we add constraints derived from both syntactic parses and Rhetorical Structure Theory (RST) to enforce grammaticality. Finally, we add anaphora constraints derived from coreference in order to improve summary coherence. We introduce additional binary variables $\mathbf{x}^{\text{REF}}$ that control whether each pronoun is replaced with its antecedent using a candidate replacement r_{ij}. These are also scored in the objective and are incorporated into the length constraint.

Yoshida et al. (2014) and approaching the clarity of a sentence-extractive baseline—and still achieves substantially higher ROUGE score than either method. These results indicate that our model has the expressive capacity to extract important content, but is sufficiently constrained to ensure fluency is not sacrificed as a result.

Past work has explored various kinds of structure for summarization. Some work has focused on improving content selection using discourse structure (Louis et al., 2010; Hirao et al., 2013), topical structure (Barzilay and Lee, 2004), or related techniques (Mithun and Kosseim, 2011). Other work has used structure primarily to reorder summaries and ensure coherence (Barzilay et al., 2001; Barzilay and Lapata, 2008; Louis and Nenkova, 2012; Christensen et al., 2013) or to represent content for sentence fusion or abstraction (Thadani and McKeown, 2013; Pighin et al., 2014). Similar to these approaches, we appeal to structures from upstream NLP tasks (syntactic parsing, RST parsing, and coreference) to restrict our model's capacity to generate. However, we go further by optimizing for ROUGE subject to these constraints with end-to-end learning.

2 Model

Our model is shown in Figure 1. Broadly, our ILP takes a set of textual units $\mathbf{u} = (u_1, \ldots, u_n)$ from a document and finds the highest-scoring extractive summary by optimizing over variables $\mathbf{x}^{\text{UNIT}} = x_1^{\text{UNIT}}, \ldots, x_n^{\text{UNIT}}$, which are binary indicators of whether each unit is included. Textual units are contiguous parts of sentences that serve as the fundamental units of extraction in our model. For a sentence-extractive model, these would be entire sentences, but for our compressive models we will have more fine-grained units, as shown in Figure 2 and described in Section 2.1. Textual units are scored according to features $\mathbf{f}$ and model parameters $\mathbf{w}$ learned on training data. Finally, the extraction process is subject to a length constraint of k words. This approach is similar in spirit to ILP formulations of multi-document summarization systems, though in those systems content is typically modeled in terms of bigrams (Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011; Hong and Nenkova, 2014; Li et al., 2015). For our model, type-level n-gram scoring only arises when we compute our loss function in max-margin training (see Section 3).

In Section 2.1, we discuss grammaticality constraints, which take the form of introducing dependencies between textual units, as shown in Figure 2. If one textual unit *requires* another, it cannot be included unless its prerequisite is. We will show that different sets of requirements can capture both syntactic and discourse-based compression schemes.

Furthermore, we introduce anaphora constraints (Section 2.2) via a new set of variables that capture the process of rewriting pronouns to make them

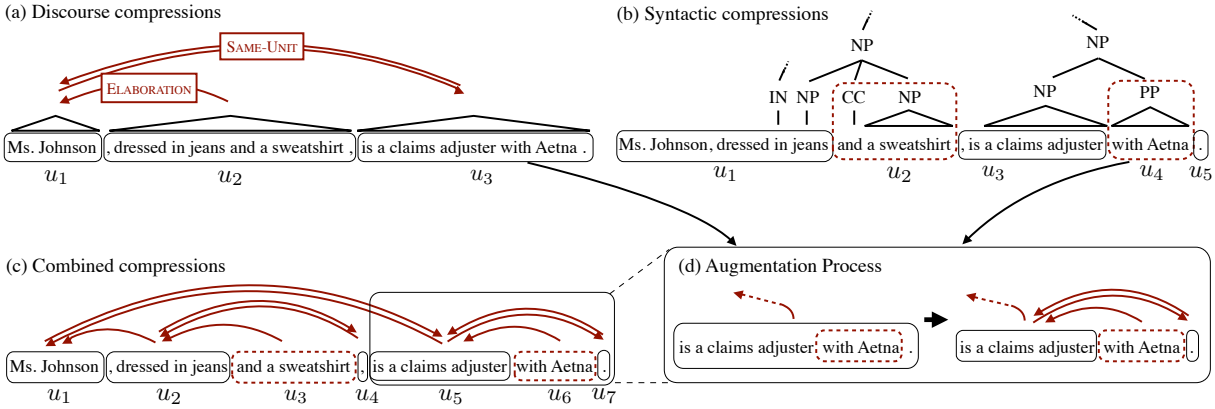


Figure 2: Compression constraints on an example sentence. (a) RST-based compression structure like that in Hirao et al. (2013), where we can delete the ELABORATION clause. (b) Two syntactic compression options from Berg-Kirkpatrick et al. (2011), namely deletion of a coordinate and deletion of a PP modifier. (c) Textual units and requirement relations (arrows) after merging all of the available compressions. (d) Process of augmenting a textual unit with syntactic compressions.

explicit mentions. That is, $x_{ij}^{\text{REF}} = 1$ if we should rewrite the jth pronoun in the ith unit with its antecedent. These pronoun rewrites are scored in the objective and introduced into the length constraint to make sure they do not cause our summary to be too long. Finally, constraints on these variables control when they are used and also require the model to include antecedents of pronouns when the model is not confident enough to rewrite them.

2.1 Grammaticality Constraints

Following work on isolated sentence compression (McDonald, 2006; Clarke and Lapata, 2008) and compressive summarization (Lin, 2003; Martins and Smith, 2009; Berg-Kirkpatrick et al., 2011; Woodsend and Lapata, 2012; Almeida and Martins, 2013), we wish to be able to compress sentences so we can pack more information into a summary. During training, our model learns how to take advantage of available compression options and select content to match human generated summaries as closely possible.[2] We explore two ways of deriving units for compression: the RST-based compressions of Hirao et al. (2013) and the syntactic compressions of Berg-Kirkpatrick et al. (2011).

RST compressions Figure 2a shows how to derive compressions from Rhetorical Structure Theory (Mann and Thompson, 1988; Carlson et al., 2001). We show a sentence broken into elementary discourse units (EDUs) with RST relations between them. Units marked as SAME-UNIT must both be kept or both be deleted, but other nodes in the tree structure can be deleted as long as we do not delete the parent of an included node. For example, we can delete the ELABORATION clause, but we can delete neither the first nor last EDU. Arrows depict the constraints this gives rise to in the ILP (see Figure 1): u_2 requires u_1, and u_1 and u_3 mutually require each other. This is a more constrained form of compression than was used in past work (Hirao et al., 2013), but we find that it improves human judgments of fluency (Section 4.3).

Syntactic compressions Figure 2b shows two examples of compressions arising from syntactic patterns (Berg-Kirkpatrick et al., 2011): deletion of the second part of a coordinated NP and deletion of a PP modifier to an NP. These patterns were curated to leave sentences as grammatical after being compressed, though perhaps with damaged semantic content.

Combined compressions Figure 2c shows the textual units and requirement relations yielded by combining these two types of compression. On this example, the two schemes capture orthogonal compressions, and more generally we find that they stack to give better results for our final system (see Section 4.3). To actually synthesize textual units and the constraints between them, we start from the set of RST textual units and introduce syntactic compressions as new children when they don't cross existing brackets; because syntactic compressions are typically narrower in scope, they are usually completely contained in EDUs.

[2]The features in our model are actually rich enough to learn a sophisticated compression model, but the data we have (abstractive summaries) does not directly provide examples of correct compressions; past work has gotten around this with multi-task learning (Almeida and Martins, 2013), but we simply treat grammaticality as a constraint from upstream models.

Figure 2d shows an example of this process: the possible deletion of *with Aetna* is grafted onto the textual unit and appropriate requirement relations are introduced. The net effect is that the textual unit is wholly included, partially included (*with Aetna* removed), or not at all.

Formally, we define an RST tree as $T_{\text{rst}} = (S_{\text{rst}}, \pi_{\text{rst}})$ where S_{rst} is a set of EDU spans (i, j) and $\pi : S \rightarrow 2^S$ is a mapping from each EDU span to EDU spans it depends on. Syntactic compressions can be expressed in a similar way with trees T_{syn}. These compressions are typically smaller-scale than EDU-based compressions, so we use the following modification scheme. Denote by $T_{\text{syn}(kl)}$ a nontrivial (supports some compression) subtree of T_{syn} that is completely contained in an EDU (i, j). We build the following combined compression tree, which we refer to as the *augmentation* of T_{rst} with $T_{\text{syn}(kl)}$:

$$T_{\text{comb}} = (S \cup S_{\text{syn}}(kl) \cup \{(i,k),(l,j)\}, \pi_{\text{rst}} \cup \pi_{\text{syn}(kl)} \cup \{(i,k) \rightarrow (l,j), (l,j) \rightarrow (i,k), (k,l) \rightarrow (i,k)\})$$

That is, we maintain the existing tree structure except for the EDU (i, j), which is broken into three parts: the outer two depend on each other (*is a claims adjuster* and . from Figure 2d) and the inner one depends on the others and preserves the tree structure from T_{syn}. We augment T_{rst} with all maximal subtrees of T_{syn}, i.e. all trees that are not contained in other trees that are used in the augmentation process.

This is broadly similar to the combined compression scheme in Kikuchi et al. (2014) but we use a different set of constraints that more strictly enforce grammaticality.[3]

2.2 Anaphora Constraints

What kind of cross-sentential coherence do we need to ensure for the kinds of summaries our system produces? Many notions of coherence are useful, including centering theory (Grosz et al., 1995) and lexical cohesion (Nishikawa et al., 2014), but one of the most pressing phenomena to deal with is pronoun anaphora (Clarke and Lapata, 2010). Cases of pronouns being "orphaned" during extraction (their antecedents are deleted) are relatively common: they occur in roughly 60% of examples produced by our summarizer when no anaphora constraints are enforced. This kind of error is particularly concerning for summary interpretation and impedes the ability of summaries to convey information effectively (Grice, 1975). Our solution is to explicitly impose constraints on the model based on pronoun anaphora resolution.[4]

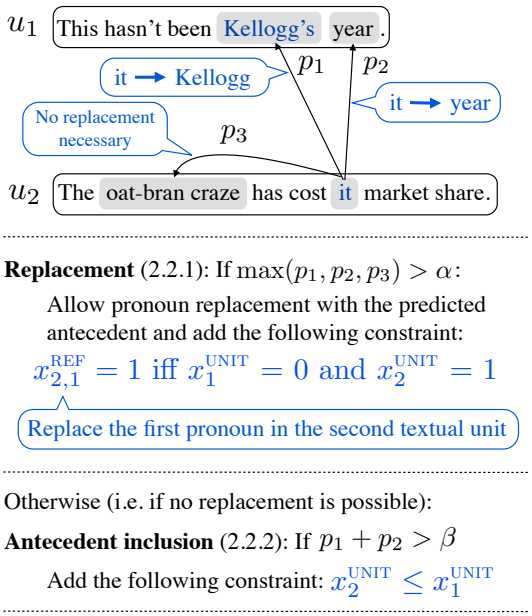


Figure 3: Modifications to the ILP to capture pronoun coherence. *It*, which refers to *Kellogg*, has several possible antecedents from the standpoint of an automatic coreference system (Durrett and Klein, 2014). If the coreference system is confident about its selection (above a threshold α on the posterior probability), we allow for the model to explicitly replace the pronoun if its antecedent would be deleted (Section 2.2.1). Otherwise, we merely constrain one or more probable antecedents to be included (Section 2.2.2); even if the coreference system is incorrect, a human can often correctly interpret the pronoun with this additional context.

Figure 3 shows an example of a problem case. If we extract only the second textual unit shown, the pronoun *it* will lose its antecedent, which in this case is *Kellogg*. We explore two types of constraints for dealing with this: rewriting the pronoun explicitly, or constraining the summary to include the pronoun's antecedent.

2.2.1 Pronoun Replacement

One way of dealing with these pronoun reference issues is to explicitly replace the pronoun with what it refers to. This replacement allows us to maintain maximal extraction flexibility, since we

[3] We also differ from past work in that we do not use cross-sentential RST constraints (Hirao et al., 2013; Yoshida et al., 2014). We experimented with these and found no improvement from using them, possibly because we have a feature-based model rather than a heuristic content selection procedure, and possibly because automatic discourse parsers are less good at recovering cross-sentence relations.

[4] We focus on pronoun coreference because it is the most pressing manifestation of this problem and because existing coreference systems perform well on pronouns compared to harder instances of coreference (Durrett and Klein, 2013).

can make an isolated textual unit meaningful even if it contains a pronoun. Figure 3 shows how this process works. We run the Berkeley Entity Resolution System (Durrett and Klein, 2014) and compute posteriors over possible links for the pronoun. If the coreference system is sufficiently confident in its prediction (i.e. $\max_i p_i > \alpha$ for a specified threshold $\alpha > \frac{1}{2}$), we allow ourselves to replace the pronoun with the first mention of the entity corresponding to the pronoun's most likely antecedent. In Figure 3, if the system correctly determines that *Kellogg* is the correct antecedent with high probability, we enable the first replacement shown there, which is used if u_2 is included the summary without u_1.[5]

As shown in the ILP in Figure 1, we instantiate corresponding pronoun replacement variables $\mathbf{x}^{\text{REF}}$ where $x_{ij}^{\text{REF}} = 1$ implies that the jth pronoun in the ith sentence should be replaced in the summary. We use a candidate pronoun replacement if and only if the pronoun's corresponding (predicted) entity hasn't been mentioned previously in the summary.[6] Because we are generally replacing pronouns with longer mentions, we also need to modify the length constraint to take this into account. Finally, we incorporate features on pronoun replacements in the objective, which helps the model learn to prefer pronoun replacements that help it to more closely match the human summaries.

2.2.2 Pronoun Antecedent Constraints

Explicitly replacing pronouns is risky: if the coreference system makes an incorrect prediction, the intended meaning of the summary may be damaged. Fortunately, the coreference model's posterior probabilities have been shown to be well-calibrated (Nguyen and O'Connor, 2015), meaning that cases where it is likely to make errors are signaled by flatter posterior distributions. In this case, we enable a more conservative set of constraints that include additional content in the summary to make the pronoun reference clear without explicitly replacing it. This is done by requiring the inclusion of any textual unit which contains possible pronoun references whose posteriors sum to at least a threshold parameter β. Figure 3 shows that this constraint can force the inclusion of u_1 to provide additional context. Although this could still lead to unclear pronouns if text is stitched together in an ambiguous or even misleading way, in practice we observe that the textual units we force to be added almost always occur very recently before the pronoun, giving enough additional context for a human reader to figure out the pronoun's antecedent unambiguously.

[5] If the proposed replacement is a proper mention, we replace the pronoun just with the subset of the mention that constitutes a named entity (rather than the whole noun phrase). We control for possessive pronouns by deleting or adding *'s* as appropriate.

[6] Such a previous mention may be a pronoun; however, note that that pronoun would then be targeted for replacement unless its antecedent were included somehow.

2.3 Features

The features in our model (see Figure 1) consist of a set of surface indicators capturing mostly lexical and configurational information. Their primary role is to identify important document content. The first three types of features fire over textual units, the last over pronoun replacements.

Lexical These include indicator features on non-stopwords in the textual unit that appear at least five times in the training set and analogous POS features. We also use lexical features on the first, last, preceding, and following words for each textual unit. Finally, we conjoin each of these features with an indicator of bucketed position in the document (the index of the sentence containing the textual unit).

Structural These features include various conjunctions of the position of the textual unit in the document, its length, the length of its corresponding sentence, the index of the paragraph it occurs in, and whether it starts a new paragraph (all values are bucketed).

Centrality These features capture rough information about the centrality of content: they consist of bucketed word counts conjoined with bucketed sentence index in the document. We also fire features on the number of times of each entity mentioned in the sentence is mentioned in the rest of the document (according to a coreference system), the number of entities mentioned in the sentence, and surface properties of mentions including type and length

Pronoun replacement These target properties of the pronoun replacement such as its length, its sentence distance from the current mention, its type (nominal or proper), and the identity of the pronoun being replaced.

3 Learning

We learn weights $\mathbf{w}$ for our model by training on a large corpus of documents $\mathbf{u}$ paired with reference summaries $\mathbf{y}$. We formulate our learning problem as a standard instance of structured SVM (see Smith (2011) for an introduction). Because we want to optimize explicitly for ROUGE-1,[7] we define a ROUGE-based loss function that accommodates the nature of our supervision, which is in terms of abstractive summaries $\mathbf{y}$ that in general cannot be produced by our model. Specifically, we take:

$$\ell(\mathbf{x}^{\text{NGRAM}}, \mathbf{y}) = \max_{\mathbf{x}^*} \text{ROUGE-1}(\mathbf{x}^*, \mathbf{y}) - \text{ROUGE-1}(\mathbf{x}^{\text{NGRAM}}, \mathbf{y})$$

i.e. the gap between the hypothesis's ROUGE score and the oracle ROUGE score achievable under the model (including constraints). Here $\mathbf{x}^{\text{NGRAM}}$ are indicator variables that track, for each n-gram type in the reference summary, whether that n-gram is present in the system summary. These are the sufficient statistics for computing ROUGE.

We train the model via stochastic subgradient descent on the primal form of the structured SVM objective (Ratliff et al., 2007; Kummerfeld et al., 2015). In order to compute the subgradient for a given training example, we need to find the most violated constraint on the given instance through a loss-augmented decode, which for a linear model takes the form $\arg\max_{\mathbf{x}} \mathbf{w}^\top \mathbf{f}(\mathbf{x}) + \ell(\mathbf{x}, \mathbf{y})$. To do this decode at training time in the context of our model, we use an extended version of our ILP in Figure 1 that is augmented to explicitly track type-level n-grams:

$$\max_{\mathbf{x}^{\text{UNIT}}, \mathbf{x}^{\text{REF}}, \mathbf{x}^{\text{NGRAM}}} \left[\sum_i \left[x_i^{\text{UNIT}} (\mathbf{w}^\top \mathbf{f}(u_i)) \right] + \sum_{(i,j)} \left[x_{ij}^{\text{REF}} (\mathbf{w}^\top \mathbf{f}(r_{ij})) \right] - \ell(\mathbf{x}^{\text{NGRAM}}, \mathbf{y}) \right]$$

subject to all constraints from Figure 1, and

$x_i^{\text{NGRAM}} = 1$ iff an included textual unit or replacement contains the ith reference n-gram

These kinds of variables and constraints are common in multi-document summarization systems that score bigrams (Gillick and Favre, 2009 *inter alia*). Note that since ROUGE is only computed over non-stopword n-grams and pronoun replacements only replace pronouns, pronoun replacement can never remove an n-gram that would otherwise be included.

For all experiments, we optimize our objective using AdaGrad (Duchi et al., 2011) with ℓ_1 regularization ($\lambda = 10^{-8}$, chosen by grid search), with a step size of 0.1 and a minibatch size of 1. We train for 10 iterations on the training data, at which point held-out model performance no longer improves. Finally, we set the anaphora thresholds $\alpha = 0.8$ and $\beta = 0.6$ (see Section 2.2). The values of these and other hyperparameters were determined on a held-out development set from our New York Times training data. All ILPs are solved using GLPK version 4.55.

4 Experiments

We primarily evaluate our model on a roughly 3000-document evaluation set from the New York Times Annotated Corpus (Sandhaus, 2008). We also investigate its performance on the RST Discourse Treebank (Carlson et al., 2001), but because this dataset is only 30 documents it provides much less robust estimates of performance.[8] Throughout this section, when we decode a document, we set the word budget for our summarizer to be the same as the number of words in the corresponding reference summary, following previous work (Hirao et al., 2013; Yoshida et al., 2014).

4.1 Preprocessing

We preprocess all data using the Berkeley Parser (Petrov et al., 2006), specifically the GPU-accelerated version of the parser from Hall et al. (2014), and the Berkeley Entity Resolution System (Durrett and Klein, 2014). For RST discourse analysis, we segment text into EDUs using a semi-Markov CRF trained on the RST treebank with features on boundaries similar to those of Hernault et al. (2010), plus novel features on spans including span length and span identity for short spans.

To follow the conditions of Yoshida et al. (2014) as closely as possible, we also build a discourse parser in the style of Hirao et al. (2013), since their parser is not publicly available. Specifically,

[7] We found that optimizing for ROUGE-1 actually resulted in a model with better performance on both ROUGE-1 and ROUGE-2. We hypothesize that this is because framing our optimization in terms of ROUGE-2 would lead to a less nuanced set of constraints: bigram matches are relatively rare when the reference is a short, abstractive summary, so a loss function based on ROUGE-2 will express a flatter preference structure among possible outputs.

[8] Tasks like DUC and TAC have focused on multi-document summarization since around 2003, hence the lack of more standard datasets for single-document summarization.

NYT50 article:

Federal officials reported yesterday that students in 4th, 8th and 12th grades had scored modestly higher on an American history test than five years earlier, although more than half of high school seniors still showed poor command of basic facts like the effect of the cotton gin on the slave economy or the causes of the Korean War. Federal officials said they considered the results encouraging because at each level tested, student performance had improved since the last time the exam was administered, in 2001. "In U.S. history there were higher scores in 2006 for all three grades," said Mark Schneider, commissioner of the National Center for Education Statistics, which administers the test, at a Boston news conference that the Education Department carried by Webcast. **The results were less encouraging on a national civics test, on which only fourth graders made any progress.** The best results in the history test were also in fourth grade, where 70 percent of students attained the basic level of achievement or better. **The test results in the two subjects are likely to be closely studied, because Congress is considering the renewal of President Bush's signature education law, the No Child Left Behind Act.** A number of studies have shown that because No Child Left Behind requires states…

Summary:

National Center for Education Statistics reports students in 4th, 8th and 12th grades scored modestly higher on American history test than five years earlier. Says more than half of high school seniors still show poor command of basic facts. Only 4th graders made any progress in civics test. New exam results are another ingredient in debate over renewing Pres Bush's signature No Child Left Behind Act.

Filtered article:

Long before President Bush's proposal to rethink Social Security became part of the national conversation, Westchester County came up with its own dialogue to bring issues of aging to the forefront. Before the White House Conference on Aging scheduled in October, **the county's Office for the Aging a year ago started Speak-Up,** which stands for Student Participants Embrace Aging Issues of Key Concern, to reach students in the county's 13 colleges and universities. Through a variety of events **to bring together the elderly and college students,** organizers said they hoped to have by this spring a series of recommendations that could be given to Washington…

Summary:

Article on Speak-Up, program begun by Westchester County Office for the Aging to bring together elderly and college students.

Figure 4: Examples of an article kept in the NYT50 dataset (top) and an article removed because the summary is too short. The top summary has a rich structure to it, corresponding to various parts of the document (bolded) and including some text that is essentially a direct extraction.

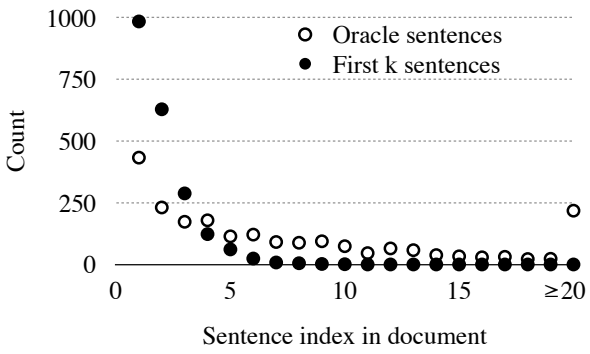


Figure 5: Counts on a 1000-document sample of how frequently both a document prefix baseline and a ROUGE oracle summary contain sentences at various indices in the document. There is a long tail of useful sentences later in the document, as seen by the fact that the oracle sentence counts drop off relatively slowly. Smart selection of content therefore has room to improve over taking a prefix of the document.

we use the first-order projective parsing model of McDonald et al. (2005) and features from Soricut and Marcu (2003), Hernault et al. (2010), and Joty et al. (2013). When using the same head annotation scheme as Yoshida et al. (2014), we outperform their discourse dependency parser on unlabeled dependency accuracy, getting 56% as opposed to 53%.

4.2 New York Times Corpus

We now provide some details about the New York Times Annotated corpus. This dataset contains 110,540 articles with abstractive summaries; we split these into 100,834 training and 9706 test examples, based on date of publication (test is all articles published on January 1, 2007 or later). Examples of two documents from this dataset are shown in Figure 4. The bottom example demonstrates that some summaries are extremely short and formulaic (especially those for obituaries and editorials). To counter this, we filter the raw dataset by removing all documents with summaries that are shorter than 50 words. One benefit of filtering is that the length distribution of our resulting dataset is more in line with standard summarization evaluations like DUC; it also ensures a sufficient number of tokens in the budget to produce nontrivial summaries. The filtered test set, which we call NYT50, includes 3,452 test examples out of the original 9,706.

Interestingly, this dataset is one where the classic document prefix baseline can be substantially outperformed, unlike in some other summarization settings (Penn and Zhu, 2008). We show this fact explicitly in Section 4.3, but Figure 5 provides additional analysis in this regard. We compute oracle ROUGE-1 sentence-extractive summaries on a 1000-document subset of the training set and look at where the extracted sentences lie in the document. While they certainly skew earlier in the document, they do not all fall within the doc-

	R-1 ↑	R-2 ↑	CG ↑	UP ↓
Baselines				
First sentences	28.6	17.3	8.21	0.28
First k words	35.7	21.6	–	–
Bigram Frequency	25.1	9.8	–	–
Past work				
Tree Knapsack	34.7	19.6	7.20	0.42
This work				
Sentence extraction	38.8	23.5	7.93	0.32
EDU extraction	41.9	25.3	6.38	0.65
Full	**42.2**	**25.9**	*†7.52	*0.36
Ablations from Full				
No Anaphoricity	42.5	26.3	7.46	0.44
No Syntactic Compr	41.1	25.0	–	–
No Discourse Compr	40.5	24.7	–	–

Table 1: Results on the NYT50 test set (documents with summaries of at least 50 tokens) from the New York Times Annotated Corpus (Sandhaus, 2008). We report ROUGE-1 (R-1), ROUGE-2 (R-2), clarity/grammaticality (CG), and number of unclear pronouns (UP) (lower is better). On content selection, our system substantially outperforms all baselines, our implementation of the tree knapsack system (Yoshida et al., 2014), and learned extractive systems with less compression, even an EDU-extractive system that sacrifices grammaticality. On clarity metrics, our final system performs nearly as well as sentence-extractive systems. The symbols * and † indicate statistically significant gains compared to No Anaphoricity and Tree Knapsack (respectively) with $p < 0.05$ according to a bootstrap resampling test. We also see that removing either syntactic or EDU-based compressions decreases ROUGE.

ument prefix summary. One reason for this is that many of the articles are longer-form pieces that begin with a relatively content-free lede of several sentences, which should be identifiable with lexicosyntactic indicators as are used in our discriminative model.

4.3 New York Times Results

We evaluate our system along two axes: first, on content selection, using ROUGE[9] (Lin and Hovy, 2003), and second, on clarity of language and referential structure, using annotators from Amazon Mechanical Turk. We follow the method of Gillick and Liu (2010) for this evaluation and ask Turkers to rate a summary on how grammatical it is using a 10-point Likert scale. Furthermore, we ask how many unclear pronouns references there were in the text. The Turkers do not see the original document or the reference summary, and rate each summary in isolation. Gillick and Liu (2010) showed that for linguistic quality judgments (as opposed to content judgments), Turkers reproduced the ranking of systems according to expert judgments.

To speed up preprocessing and training time on this corpus, we further restrict our training set to only contain documents with fewer than 100 EDUs. All told, the final system takes roughly 20 hours to make 10 passes through the subsampled training data (22,000 documents) on a single core of an Amazon EC2 r3.4xlarge instance.

Table 1 shows the results on the NYT50 corpus. We compare several variants of our system and baselines. For baselines, we use two variants of first k: one which must stop on a sentence boundary (which gives better linguistic quality) and one which always consumes k tokens (which gives better ROUGE). We also use a heuristic sentence-extractive baseline that maximizes the document counts (term frequency) of bigrams covered by the summary, similar in spirit to the multi-document method of Gillick and Favre (2009).[10] We also compare to our implementation of the Tree Knapsack method of Yoshida et al. (2014), which matches their results very closely on the RST Discourse Treebank when discourse trees are controlled for. Finally, we compare several variants of our system: purely extractive systems operating over sentences and EDUs respectively, our full system, and ablations removing either the anaphoricity component or parts of the compression module.

In terms of content selection, we see that all of the systems that incorporate end-to-end learning (under "This work") substantially outperform our various heuristic baselines. Our full system using the full compression scheme is substantially better on ROUGE than ablations where the syntactic or discourse compressions are removed. These improvements reflect the fact that more compression options give the system more flexibility to include key content words. Removing the anaphora resolution constraints actually causes ROUGE to increase slightly (as a result of granting the model flexibility), but has a negative impact on the linguistic quality metrics.

On our linguistic quality metrics, it is no surprise that the sentence prefix baseline performs the best. Our sentence-extractive system also does well on these metrics. Compared to the EDU-extractive system with no constraints, our constrained compression method improves substantially on both linguistic quality and reduces the

[9] We use the ROUGE 1.5.5 script with the following command line arguments: `-n 2 -x -m -s`. All given results are macro-averaged recall values over the test set.

[10] Other heuristic multi-document approaches could be compared to, e.g. He et al. (2012), but a simple term frequency method suffices to illustrate how these approaches can underperform in the single-document setting.

	ROUGE-1	ROUGE-2
First k words	23.5	8.3
Tree Knapsack	25.1	8.7
Full	26.3	8.0

Table 2: Results for RST Discourse Treebank (Carlson et al., 2001). Differences between our system and the Tree Knapsack system of Yoshida et al. (2014) are not statistically significant, reflecting the high variance in this small (20 document) test set.

number of unclear pronouns, and adding the pronoun anaphora constraints gives further improvement. Our final system is approaches the sentence-extractive baseline, particularly on unclear pronouns, and achieves substantially higher ROUGE score.

4.4 RST Treebank

We also evaluate on the RST Discourse Treebank, of which 30 documents have abstractive summaries. Following Hirao et al. (2013), we use the gold EDU segmentation from the RST corpus but automatic RST trees. We break this into a 10-document development set and a 20-document test set. Table 2 shows the results on the RST corpus. Our system is roughly comparable to Tree Knapsack here, and we note that none of the differences in the table are statistically significant. We also observed significant variation between multiple runs on this corpus, with scores changing by 1-2 ROUGE points for slightly different system variants.[11]

5 Conclusion

We presented a single-document summarization system trained end-to-end on a large corpus. We integrate a compression model that enforces grammaticality as well as pronoun anaphoricity constraints that enforce coherence. Our system improves substantially over baseline systems on ROUGE while still maintaining good linguistic quality.

Our system and models are publicly available at `http://nlp.cs.berkeley.edu`

[11] The system of Yoshida et al. (2014) is unavailable, so we use a reimplementation. Our results differ from theirs due to having slightly different discourse trees, which cause large changes in metrics due to high variance on the test set.

Acknowledgments

This work was partially supported by NSF Grant CNS-1237265 and a Google Faculty Research Award. Thanks to Tsutomu Hirao for providing assistance with our reimplementation of the Tree Knapsack model, and thanks the anonymous reviewers for their helpful comments.

References

Miguel Almeida and Andre Martins. 2013. Fast and Robust Compressive Summarization with Dual Decomposition and Multi-Task Learning. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Regina Barzilay and Mirella Lapata. 2008. Modeling Local Coherence: An Entity-based Approach. *Computational Linguistics*, 34(1):1–34, March.

Regina Barzilay and Lillian Lee. 2004. Catching the Drift: Probabilistic Content Models, with Applications to Generation and Summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2001. Sentence Ordering in Multidocument Summarization. In *Proceedings of the International Conference on Human Language Technology Research*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly Learning to Extract and Compress. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Jaime Carbonell and Jade Goldstein. 1998. The Use of MMR, Diversity-based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*.

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

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2013. Towards Coherent Multi-Document Summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

James Clarke and Mirella Lapata. 2008. Global Inference for Sentence Compression an Integer Linear Programming Approach. *Journal of Artificial Intelligence Research*, 31(1):399–429, March.

James Clarke and Mirella Lapata. 2010. Discourse Constraints for Document Compression. *Computational Linguistics*, 36(3):411–441, September.

Hal Daumé, III and Daniel Marcu. 2002. A Noisy-Channel Model for Document Compression. In *Proceedings of the Association for Computational Linguistics (ACL)*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12:2121–2159, July.

Jesse Dunietz and Daniel Gillick. 2014. A New Entity Salience Task with Millions of Training Examples. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Greg Durrett and Dan Klein. 2013. Easy Victories and Uphill Battles in Coreference Resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, October.

Greg Durrett and Dan Klein. 2014. A Joint Model for Entity Analysis: Coreference, Typing, and Linking. In *Transactions of the Association for Computational Linguistics (TACL)*.

Dan Gillick and Benoit Favre. 2009. A Scalable Global Model for Summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*.

Dan Gillick and Yang Liu. 2010. Non-Expert Evaluation of Summarization Systems is Risky. In *Proceedings of the NAACL Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

H.P. Grice. 1975. Logic and Conversation. *Syntax and Semantics 3: Speech Acts*, pages 41–58.

Barbara J. Grosz, Scott Weinstein, and Aravind K. Joshi. 1995. Centering: A Framework for Modeling the Local Coherence of Discourse. *Computational Linguistics*, 21(2):203–225, June.

David Hall, Taylor Berg-Kirkpatrick, John Canny, and Dan Klein. 2014. Sparser, Better, Faster GPU Parsing. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Zhanying He, Chun Chen, Jiajun Bu, Can Wang, Lijun Zhang, Deng Cai, and Xiaofei He. 2012. Document Summarization Based on Data Reconstruction. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Hugo Hernault, Helmut Prendinger, David A. Duverle, Mitsuru Ishizuka, and Tim Paek. 2010. HILDA: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1:1–33.

Tsutomu Hirao, Yasuhisa Yoshida, Masaaki Nishino, Norihito Yasuda, and Masaaki Nagata. 2013. Single-Document Summarization as a Tree Knapsack Problem. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Kai Hong and Ani Nenkova. 2014. Improving the Estimation of Word Importance for News Multi-Document Summarization. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

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

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single Document Summarization based on Nested Tree Structure. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein. 2015. An Empirical Analysis of Optimization for Max-Margin NLP. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Chen Li, Yang Liu, and Lin Zhao. 2015. Using External Resources and Joint Learning for Bigram Weighting in ILP-Based Multi-Document Summarization. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Hui Lin and Jeff Bilmes. 2011. A Class of Submodular Functions for Document Summarization. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Chin-Yew Lin and Eduard Hovy. 2003. Automatic Evaluation of Summaries Using N-gram Co-Occurrence Statistics. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Chin-Yew Lin. 2003. Improving Summarization Performance by Sentence Compression: A Pilot Study. In *Proceedings of the International Workshop on Information Retrieval with Asian Languages*.

Annie Louis and Ani Nenkova. 2012. A Coherence Model Based on Syntactic Patterns. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Annie Louis, Aravind Joshi, and Ani Nenkova. 2010. Discourse Indicators for Content Selection in Summarization. In *Proceedings of the SIGDIAL 2010 Conference*.

Inderjeet Mani. 2001. *AutomaticSummarization*. John Benjamins Publishing.

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

Daniel Marcu. 1998. Improving summarization through rhetorical parsing tuning. In *Proceedings of the Workshop on Very Large Corpora*.

Andre Martins and Noah A. Smith. 2009. Summarization with a Joint Model for Sentence Extraction and Compression. In *Proceedings of the Workshop on Integer Linear Programming for Natural Language Processing*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-margin Training of Dependency Parsers. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Ryan McDonald. 2006. Discriminative Sentence Compression With Soft Syntactic Evidence. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)*.

Kathleen McKeown, Jacques Robin, and Karen Kukich. 1995. Generating Concise Natural Language Summaries. *Information Processing and Management*, 31(5):703–733, September.

Shamima Mithun and Leila Kosseim. 2011. Discourse Structures to Reduce Discourse Incoherence in Blog Summarization. In *Proceedings of Recent Advances in Natural Language Processing*.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Foundations and Trends in Information Retrieval*, 5(2?3):103–233.

Khanh Nguyen and Brendan O'Connor. 2015. Posterior Calibration and Exploratory Analysis for Natural Language Processing Models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hitoshi Nishikawa, Kazuho Arita, Katsumi Tanaka, Tsutomu Hirao, Toshiro Makino, and Yoshihiro Matsuo. 2014. Learning to Generate Coherent Summary with Discriminative Hidden Semi-Markov Model. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Gerald Penn and Xiaodan Zhu. 2008. A Critical Reassessment of Evaluation Baselines for Speech Summarization. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the Conference on Computational Linguistics and the Association for Computational Linguistics (ACL-COLING)*.

Daniele Pighin, Marco Cornolti, Enrique Alfonseca, and Katja Filippova. 2014. Modelling Events through Memory-based, Open-IE Patterns for Abstractive Summarization. In *Proceedings of the Association for Computational Linguistics (ACL)*.

Nathan J. Ratliff, Andrew Bagnell, and Martin Zinkevich. 2007. (Online) Subgradient Methods for Structured Prediction. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Evan Sandhaus. 2008. The New York Times Annotated Corpus. In *Linguistic Data Consortium*.

Noah A. Smith. 2011. *Linguistic Structure Prediction*. Morgan & Claypool Publishers, 1st edition.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Kapil Thadani and Kathleen McKeown. 2013. Supervised Sentence Fusion with Single-Stage Inference. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*.

Kristian Woodsend and Mirella Lapata. 2012. Multiple Aspect Summarization Using Integer Linear Programming. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*.

Yasuhisa Yoshida, Jun Suzuki, Tsutomu Hirao, and Masaaki Nagata. 2014. Dependency-based Discourse Parser for Single-Document Summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Set-Theoretic Alignment for Comparable Corpora

Thierry Etchegoyhen and **Andoni Azpeitia**
Vicomtech-IK4
Mikeletegi Pasalekua, 57
Donostia / San Sebastián, Gipuzkoa, Spain
{tetchegoyhen, aazpeitia}@vicomtech.org

Abstract

We describe and evaluate a simple method to extract parallel sentences from comparable corpora. The approach, termed STACC, is based on expanded lexical sets and the Jaccard similarity coefficient. We evaluate our system against state-of-the-art methods on a large range of datasets in different domains, for ten language pairs, showing that it either matches or outperforms current methods across the board and gives significantly better results on the noisiest datasets. STACC is a portable method, requiring no particular adaptation for new domains or language pairs, thus enabling the efficient mining of parallel sentences in comparable corpora.

1 Introduction

With the rise of data-driven machine translation, be it statistical (Brown et al., 1990), example-based (Nagao, 1984), or rooted in neural networks (Bahdanau et al., 2014), the need for large parallel corpora has increased accordingly. Although quality bitexts have been made available over the years (Tiedemann, 2012), creating parallel corpora is a resource-consuming effort involving professional human translation of large volumes of texts in multiple languages. As a consequence, there is still a lack of parallel data to properly model translation across languages and domains.

To overcome this limitation, special emphasis has been placed in the last two decades on the exploitation of comparable corpora, with the development of a range of methods to mine parallel sentences from texts addressing similar topics in different languages. The work we present follows this line of research, describing and evaluating a simple method that allows parallel sentences to be efficiently mined in different languages and domains with minimal adaptation effort.

The method we describe, termed STACC, is based on expanded lexical sets and the Jaccard similarity coefficient (Jaccard, 1901), which is computed as the ratio of set intersection over union. We evaluate this simple approach against state-of-the-art methods for comparable sentence alignment on a variety of datasets for ten different language pairs, showing that STACC either matches or outperforms competing approaches.

The paper is organised as follows: Section 2 describes related work on parallel sentence mining in comparable corpora; Section 3 presents the STACC method; Section 4 describes the experiments in comparable sentence alignment, including the description of test corpora and systems, and an analysis of the results; Section 5 presents results obtained with an optimised version of the alignment process, beyond system comparison; finally, Section 6 draws conclusions from the work described in the paper.

2 Related work

A large variety of techniques have been proposed to mine parallel sentences in comparable corpora. One of the first approaches was proposed by (Zhao and Vogel, 2002), who combined sentence length and bilingual lexicon models under a maximum likelihood criterion. (Munteanu and Marcu, 2002) explored the use of suffix trees, later opting for maximum entropy-based binary classification using a modified version of IBM Model 1 word translation probabilities (Brown et al., 1993)

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2009–2018,
Berlin, Germany, August 7-12, 2016.

and both general and alignment-specific features (Munteanu and Marcu, 2005). (Fung and Cheung, 2004) describe the first approach to tackle parallel sentence mining in very non-parallel corpora, using cosine similarity as their sentence selection criterion.

Several approaches have employed full statistical machine translation models instead of relying only on lexical tables. (Abdul-Rauf and Schwenk, 2009), for instance, apply the TER metric (Snover et al., 2006) on fully machine translated output to identify parallel sentences; (Sarikaya et al., 2009) use a similar approach but with BLEU (Papineni et al., 2002) as their similarity metric. One of the noted advantages of including full machine translation is the ability to better model the complex factors found in translation, e.g. fertility and contextual information, as compared to lexicon-based approaches. The latter enable, in principle, the capture of a larger set of lexical translation variants, and do not require the training of complete translation models.

Sophisticated feature-based approaches have been developed in recent years in order to provide a method that may apply to larger sets of language pairs and domains. (Stefănescu et al., 2012) report improvements over previous methods with a feature-based sentence similarity measure, an approach which is described in more detail in Section 4.2.1. Another feature-rich approach is described in (Smith et al., 2010), showing improvements over standard and improved binary classifiers; we describe their model in more details in Section 4.2.2.

Jaccard similarity, a core component of the approach we describe, has been standardly used as a text similarity measure in information retrieval and text summarisation tasks, or to compute semantic similarity (Pilehvar et al., 2013). For comparable corpora, it has been notably employed by (Paramita et al., 2013), who estimate document comparability by computing the coefficient on a subset of translated source sentences, discarding those containing large amounts of named entities or numbers, and taking the average of these sentence-level scores. The method we present in the next section builds on a related similarity measure as a direct indicator of comparable sentence similarity.

3 STACC

STACC is an approach to sentence similarity based on expanded lexical sets, whose main goal is to provide a simple yet effective procedure that can be applied across domains and corpora with minimal adaptation and deployment costs.

We start with the minimal set of bilingual information that can be automatically extracted from a seed parallel corpus, using lexical translations determined and ranked according to IBM models; word translations are computed in both directions using the GIZA++ toolkit (Och and Ney, 2003).

STACC relies on the Jaccard index, which defines set similarity as the ratio of set intersection over union. We base our comparable sentence similarity measure strictly on this index, applying it to expanded lexical sets as described below.

Let s_i and s_j be two tokenised and truecased sentences in languages l_1 and l_2, respectively, S_i the set of tokens in s_i, S_j the set of tokens in s_j, T_{ij} the set of expanded translations into l_2 for all tokens in S_i, and T_{ji} the set of expanded translations into l_1 for all tokens in S_j. The STACC similarity score is then computed as in Equation 1:

$$sim_{stacc} = \frac{\frac{|T_{ij} \cap S_j|}{|T_{ij} \cup S_j|} + \frac{|T_{ji} \cap S_i|}{|T_{ji} \cup S_i|}}{2} \quad (1)$$

That is, the score is defined as the average of the Jaccard similarity coefficients obtained between sentence token sets and expanded lexical translations in both directions.

The translation sets T_{ij} and T_{ji} are initially computed from sentences s_i and s_j by retaining the k-best lexical translations found in GIZA tables, if any. Lexical translations are selected according to the ranking provided by the precomputed lexical probabilities but the specific probability values are not used any further to compute similarity:[1] all potential translations are members of the translation set as tokens. Discarding this source of potentially exploitable information is mostly motivated by the relative reliability of lexical translation probabilities across domains. Lexical translations are usually extracted from a different domain than that of the comparable corpora at hand, typically using professionally created institutional corpora such as Europarl (Koehn, 2005), and lexical distributions across

[1] This differs from (Skadiņa et al., 2012), who include a lexical translation feature where actual probabilities are used to compute the final score.

domains can be expected to be quite different. This casts doubt on the usefulness of using precomputed translation probabilities and simple set membership was favoured in our approach.

The initial lexical translation sets undergo a first expansion step to capture morphological variation, using longest common prefix matching (hereafter, LCP). To apply prefix matching to the minimal set of elements necessary, we compute the following two set differences:

- Set of elements in the source to target translation set that are not members of the target token set: $T'_{ij} = T_{ij} - S_j$
- Set of elements in the target to source translation set that are not members of the source token set: $T'_{ji} = T_{ji} - S_i$

For each element in T'_{ij} (respectively T'_{ji}) and each element in S_j (respectively S_i), if a common prefix is found with a minimal length of more than n characters, the prefix is added to both translation sets.[2]

This simplified approach to stemming removes the need to rely on manually constructed endings lists to compute similarity or on a complete morphological analyser, which might not be available at all for under-resourced languages. It is also computationally more efficient as it exploits the nature of the alignment problem to reduce the search space: instead of matching each source and target word against every potential ending, with hundreds of possible endings in some languages, only the prefixes of word pairs within the subsets created through set difference need to be compared using LCP.

Another set expansion operation is defined to handle named entities, which are strong indicators of potential alignment, given their low relative frequency, and are likely to be missing from translation tables trained on a different domain. While creating the previously defined lexical translation sets from truecased sentences, capitalised tokens that are not found in the translation tables are added to the translation sets. Numbers are similarly handled and added to the expanded sets, as they can also act as alignment indicators, in particular when they denote dates.

These two expansions steps are essential to a successful use of Jaccard similarity for comparable sentence alignment. For instance, LCP gives a 2.9 points improvement in F1 measure on the initial Basque-Spanish test set described in Section 4.1, whereas the NE/Number expansion resulted in a 1.3 points gain; the two expansions combined gave a 4.3 points increase in terms of F1 measure. For the English-Bulgarian pair on the initial Wikipedia test set, the gains were 3.7, 2.6 and 5.5, respectively. Combining the two operations thus contributed to the improvements over the state of the art described in Section 4.3.

No additional operations are performed on the created sets, and in particular no filtering is applied, with punctuation and functional words kept alongside content words in the final sets. This notably eliminates the use of stop word lists from the computation of similarity.

Although it builds on fairly standard ideas, such as the use of GIZA tables or the Jaccard index, the approach is original in its conjoined use of these elements with surface-based information and simple set-theoretic operations to form a similarity assessment mechanism that proved efficient on comparable corpora, as shown in the next section.

4 Comparable sentence alignment

We performed a systematic comparison between different approaches to comparable sentence alignment on a variety of comparable corpora and language pairs. This section describes the components of the experimental setup.

4.1 Corpora

Three core sets of corpora were used in the evaluation, which we describe in turn. The selected test sets, all manually aligned, were used in different settings with gradual amounts of alignment noise added to the original sets. The goal of noisification is to assess the behavior of each approach in different scenarios and evaluate their ability to properly align data from ideal conditions to gradually noisier environments, the latter being a more realistic case when dealing with comparable corpora.

The first corpus consists in the public datasets created within the Accurat project.[3] The corpus covers 7 language pairs, each one composed of English and an under-resourced language. The datasets contain manually verified alignments that were created from news articles. We noisified these datasets by adding sentences from the

[2] Throughout the experiments we describe, n was set to 3.

[3] http://www.accurat-project.eu/. The corpus is available from: http://metashare.elda.org/repository/search/?q=accurat

TEST SETS	EN-DE	EN-EL	EN-ET	EN-LT	EN-LV	EN-RO	EN-SL
1:1	ATS: 512	ATS: 512	ATS: 512	ATS: 512	ATS: 512	ATS: 512	ATS: 512
2:1	ATS: 512 AOC: 512	ATS: 512 AOC: 512	ATS: 512 AOC: 512	ATS: 512 AOC: 512	ATS: 512 AOC: 512	ATS: 512 AOC: 512	ATS: 512 AOC: 512
100:1	ATS: 512 AOC: 6891 EUP: 43797	ATS: 512 AOC: 24276 EUP: 26412	ATS: 512 AOC: 50688	ATS: 512 AOC: 50688	ATS: 512 AOC: 50688	ATS: 512 AOC: 50688	ATS: 512 AOC: 15857 EUP: 34831

Table 1: Accurat evaluation sets

TEST SETS	BG-EN	DE-EN	ES-EN
1:1	WTS: 516	WTS: 314	WTS: 500
100:1	WTS: 516 EUP: 51084	WTS: 314 NC: 31086	WTS: 500 NC: 49500

Table 2: Wikipedia evaluation sets

TEST SETS	ES-EU
1:1	500-500
EITB_NOISE1	1000-1000
EITB_NOISE2	1000-1500

Table 3: EITB evaluation sets

original comparable corpora collected within the project, creating the following additional variants: (i) a 2:1 noisified version, where for each sentence in the original sets, 2 additional sentences without corresponding alignments were added; and (ii) a 100:1 noisified version with 100 sentences added for each sentence in the test sets. For each language pair, the additional sentences were taken from the initial portion of the selected additional corpora in one language and the final portion in the other language. For the 2:1 datasets, and the 100:1 variants in some language pairs, the original comparable corpora were used as additional data. For other language pairs, creating the 100:1 variant required adding sentences from different corpora to reach the required amount of data. Table 1 describes the final datasets used in the evaluation.[4]

As a second corpus, we used the data described in (Smith et al., 2010).[5] The texts were extracted from Wikipedia articles in 3 language pairs (English-German, English-Spanish and English-Bulgarian) and manually annotated for parallelism. We used the provided test sets (hereafter, WTS) and added a 100:1 noisified variant using sentences from the News Crawl corpus[6] for English-German and English-Spanish, and from Europarl for the English-Bulgarian pair. Table 2 describes these datasets, to which we will refer collectively as the Wikipedia corpus.

Finally, we used the EITB corpus, composed of news generated by the Basque Country's public broadcasting service.[7] The news are written independently in Basque and Spanish but refer to the same specific events and the corpus can thus be categorized as strongly comparable. We defined initial test sets of 500 manually aligned sentences in each language, and created two noisified variants: (i) a test set with 500 additional sentences in both languages, and (ii) a test set with 500 additional sentences in Spanish and 1000 in Basque. All additional sentences were taken from unaligned portions of the same EITB corpus. Table 3 summarises the EITB test sets.

The selected corpora thus cover 10 different language pairs and different domains, with varying degrees of noisification, and provide for a large and diverse comparison set.

4.2 Systems

Three approaches were evaluated against the previously described corpora: LEXACC (Stefănescu et al., 2012), the STACC method described in Section 3, and the approach based on Conditional Random Fields described in (Smith et al., 2010), to which we will refer as CRF. The latter was only evaluated on the Wikipedia corpus, using the re-

[4]In the table, ATS refers to the Accurat test sets, AOC to the Accurat original corpora, and EUP to the Europarl corpus.

[5]Available at: http://research.microsoft.com/en-us/people/chrisq/wikidownload.aspx.

[6]Refered to as NC here and available from: http://www.statmt.org/wmt13/translation-task.html.

[7]Euskal Irrati Telebista (EITB): http://www.eitb.eus. The corpus was provided courtesy of EITB and will be made available to the research community.

sults reported in the aforementioned article, as the tools to apply this method were not available to us; both LEXACC and STACC were evaluated on all test sets.

LEXACC was selected given its reported performance and its aim at portability across domains and language pairs; the system is also available as part of the Accurat toolkit,[8] which allowed for a direct comparison with STACC on all datasets.

The CRF approach has proven more effective than standard classifier-based methods on the Wikipedia datasets, with published results on publically available test sets, and was thus selected as an alternative approach to comparable sentence alignment.

Both approaches are based on sophisticated methods with demonstrated improvements over the state-of-the-art, thus providing strong baselines for system comparison.

4.2.1 LEXACC

LEXACC is a fast parallel sentence mining system based on a cross-linguistic information retrieval (CLIR) approach. It uses the Lucene search engine[9] in two major steps: target sentences are first indexed by the search engine, and a search query is built from a translation of content words in the source sentence to retrieve alignment candidates. The query is constructed using IBM Model 1 lexical translation tables, extracted from seed parallel corpora

The alignment metric in LEXACC is a translation similarity measure based on 5 feature functions briefly described here (see (Ștefănescu et al., 2012) for a detailed description):

- f_1 measures source-target candidate pairs strength in terms of content word translation and string similarity;
- f_2 is similar to f_1 but applies to functional words, as identified in manually created stop word lists;
- f_3 measures content word alignment obliqueness defined as a discounted correlation measure;
- f_4 is a binary feature that compares the number of initial/final aligned word translations over a pre-defined threshold;
- f_5 is a second binary feature which evaluates if the source and target sentences end with the same punctuation.

The similarity measure is then computed according to the sum of weighted feature functions, with optimal weights determined by means of logistic regression. We used the optimal feature weights described in (Ștefănescu et al., 2012) for the language pairs in the Accurat corpus and the provided default weights for English-Spanish and English-Bulgarian; for Basque-Spanish, optimal weights were estimated through logistic regression on a training set formed with 9500 positive parallel examples from the IVAP corpus[10] and an equal amount of non-parallel negative examples.

For the experiments, all lexical translation tables were created with GIZA++ on the JRC-Acquis Communautaire corpus.[11] Lucene searches were set to return a maximum of 100 candidates for each source sentence. We used the default setup for LEXACC, except for two minor changes. First, we removed the initial Lucene search constraint which was set to discard identical source and target sentences, a setting which prevented the retrieval of valid news candidates such as sports results. Secondly, we increased the length ratio filter from 1.5 to 7.5, as the initial value was too restrictive for the Basque-Spanish corpus. Both changes were thus meant to retrieve the most accurate set of alignment candidates, in order to get meaningful results on the test sets with both methods.

4.2.2 Conditional Random Fields

The model we refer to as CRF (Smith et al., 2010) is a first order linear chain Conditional Random Field (Lafferty et al., 2001), where for each source sentence a hidden variable indicates the corresponding target sentence to which it is aligned, or null if there is no such target sentence. This system was compared to the standard binary classifier of (Munteanu and Marcu, 2005) and to a ranking variant designed by the authors to avoid class imbalance issues that arise with binary classification. On the Wikipedia test sets, the CRF approach gave

[8] http://www.accurat-project.eu/index.php?p=accurat-toolkit

[9] http://lucenenet.apache.org/

[10] Extracted from the translation memories released by the Basque Public Administration Institute (http://opendata.euskadi.eus/catalogo/-/memorias-de-traduccion-del-servicio-oficial-de-traductores-del-ivap/), which consist of professional translations of public administration texts.

[11] We used the latest available version of the corpus, as of November 2015, in the OPUS repository: http://opus.lingfil.uu.se/JRC-Acquis.php.

the best results overall and was thus selected for our system comparison.

The sequence model comprises the following features:

- A word alignment feature set, based on IBM Model 1 and HMM alignments, which includes: log probability of the alignment; number of aligned/unaligned words; longest aligned/unaligned sequence of words; and number of words for different degrees of fertility.
- Two sentence-related features: source and target length ratio modeled through a Poisson distribution (Moore, 2002), and relative position of source and target sentences in the document.
- A set of distortion features measuring the difference in position between the previous and current aligned sentences.
- A set of features based on Wikipedia markup, including matching and non-matching links for alignment candidates.
- A set of lexicon features based on a probabilistic model of word pair alignments, trained on a set of annotated Wikipedia articles. The lexicon-based feature set includes the HMM translation probability, word-based positional differences, orthographic similarity, context translation similarity and distributional similarity.

The seed parallel data were based on the Europarl corpus for Spanish and German and the JRC-Aquis corpus for Bulgarian. The authors also included article titles of parallel Wikipedia documents and Wiktionary translations as additional seed data.

4.2.3 STACC

In order to establish a fair comparison between LEXACC and STACC, all shared settings were identical. Thus, lexical translations were based on the same previously described GIZA tables extracted from the JRC corpus, and STACC alignment was performed on the same sets of candidates retrieved from the Lucene searches by LEXACC for each language pair.

As described in Section 3, STACC is based on the k-best translations provided by lexical translation tables. For the experiments, k was set to 5, a value arbitrarily determined to be an optimal compromise between overcrowding the sets with unlikely translations and limiting translation candidates to minimal translation variants. Experimenting with different values on the test sets showed that this value for k was not actually the optimal one for some language pairs, with e.g. a 2.9 point gain in F1 measure when setting k to 2 for English-Greek on the initial Accurat test set.[12]

The results we present in the next section are thus not the best achievable ones using the STACC approach. Nonetheless, we maintained the use of a default value because of the lack of in-domain development sets on which an optimal value could be fairly computed.

4.3 Results

To evaluate the accuracy of the tested methods, precision was taken as the ratio of correct alignments over predicted alignments, and recall as the ratio of correct alignments over true alignments. We present results in terms of F1 measure, as we seek an optimal balance between alignment precision and recall.

Table 4 presents the results on the Accurat test sets for LEXACC and STACC using their respective optimal similarity thresholds.[13] On the 21 test sets, the two systems were tied on two occasions, with STACC obtaining better results in 89.5% of the remaining cases. On the noisiest datasets, STACC was consistently and markedly better across language pairs.

The results on the Wikipedia test sets are shown in Table 5. For English-Spanish and English-German, both approaches performed quite similarily on the initial test sets, with STACC obtaining the best results on the noisier sets.

The results for English-Bulgarian are interesting, as this is the only case where LEXACC outperforms STACC on both the clean and noisy datasets. The data used for noisification in this case may have had an effect on the results. Data extracted from Europarl, which compose the entire noisifi-

[12] Note that similar issues would arise if the selected translations were determined based on thresholds over translation probabilities, as the thresholds would need to be empirically set as well.

[13] The optimal thresholds were determined as the values providing the best results on the test sets. This would obviously not be an available threshold selection method when mining comparable corpora, where a default value would have to be used instead. Such a default value would however not allow for a fair comparison of the systems.

SYSTEM	TEST SETS	EN-DE	EN-EL	EN-ET	EN-LT	EN-LV	EN-RO	EN-SL
LEXACC	1:1	96.0	**89.5**	88.9	93.1	95.0	**99.4**	88.5
STACC	1:1	**96.7**	88.0	**92.0**	**96.1**	**96.6**	98.8	**89.5**
LEXACC	2:1	83.4	**83.2**	73.9	81.2	83.8	**95.3**	81.6
STACC	2:1	**89.2**	**83.2**	**79.9**	**86.9**	**88.2**	**95.3**	**82.3**
LEXACC	100:1	16.6	22.7	34.2	45.1	45.1	70.4	24.9
STACC	100:1	**33.7**	**37.3**	**42.5**	**56.0**	**56.2**	**75.7**	**35.3**

Table 4: Best F1 measures on the Accurat evaluation sets

SYSTEM	TEST SETS	EN-BG	EN-DE	EN-ES
LEXACC	1:1	**87.1**	**82.7**	98.2
STACC	1:1	84.9	82.0	**99.7**
LEXACC	100:1	**27.6**	31.0	66.2
STACC	100:1	16.6	**35.8**	**73.3**

Table 5: Best F1 measures on the Wikipedia evaluation sets

	CRF		LEXACC		STACC	
LANGUAGE PAIR	R@90	R@80	R@90	R@80	R@90	R@80
EN-BG	72.0	81.8	**80.4**↑	80.4↑	80.2	**81.6**↑
EN-DE	58.7	68.8	**75.2**	78.7	68.8	**81.8**↑
EN-ES	90.4	93.7	97.0↑	97.0↑	**99.6**↑	**99.6**↑

Table 6: Targeted recall on the Wikipedia evaluation sets

SYSTEM	TEST SETS	ES-EU
LEXACC	1:1	77.2
LEXACC_DF	1:1	80.2
STACC	1:1	**90.9**
LEXACC	EITB_NOISE1	59.2
LEXACC_DF	EITB_NOISE1	62.2
STACC	EITB_NOISE1	**82.8**
LEXACC	EITB_NOISE2	54.5
LEXACC_DF	EITB_NOISE2	57.4
STACC	EITB_NOISE2	**79.5**

Table 7: Best F1 measures on the EITB evaluation sets

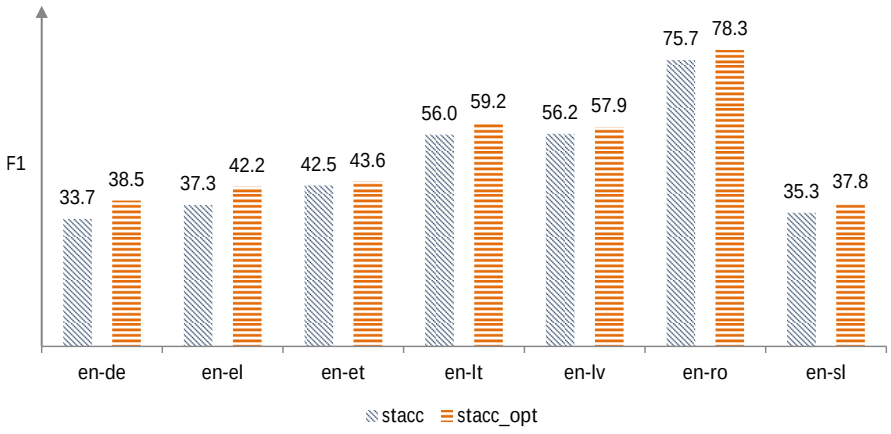


Figure 1: STACC optimisation results on the Accurat 100:1 test sets

cation set for this language pair, is closer to the JRC vocabulary than the original comparable data on which the alignment process would take place in real-world conditions. Although we have not thoroughly tested the impact of this variable, it is possible that those datasets are more confusing for an approach such as STACC, which is based mostly on lexical information extracted from seed parallel data, than for a feature-based approach where some features, like the boolean punctuation-based ones in LEXACC, may compensate for erroneous alignments due to artificial domain vocabulary overlap. Determining if this hypothesis is indeed correct would require further experiments beyond the scope of this paper

To include the CRF approach in the comparison, we used two of the provided measures, namely recall obtained at precisions of 80 and 90 percent on the 1:1 test sets.[14] We report results obtained with the best variant of CRF, namely the model which includes Wikipedia and lexicon features, with intersected results from both directions. Results are reported in Table 6. Although the comparison was limited in this case, results were in favour of LEXACC and STACC on targeted recall measures for the Wikipedia datasets.

Finally, both LEXACC and STACC were compared against the EITB test sets, with results shown in Table 7. For this language pair, STACC performed markedly better with differences of up to 25 points. A likely explanation for these results is the nature of the features that compose the LEXACC model. In particular the features related to alignment obliqueness and number of initial/final aligned words might be detrimental in the case of Basque, which exhibits free word order. Given the poor results obtained with feature weights optimised on the IVAP corpus, we also checked the results using the provided default weights. This resulted in slightly better performance, as shown in the rows named LEXACC_DF in Table 7, though still far from the results achieved with STACC.

4.4 Discussion

Overall, STACC provided the best results across domains and language pairs, in particular for noisier datasets. Additionally, the approach has several advantages over existing methods and systems for comparable segment alignment.

First, it is undoubtedly simpler, as it requires but minimal information to reach optimal results. Lexical tables and simple set expansion operations based on surface properties of the tokens are the only components of the approach, as compared to the more sophisticated feature-based approaches which rely on larger sets of components for which optimal weights need to be computed prior to applying the models.

Secondly, because of its simplicity, STACC is a more portable method, as is it is not necessary to perform any type of adaptation for new domains and language pairs, nor to rely on domain-specific information such as link structure in Wikipedia. In actual practice, portability is an important issue which hinders on the exploitation of comparable corpora. An efficient yet easily deployable method is therefore a welcome addition to the toolset for parallel data extraction.

Finally, STACC results in fewer computational steps when compared to more complex feature-based methods. First, it involves simple binary set intersection and union operations for the computation of similarity, instead of conjoined feature computation on larger component sets. Secondly, the approach relies on tractable set differences for its most computationally expensive operation of longest common prefix matching, compared to matching all tokens against lists of word endings which can be quite large, notably in the case of agglutinative languages.

Although promising, the approach could be further evaluated, and potentially improved, along two main lines.

It might be worth exploring for instance the impact of filtering alignment candidates according to the relative position of sentence pairs in the original source and target documents, a document-level property notably exploited by (Smith et al., 2010). As the STACC approach is featureless, and meant to remain as such in order to maintain its portability and ease of deployment, filtering distant sentence pairs would need to take place prior to the computation of alignment scores. A simple approach compatible with STACC would consist in constraining candidate sets by including sentence position information when performing indexing and candidate querying in a CLIR approach. This would provide an additional evalua-

[14] Note that, for both LEXACC and STACC, in some scenarios even the lowest thresholds gave precisions higher than 90, rendering the comparison moot. We indicate these cases with a ↑ sign next to the highest recall obtained at the closest precision to the arbitrary 80 and 90 precision points.

tion of the accuracy of the approach in scenarios where document-level information is exploitable.

Additionally, given the importance of k-best lexical translations in computing STACC similarity, variations in lexical coverage obtained with different translation tables can be expected to impact alignment accuracy. Although mining comparable corpora usually requires the use of seed translation knowledge extracted from a domain that differs from the one being mined, default tables with wide lexical coverage can be built from existing parallel corpora in different domains. Thus, improvements might be obtained with larger and more diverse tables than the ones used in the experiments reported here, which were based on translations extracted from a single domain. A precise assessment of the evolution of alignment accuracy given variations in lexical translation coverage is left for future research.

5 Alignment optimisation

As previously mentioned, for both LEXACC and STACC, alignments were computed for every source sentence against candidate translations retrieved by Lucene and all cases where a given target sentence has more than one source alignment were left as is.

Although this methodology enabled a fair comparison between the two systems, it evidently impacts alignment accuracy. One simple optimisation is to retain only the best overall source-target alignments, discarding all alignments established between a given source sentence and a target sentence if the latter is linked to better scoring source sentences.

The net effect of this procedure is the promotion of better alignments, as some correct alignments would not be hidden anymore by other better scoring shared alignments. This is most likely to occur with source-target pairs that are close variants of each other, with close similarity scores.

We applied this simple optimisation to the Accurat test sets and observed improvements across the board, as shown in Figure 1. Depending on actual usage, this optimised version of STACC alignment can constitute the best alternative for the extraction of parallel sentences from comparable corpora.

6 Conclusions

We described a simple approach to comparable sentence alignment, termed STACC, which is based on automatically extracted seed lexical translations, the Jaccard similarity coefficient, and simple set expansion operations that target named entities, numbers, and morphological variation using longest common prefixes. Building on fairly standard components for the computation of similarity, this method is shown to perform better than current alternatives.

The approach was evaluated on a large range of datasets from various domains for ten language pairs, giving the best results overall when compared to sophisticated state-of-the-art methods. STACC also performed better than competing approaches on noisier corpora, showing promises for the exploitation of the typically noisy data found when mining comparable corpora.

STACC is a highly portable method which requires no adaptation for its application to new domains and language pairs. It thus allows for the fast deployment of a crucial component in comparable corpora alignment, which opens the path for an increase in the amount of such corpora that can be exploited in the future.

Acknowledgments

This work was partially funded by the Spanish Ministry of Economy and Competitiveness and the Department of Economic Development and Competitiveness of the Basque Government through the AdapTA (RTC-2015-3627-7), PLATA (IG-2014/00037) and TRADIN (IG-2015/0000347) projects. We would like to thank MondragonLingua Translation & Communication as coordinator of these projects and the three anonymous reviewers for their helpful feedback and suggestions.

References

Sadaf Abdul-Rauf and Holger Schwenk. 2009. On the use of comparable corpora to improve SMT performance. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 16–23, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. 1990. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85.

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

Pascale Fung and Percy Cheung. 2004. Mining Very Non-Parallel Corpora: Parallel Sentence and Lexicon Extraction via Bootstrapping and E.M. In *Proceedings of Empirical Methods in Natural Language Processing*, pages 57–63.

Paul Jaccard. 1901. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241 – 272.

Philipp Koehn. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the 10th Machine Translation Summit*, pages 79–86.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Robert C. Moore. 2002. Fast and accurate sentence alignment of bilingual corpora. In *Proceedings of the 5th Conference of the Association for Machine Translation in the Americas on Machine Translation: From Research to Real Users*, AMTA '02, pages 135–144, London, UK, UK. Springer-Verlag.

Dragos Stefan Munteanu and Daniel Marcu. 2002. Processing Comparable Corpora With Bilingual Suffix Trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 289–295. Association for Computational Linguistics.

Dragos Stefan Munteanu and Daniel Marcu. 2005. Improving machine translation performance by exploiting non-parallel corpora. *Computational Linguistics*, 31(4):477–504.

Makoto Nagao. 1984. A Framework for a Mechanical Translation Between Japanese and English by Analogy Principle. In *Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, pages 173–180, New York, NY, USA. Elsevier North-Holland, Inc.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318. Association for Computational Linguistics.

Monica Lestari Paramita, David Guthrie, Evangelos Kanoulas, Rob Gaizauskas, Paul Clough, and Mark Sanderson. 2013. Methods for collection and evaluation of comparable documents. In *Building and Using Comparable Corpora*, pages 93–112. Springer.

Mohammad Taher Pilehvar, David Jurgens, and Roberto Navigli. 2013. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st meeting of the Association for Computational Linguistics*, pages 1341–1351. The Association for Computational Linguistics.

Ruhi Sarikaya, Sameer Maskey, R Zhang, Ea-Ee Jan, D Wang, Bhuvana Ramabhadran, and Salim Roukos. 2009. Iterative sentence-pair extraction from quasi-parallel corpora for machine translation. In *Proceedings of InterSpeech*, pages 432–435.

Inguna Skadiņa, Ahmet Aker, Nikos Mastropavlos, Fangzhong Su, Dan Tufis, Mateja Verlic, Andrejs Vasiļjevs, Bogdan Babych, Paul Clough, Robert Gaizauskas, et al. 2012. Collecting and using comparable corpora for statistical machine translation. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*.

Jason R. Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 403–411, Stroudsburg, PA, USA. Association for Computational Linguistics.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Dan Ştefănescu, Radu Ion, and Sabine Hunsicker. 2012. Hybrid parallel sentence mining from comparable corpora. In *Proceedings of the 16th Conference of the European Association for Machine Translation*, pages 137–144.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the 8th Language Resources and Evaluation Conference*, pages 2214–2218.

Bing Zhao and Stephan Vogel. 2002. Adaptive parallel sentences mining from web bilingual news collection. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 745–748. IEEE.

Jointly Learning to Embed and Predict with Multiple Languages

Daniel C. Ferreira[*] **André F. T. Martins**[†*♯] **Mariana S. C. Almeida**[*♯]
[*]Priberam Labs, Alameda D. Afonso Henriques, 41, 2º, 1000-123 Lisboa, Portugal
[†]Unbabel Lda, Rua Visconde de Santarém, 67-B, 1000-286 Lisboa, Portugal
[♯]Instituto de Telecomunicações, Instituto Superior Técnico, 1049-001 Lisboa, Portugal
{dcf,mla}@priberam.pt, {andre.martins}@unbabel.com

Abstract

We propose a joint formulation for learning task-specific cross-lingual word embeddings, along with classifiers for that task. Unlike prior work, which first learns the embeddings from parallel data and then plugs them in a supervised learning problem, our approach is one-shot: a single optimization problem combines a co-regularizer for the multilingual embeddings with a task-specific loss. We present theoretical results showing the limitation of Euclidean co-regularizers to increase the embedding dimension, a limitation which does not exist for other co-regularizers (such as the ℓ_1-distance). Despite its simplicity, our method achieves state-of-the-art accuracies on the RCV1/RCV2 dataset when transferring from English to German, with training times below 1 minute. On the TED Corpus, we obtain the highest reported scores on 10 out of 11 languages.

1 Introduction

Distributed representations of text (embeddings) have been the target of much research in natural language processing (Collobert and Weston, 2008; Mikolov et al., 2013; Pennington et al., 2014; Levy et al., 2015). Word embeddings partially capture semantic and syntactic properties of text in the form of dense real vectors, making them apt for a wide variety of tasks, such as language modeling (Bengio et al., 2003), sentence tagging (Turian et al., 2010; Collobert et al., 2011), sentiment analysis (Socher et al., 2011), parsing (Chen and Manning, 2014), and machine translation (Zou et al., 2013).

At the same time, there has been a consistent progress in devising "universal" multilingual models via cross-lingual transfer techniques of various kinds (Hwa et al., 2005; Zeman and Resnik, 2008; McDonald et al., 2011; Ganchev and Das, 2013; Martins, 2015). This line of research seeks ways of using data from resource-rich languages to solve tasks in resource-poor languages. Given the difficulty of handcrafting language-independent features, it is highly appealing to obtain rich, delexicalized, multilingual representations embedded in a shared space.

A string of work started with Klementiev et al. (2012) on learning bilingual embeddings for text classification. Hermann and Blunsom (2014) proposed a noise-contrastive objective to push the embeddings of parallel sentences to be close in space. A bilingual auto-encoder was proposed by Chandar et al. (2014), while Faruqui and Dyer (2014) applied canonical correlation analysis to parallel data to improve monolingual embeddings. Other works optimize a sum of monolingual and cross-lingual terms (Gouws et al., 2015; Soyer et al., 2015), or introduce bilingual variants of skip-gram (Luong et al., 2015; Coulmance et al., 2015). Recently, Pham et al. (2015) extended the non-compositional paragraph vectors of Le and Mikolov (2014) to a bilingual setting, achieving a new state of the art at the cost of more expensive (and non-deterministic) prediction.

In this paper, we propose an alternative joint formulation that learns embeddings suited to a particular task, together with the corresponding classifier for that task. We do this by minimizing a combination of a supervised loss function and a multilingual regularization term. Our approach leads to a convex optimization problem and makes a bridge between classical co-regularization approaches for semi-supervised learning (Sindhwani et al., 2005; Altun et al., 2005; Ganchev et al.,

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2019–2028,
Berlin, Germany, August 7-12, 2016.

2008) and modern representation learning. In addition, we show that Euclidean co-regularizers have serious limitations to learn rich embeddings, when the number of task labels is small. We establish this by proving that the resulting embedding matrices have their rank upper bounded by the number of labels. This limitation does not exist for other regularizers (convex or not), such as the ℓ_1-distance and noise-contrastive distances.

Our experiments in the RCV1/RCV2 dataset yield state-of-the-art accuracy (92.7%) with this simple convex formulation, when transferring from English to German, without the need of negative sampling, extra monolingual data, or non-additive representations. For the reverse direction, our best number (79.3%), while far behind the recent `para_doc` approach (Pham et al., 2015), is on par with current compositional methods.

On the TED corpus, we obtained general purpose multilingual embeddings for 11 target languages, by considering the (auxiliary) task of reconstructing pre-trained English word vectors. The resulting embeddings led to cross-lingual multi-label classifiers that achieved the highest reported scores on 10 out of these 11 languages.[1]

2 Cross-Lingual Text Classification

We consider a **cross-lingual classification** framework, where a classifier is trained on a dataset from a source language (such as English) and applied to a target language (such as German). Later, we generalize this setting to multiple target languages and to other tasks besides classification.

The following data are assumed available:

1. A **labeled dataset** $\mathcal{D}_l := \{\langle \boldsymbol{x}^{(m)}, y^{(m)}\rangle\}_{m=1}^{M}$, consisting of text documents $\boldsymbol{x}$ in the source language categorized with a label $y \in \{1, \ldots, L\}$.
2. An **unlabeled parallel corpus** $\mathcal{D}_u := \{(\boldsymbol{s}^{(n)}, \boldsymbol{t}^{(n)})\}_{n=1}^{N}$, containing sentences $\boldsymbol{s}$ in the source language paired with their translations $\boldsymbol{t}$ in the target language (but no information about their categories).

Let V_{S} and V_{T} be the vocabulary size of the source and target languages, respectively. Throughout, we represent sentences $\boldsymbol{s} \in \mathbb{R}^{V_{\text{S}}}$ and $\boldsymbol{t} \in \mathbb{R}^{V_{\text{T}}}$ as vectors of word counts, and documents $\boldsymbol{x}$ as an average of sentence vectors. We assume that the unlabeled sentences largely outnumber the labeled documents, $N \gg M$, and that the number of labels L is relatively small. The goal is to use the data above to learn a classifier $h : \mathbb{R}^{V_{\text{T}}} \to \{1, \ldots, L\}$ for the target language.

This problem is usually tackled with a two-stage approach: in the first step, bilingual word embeddings $\boldsymbol{P} \in \mathbb{R}^{V_{\text{S}} \times K}$ and $\boldsymbol{Q} \in \mathbb{R}^{V_{\text{T}} \times K}$ are learned from $\mathcal{D}_u$, where each row of these matrices contains a Kth dimensional word representation in a shared vector space. In the second step, a standard classifier is trained on $\mathcal{D}_l$, using the source embeddings $\boldsymbol{P} \in \mathbb{R}^{V_{\text{S}} \times K}$. Since the embeddings are in a shared space, the trained model can be applied directly to classify documents in the target language. We describe next these two steps in more detail. We assume throughout an additive representation for sentences and documents (denoted ADD by Hermann and Blunsom (2014)). These representations can be expressed algebraically as $\boldsymbol{P}^\top \boldsymbol{x}, \boldsymbol{P}^\top \boldsymbol{s}, \boldsymbol{Q}^\top \boldsymbol{t} \in \mathbb{R}^K$, respectively.

Step 1: Learning the Embeddings. The cross-lingual embeddings $\boldsymbol{P}$ and $\boldsymbol{Q}$ are trained so that the representations of paired sentences $(\boldsymbol{s}, \boldsymbol{t}) \in \mathcal{D}_u$ have a small (squared) Euclidean distance

$$d_{\ell_2}(\boldsymbol{s}, \boldsymbol{t}) = \frac{1}{2}\|\boldsymbol{P}^\top \boldsymbol{s} - \boldsymbol{Q}^\top \boldsymbol{t}\|^2. \quad (1)$$

Since a direct minimization of Eq. 1 leads to a degenerate solution ($\boldsymbol{P} = \boldsymbol{0}, \boldsymbol{Q} = \boldsymbol{0}$), Hermann and Blunsom (2014) use instead a noise-contrastive large-margin distance obtained via negative sampling,

$$d_{\text{ns}}(\boldsymbol{s}, \boldsymbol{t}, \boldsymbol{n}) = [m + d_{\ell_2}(\boldsymbol{s}, \boldsymbol{t}) - d_{\ell_2}(\boldsymbol{s}, \boldsymbol{n})]_+, \quad (2)$$

where $\boldsymbol{n}$ is a random (unpaired) target sentence, m is a "margin" parameter, and $[x]_+ := \max\{0, x\}$. Letting J be the number of negative examples in each sample, they arrive at the following objective function to be minimized:

$$\mathcal{R}_{\text{ns}}(\boldsymbol{P}, \boldsymbol{Q}) := \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{J} d_{\text{ns}}(\boldsymbol{s}^{(n)}, \boldsymbol{t}^{(n)}, \boldsymbol{n}^{(n,j)}). \quad (3)$$

This minimization can be carried out efficiently with gradient-based methods, such as stochastic gradient descent or AdaGrad (Duchi et al., 2011). Note however that the objective function in Eq. 3 is not convex. Therefore, one may land at different local minima, depending on the initialization.

[1] We provide the trained embeddings at `http://www.cs.cmu.edu/~afm/projects/multilingual_embeddings.html`.

Step 2: Training the Classifier. Once we have the bilingual embeddings $\boldsymbol{P}$ and $\boldsymbol{Q}$, we can compute the representation $\boldsymbol{P}^\top \boldsymbol{x} \in \mathbb{R}^K$ of each document $\boldsymbol{x}$ in the labeled dataset $\mathcal{D}_l$. Let $\boldsymbol{V} \in \mathbb{R}^{K \times L}$ be a matrix of parameters (weights), with one column $\boldsymbol{v}_y$ per label. A linear model is used to make predictions, according to

$$\begin{aligned} \widehat{y} &= \text{argmax}_{y \in \{1,\ldots,L\}} \boldsymbol{v}_y^\top \boldsymbol{P}^\top \boldsymbol{x} \\ &= \text{argmax}_{y \in \{1,\ldots,L\}} \boldsymbol{w}_y^\top \boldsymbol{x}, \end{aligned} \quad (4)$$

where $\boldsymbol{w}_y$ is a column of the matrix $\boldsymbol{W} := \boldsymbol{PV} \in \mathbb{R}^{V_\text{S} \times L}$. In prior work, the perceptron algorithm was used to learn the weights $\boldsymbol{V}$ from the labeled examples in $\mathcal{D}_l$ (Klementiev et al., 2012; Hermann and Blunsom, 2014). Note that, at test time, it is not necessary to store the full embeddings: if $L \ll K$, we may simply precompute $\boldsymbol{W} := \boldsymbol{PV}$ (one weight per word and label) if the input is in the source language—or $\boldsymbol{QV}$, if the input is in the target language—and treat this as a regular bag-of-words linear model.

3 Jointly Learning to Embed and Classify

Instead of a two-stage approach, we propose to learn the bilingual embeddings and the classifier *jointly* on $\mathcal{D}_l \cup \mathcal{D}_u$, as described next.

Our formulation optimizes a combination of a **co-regularization function** $\mathcal{R}$, whose goal is to push the embeddings of paired sentences in $\mathcal{D}_u$ to stay close, and a **loss function** $\mathcal{L}$, which fits the model to the labeled data in $\mathcal{D}_l$.

The simplest choice for $\mathcal{R}$ is a simple Euclidean co-regularization function:

$$\begin{aligned} \mathcal{R}_{\ell_2}(\boldsymbol{P}, \boldsymbol{Q}) &= \frac{1}{N} \sum_{n=1}^{N} d_{\ell_2}(\boldsymbol{s}^{(n)}, \boldsymbol{t}^{(n)}) \\ &= \frac{1}{2N} \sum_{n=1}^{N} \|\boldsymbol{P}^\top \boldsymbol{s}^{(n)} - \boldsymbol{Q}^\top \boldsymbol{t}^{(n)}\|^2. \end{aligned} \quad (5)$$

An alternative is the ℓ_1-distance:

$$\mathcal{R}_{\ell_1}(\boldsymbol{P}, \boldsymbol{Q}) = \frac{1}{N} \sum_{n=1}^{N} \|\boldsymbol{P}^\top \boldsymbol{s}^{(n)} - \boldsymbol{Q}^\top \boldsymbol{t}^{(n)}\|_1. \quad (6)$$

One possible advantage of $\mathcal{R}_{\ell_1}(\boldsymbol{P}, \boldsymbol{Q})$ over $\mathcal{R}_{\ell_2}(\boldsymbol{P}, \boldsymbol{Q})$ is that the ℓ_1-distance is more robust to outliers, hence it is less sensitive to differences in the parallel sentences. Note that both functions in Eqs. 5–6 are jointly convex on $\boldsymbol{P}$ and $\boldsymbol{Q}$, unlike the one in Eq. 3. They are also simpler and do not require negative sampling. While these functions have a degenerate behavior in isolation (since they are both minimized by $\boldsymbol{P} = \boldsymbol{0}$ and $\boldsymbol{Q} = \boldsymbol{0}$), we will see that they become useful when plugged into a joint optimization framework.

The next step is to define the loss function $\mathcal{L}$ to leverage the labeled data in $\mathcal{D}_l$. We consider a log-linear model $P(y \mid \boldsymbol{x}; \boldsymbol{W}) \propto \exp(\boldsymbol{w}_y^\top \boldsymbol{x})$, which leads to the following logistic loss function:

$$\mathcal{L}_{LL}(\boldsymbol{W}) = -\frac{1}{M} \sum_{m=1}^{M} \log P(y^{(m)} \mid \boldsymbol{x}^{(m)}; \boldsymbol{W}). \quad (7)$$

We impose that $\boldsymbol{W}$ is of the form $\boldsymbol{W} = \boldsymbol{PV}$ for a fixed $\boldsymbol{V} \in \mathbb{R}^{K \times L}$, whose choice we discuss below.

Putting the pieces together and adding some extra regularization terms, we formulate our joint objective function as follows:

$$\begin{aligned} \mathcal{F}(\boldsymbol{P}, \boldsymbol{Q}) &= \mu \mathcal{R}(\boldsymbol{P}, \boldsymbol{Q}) + \mathcal{L}(\boldsymbol{PV}) \\ &\quad + \frac{\mu_\text{S}}{2} \|\boldsymbol{P}\|_\text{F}^2 + \frac{\mu_\text{T}}{2} \|\boldsymbol{Q}\|_\text{F}^2, \end{aligned} \quad (8)$$

where $\mu, \mu_\text{S}, \mu_\text{T} \geq 0$ are regularization constants. By minimizing a combination of $\mathcal{L}(\boldsymbol{PV})$ and $\mathcal{R}(\boldsymbol{P}, \boldsymbol{Q})$, we expect to obtain embeddings $\boldsymbol{Q}^*$ that lead to an accurate classifier h for the target language. Note that $\boldsymbol{P} = \boldsymbol{0}$ and $\boldsymbol{Q} = \boldsymbol{0}$ is no longer a solution, due to the presence of the loss term $\mathcal{L}(\boldsymbol{PV})$ in the objective.

Choice of $\boldsymbol{V}$. In Eq. 8, we chose to keep $\boldsymbol{V}$ fixed rather than optimize it. The rationale is that there are many more degrees of freedom in the embedding matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ than in $\boldsymbol{V}$ (concretely, $\mathcal{O}(K(V_\text{S} + V_\text{T}))$ versus $\mathcal{O}(KL)$, where we are assuming a small number of labels, $L \ll V_\text{S} + V_\text{T}$). Our assumption is that we have enough degrees of freedom to obtain an accurate model, regardless of the choice of $\boldsymbol{V}$. These claims will be backed in §4 by a more rigorous theoretical result. Keeping $\boldsymbol{V}$ fixed has another important advantage: it allows to minimize $\mathcal{F}$ with respect to $\boldsymbol{P}$ and $\boldsymbol{Q}$ only, which makes it a convex optimization problem if we choose $\mathcal{R}$ and $\mathcal{L}$ to be both convex—*e.g.*, setting $\mathcal{R} \in \{\mathcal{R}_{\ell_2}, \mathcal{R}_{\ell_1}\}$ and $\mathcal{L} := \mathcal{L}_{LL}$.

Relation to Multi-View Learning. An interesting particular case of this formulation arises if $K = L$ and $\boldsymbol{V} = \boldsymbol{I}_L$ (the identity matrix). In that case, we have $\boldsymbol{W} = \boldsymbol{P}$ and the embedding matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ are in fact weights for every pair of word and label, as in standard bag-of-word

models. In this case, we may interpret the co-regularizer $\mathcal{R}(\boldsymbol{P}, \boldsymbol{Q})$ in Eq. 8 as a term that pushes the *label scores* of paired sentences $\boldsymbol{P}^\top \boldsymbol{s}^{(n)}$ and $\boldsymbol{Q}^\top \boldsymbol{t}^{(n)}$ to be similar, while the source-based log-linear model is fit via $\mathcal{L}(\boldsymbol{W})$. The same idea underlies various semi-supervised co-regularization methods that seek agreement between multiple views (Sindhwani et al., 2005; Altun et al., 2005; Ganchev et al., 2008). In fact, we may regard the joint optimization in Eq. 8 as a generalization of those methods, making a bridge between those methods and representation learning.

Multilingual Embeddings. It is straightforward to extend the framework herein presented to the case where there are *multiple* target languages (say R of them), and we want to learn one embedding matrix for each, $\{\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_R\}$. The simplest way is to consider a sum of pairwise co-regularizers,

$$\mathcal{R}'(\boldsymbol{P}, \{\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_R\}) := \sum_{r=1}^{R} \mathcal{R}(\boldsymbol{P}, \boldsymbol{Q}_r). \quad (9)$$

If $\mathcal{R}$ is additive over the parallel sentences (which is the case for $\mathcal{R}_{\ell_2}$, $\mathcal{R}_{\ell_1}$ and $\mathcal{R}_{\text{ns}}$), then this procedure is equivalent to concatenating all the parallel sentences (regardless of the target language) and adding a language suffix to the words to distinguish them. This reduces directly to a problem in the same form as Eq. 8.

Pre-Trained Source Embeddings. In practice, it is often the case that pre-trained embeddings for the source language are already available (let $\bar{\boldsymbol{P}}$ be the available embedding matrix). It would be foolish not to exploit those resources. In this scenario, the goal is to use $\bar{\boldsymbol{P}}$ and the dataset $\mathcal{D}_u$ to obtain "good" embeddings for the target languages (possibly tweaking the source embeddings too, $\boldsymbol{P} \approx \bar{\boldsymbol{P}}$). Our joint formulation in Eq. 8 can also be used to address this problem. It suffices to set $K = L$ and $\boldsymbol{V} = \boldsymbol{I}_L$ (as in the multi-view learning case discussed above) and to define an auxiliary task that pushes $\boldsymbol{P}$ and $\bar{\boldsymbol{P}}$ to be similar. The simplest way is to use a reconstruction loss:

$$\mathcal{L}_{\ell_2}(\boldsymbol{P}, \bar{\boldsymbol{P}}) := \frac{1}{2}\|\boldsymbol{P} - \bar{\boldsymbol{P}}\|_{\text{F}}^2. \quad (10)$$

The resulting optimization problem has resemblances with the retrofitting approach of Faruqui et al. (2015), except that the goal here is to extend the embeddings to other languages, instead of pushing monolingual embeddings to agree with a semantic lexicon. We will present some experiments in §5.2 using this framework.

4 Limitations of the Euclidean Co-Regularizer

One may wonder how much the embedding dimension K influences the learned classifier. The next proposition shows the (surprising) result that, with the formulation in Eq. 8 with $\mathcal{R} = \mathcal{R}_{\ell_2}$, it makes absolutely no difference to increase K past the number of labels L. Below, $\boldsymbol{T} \in \mathbb{R}^{V_{\text{T}} \times N}$ denotes the matrix with columns $\boldsymbol{t}^{(1)}, \ldots, \boldsymbol{t}^{(N)}$.

Proposition 1. *Let $\mathcal{R} = \mathcal{R}_{\ell_2}$ and assume $\boldsymbol{T}$ has full row rank.*[2] *Then, for any choice of $\boldsymbol{V} \in \mathbb{R}^{K \times L}$, possibly with $K > L$, the following holds:*

1. *There is an alternative, low-dimensional, $\boldsymbol{V}' \in \mathbb{R}^{K' \times L}$ with $K' \leq L$ such that the classifier obtained (for both languages) by optimizing Eq. 8 using $\boldsymbol{V}'$ is the same as if using $\boldsymbol{V}$.*[3]
2. *This classifier depends on $\boldsymbol{V}$ only via the L-by-L matrix $\boldsymbol{V}^\top \boldsymbol{V}$.*
3. *If $\boldsymbol{P}^*, \boldsymbol{Q}^*$ are the optimal embeddings obtained with $\boldsymbol{V}$, then we always have $\text{rank}(\boldsymbol{P}^*) \leq L$ and $\text{rank}(\boldsymbol{Q}^*) \leq L$ regardless of K.*

Proof. See App. A.1 in the supplemental material. □

Let us reflect for a moment on the practical impact of Prop. 1. This result shows the limitation of the Euclidean co-regularizer $\mathcal{R}_{\ell_2}$ in a very concrete manner: when $\mathcal{R} = \mathcal{R}_{\ell_2}$, we only need to consider representations of dimension $K \leq L$.

Note also that a corollary of Prop. 1 arises when $\boldsymbol{V}^\top \boldsymbol{V} = \boldsymbol{I}_L$, i.e., when $\boldsymbol{V}$ is chosen to have orthonormal columns (a sensible choice, since it corresponds to seeking embeddings that leave the label weights "uncorrelated"). Then, the second statement of Prop. 1 tells us that the resulting classifier will be the same as if we had simply set $\boldsymbol{V} = \boldsymbol{I}_L$ (the particular case discussed in §3). We will see in §5.1 that, despite this limitation, this classifier is actually a very strong baseline. Of course, if the number of labels L is large enough,

[2] This assumption is not too restrictive: it holds if $N \geq V_{\text{T}}$ and if no target sentence can be written as a linear combination of the others (this can be accomplished if we remove redundant parallel sentences).

[3] Let $\boldsymbol{P}^*, \boldsymbol{Q}^*$ and $\boldsymbol{P}'^*, \boldsymbol{Q}'^*$ be the optimal embeddings obtained with $\boldsymbol{V}$ and $\boldsymbol{V}'$, respectively. Since we are working with linear classifiers, the two classifiers are the same in the sense that $\boldsymbol{P}^*\boldsymbol{V} = \boldsymbol{P}'^*\boldsymbol{V}'$ and $\boldsymbol{Q}^*\boldsymbol{V} = \boldsymbol{Q}'^*\boldsymbol{V}'$.

this limitation might not be a reason for concern.[4] An instance will be presented in §5.2, where we will see that the Euclidean co-regularizer excels.

Finally, one might wonder whether Prop. 1 applies only to the (Euclidean) ℓ_2 norm or if it holds for arbitrary regularizers. In fact, we show in App. A.2 that this limitation applies more generally to Mahalanobis-Frobenius norms, which are essentially Euclidean norms after a linear transformation of the vector space. However, it turns out that for general norms such limitation does *not* exist, as shown below.

Proposition 2. *If $\mathcal{R} = \mathcal{R}_{\ell_1}$ in Eq. 8, then the analogous to Proposition 1 does not hold. It also does not hold for the ℓ_∞-norm and the ℓ_0-"norm."*

Proof. See App. A.3 in the supplemental material. □

This result suggests that, for other regularizers $\mathcal{R} \neq \mathcal{R}_{\ell_2}$, we may eventually obtain better classifiers by increasing K past L. As such, in the next section, we experiment with $\mathcal{R} \in \{\mathcal{R}_{\ell_2}, \mathcal{R}_{\ell_1}, \mathcal{R}_{\text{ns}}\}$, where $\mathcal{R}_{\text{ns}}$ is the (non-convex) noise-contrastive regularizer of Eq. 3.

5 Experiments

We report results on two experiments: one on cross-lingual classification on the Reuters RCV1/RCV2 dataset, and another on multi-label classification with multilingual embeddings on the TED Corpus.[5]

5.1 Reuters RCV1/RCV2

We evaluate our framework on the cross-lingual document classification task introduced by Klementiev et al. (2012). Following prior work, our dataset $\mathcal{D}_u$ consists of 500,000 parallel sentences from the Europarl v7 English-German corpus (Koehn, 2005); and our labeled dataset $\mathcal{D}_l$ consists of English and German documents from the RCV1/RCV2 corpora (Lewis et al., 2004), each categorized with one out of $L = 4$ labels. We used the same split as Klementiev et al. (2012): 1,000 documents for training, of which 200 are held out as validation data, and 5,000 for testing.

Note that, in this dataset, we are classifying documents based on their bag-of-word representations, and learning word embeddings by bringing the bag-of-word representations of parallel sentences to be close together. In this sense, we are bringing together these multiple levels of representations (document, sentence and word).

We experimented with the joint formulation in Eq. 8, with $\mathcal{L} := \mathcal{L}_{LL}$ and $\mathcal{R} \in \{\mathcal{R}_{\ell_2}, \mathcal{R}_{\ell_1}, \mathcal{R}_{\text{ns}}\}$. We optimized with AdaGrad (Duchi et al., 2011) with a stepsize of 1.0, using mini-batches of 100 Reuters RCV1/RCV2 documents and 50,000 Europarl v7 parallel sentences. We found no need to run more than 100 iterations, with most of our runs converging under 50. Our vocabulary has 69,714 and 175,650 words for English and German, respectively, when training on the English portion of the Reuters RCV1/RCV2 corpus, and 61,120 and 183,888 words for English and German, when training in the German portion of the corpus. This difference is due to the inclusion of words in the training data into the vocabulary. We do not remove any words from the vocabulary, for simplicity. We used the validation set to tune the hyperparameters $\{\mu, \mu_{\text{S}}, \mu_{\text{T}}\}$ and to choose the iteration number. When using $K = L$, we chose $\boldsymbol{V} = \boldsymbol{I}_L$; otherwise, we chose $\boldsymbol{V}$ randomly, sampling its entries from a Gaussian $\mathcal{N}(0, 0.1)$.

Table 1 shows the results. We include for comparison the most competitive systems published to date. The first thing to note is that our joint system with Euclidean co-regularization performs very well for this task, despite the theoretical limitations shown in §4. Although its embedding size is only $K = 4$ (one dimension per label), it outperformed all the two-stage systems trained on the same data, in both directions.

For the EN→DE direction, our joint system with ℓ_1 co-regularization achieved state-of-the-art results (92.7%), matching two-stage systems that use extra monolingual data, negative sampling, or non-additive document representations. It is conceivable that the better results of $\mathcal{R}_{\ell_1}$ over $\mathcal{R}_{\ell_2}$ come from its higher robustness to differences in the parallel sentences.

For the DE→EN direction, our best result (79%) was obtained with the noise-contrastive co-regularizer, which outperformed all systems except `para_doc` (Pham et al., 2015). While the accuracy of `para_doc` is quite impressive, note that it requires 500-dimensional embeddings

[4] For regression tasks (such as the one presented in the last paragraph of 3), instead of the "number of labels," L should be regarded as the number of output variables to regress.

[5] Our code is available at `https://github.com/dcferreira/multilingual-joint-embeddings`.

		K	EN→DE	DE→EN
I-Matrix	[KTB12]	40	77.6	71.1
ADD	[HB14]	40	83.7	71.4
ADD	[HB14]	128	86.4	74.7
BI	[HB14]	40	83.4	69.2
BI	[HB14]	128	86.1	79.0
BilBOWA	[GBC15]	40	86.5	75.0
Binclusion	[SSA15]	40	86.8	76.7
Bincl.+RCV	[SSA15] (‡)	40	**92.7**	84.4
CLC-WA	[SLLS15] (†)	40	91.3	77.2
`para_sum`	[PLM15] (†)	100	90.6	78.8
`para_doc`	[PLM15] (†)	500	**92.7**	**91.5**
Joint, $\mathcal{R}_{\ell_2}$		4	91.2	78.2
Joint, $\mathcal{R}_{\ell_1}$		4	**92.7**	76.0
Joint, $\mathcal{R}_{\ell_1}$		40	**92.7**	76.2
Joint, $\mathcal{R}_{\text{ns}}$		4	91.2	76.8
Joint, $\mathcal{R}_{\text{ns}}$		40	91.4	79.3

Table 1: Accuracies in the RCV1/RCV2 dataset. Shown for comparison are Klementiev et al. (2012) [KTB12], Hermann and Blunsom (2014) [HB14], Gouws et al. (2015) [GBC15], Soyer et al. (2015) [SSA15], Shi et al. (2015) [SLLS15], and Pham et al. (2015) [PLM15]. Systems marked with (†) used the full 1.8M parallel sentences in Europarl. The one with (‡) used additional target monolingual data from RCV1/RCV2. The bottom rows refer to our joint method, with Euclidean (ℓ_2), ℓ_1, and noise-contrastive co-regularization.

(hence many more parameters), was trained on more parallel sentences, and requires more expensive (and non-deterministic) computation at test time to compute a document's embedding. Our method has the advantage of being simple and very fast to train: it took less than 1 minute to train the joint-$\mathcal{R}_{\ell_1}$ system for EN→DE, using a single core on an Intel Xeon @2.5 GHz. This can be compared with Klementiev et al. (2012), who took 10 days on a single core, or Coulmance et al. (2015), who took 10 minutes with 6 cores.[6]

Although our theoretical results suggest that increasing K when using the ℓ_1 norm may increase the expressiveness of our embeddings, our results do not support this claim (the improvements in DE→EN from $K = 4$ to $K = 40$ were tiny). However, it led to a gain of 2.5 points when using negative sampling. For $K = 40$, this system is much more accurate than Hermann and Blunsom (2014), which confirms that learning the embeddings together with the task is highly beneficial.

[6]Coulmance et al. (2015) reports accuracies of 87.8% (EN→DE) and 78.7% (DE→EN), when using 10,000 training documents from the RCV1/RCV2 corpora.

5.2 TED Corpus

To assess the ability of our framework to handle multiple target languages, we ran a second set of experiments on the TED corpus (Cettolo et al., 2012), using the training and test partitions created by Hermann and Blunsom (2014), downloaded from `http://www.clg.ox.ac.uk/tedcorpus`. The corpus contains English transcriptions and multilingual, sentence-aligned translations of talks from the TED conference in 12 different languages, with 12,078 parallel documents in the training partition (totalling 1,641,985 parallel sentences). Following their prior work, we used this corpus both as parallel data ($\mathcal{D}_u$) and as the task dataset ($\mathcal{D}_l$). There are $L = 15$ labels and documents can have multiple labels.

We experimented with two different strategies:

- A one-stage system (*Joint*), which jointly trains the multilingual embeddings and the multi-label classifier (similarly as in §5.1). To cope with multiple target languages, we used a sum of pairwise co-regularizers as described in Eq. 9. For classification, we use multinomial logistic regression, where we select those labels with a posterior probability above 0.18 (tuned on vali-

dation data).

- A two-stage approach (*Joint w/ Aux*), where we first obtain multilingual embeddings by applying our framework with an auxiliary task with pre-trained English embeddings (as described in Eq. 10 and in the last paragraph of §3), and then use the resulting multilingual representations to train the multi-label classifier. We address this multi-label classification problem with independent binary logistic regressors (one per label), trained by running 100 iterations of L-BFGS (Liu and Nocedal, 1989). At test time, we select those labels whose posterior probability are above 0.5.

For the *Joint w/ Aux* strategy, we used the 300-dimensional GloVe-840B vectors (Pennington et al., 2014), downloaded from `http://nlp.stanford.edu/projects/glove/`.

Table 2 shows the results for cross-lingual classification, where we use English as source and each of the other 11 languages as target. We compare our two strategies above with the strong Machine Translation (MT) baseline used by Hermann and Blunsom (2014) (which translates the input documents to English with a state-of-the-art MT system) and with their two strongest systems, which build document-level representations from embeddings trained bilingually or multilingually (called DOC/ADD *single* and DOC/ADD *joint*, respectively).[7] Overall, our *Joint* system with ℓ_2 regularization outperforms both Hermann and Blunsom (2014)'s systems (but not the MT baseline) for 8 out of 11 languages, performing generally better than our ℓ_1-regularized system. However, the clear winner is our ℓ_2-regularized *Joint w/ Aux* system, which wins over all systems (including the MT baseline) by a substantial margin, for all languages. This shows that pre-trained source embeddings can be extremely helpful in bootstrapping multilingual ones.[8] On the other hand, the performance of the *Joint w/ Aux* system with ℓ_1 regularization is rather disappointing. Note that the limitations of $\mathcal{R}_{\ell_2}$ shown in §4 are not a concern here, since the auxiliary task has $L = 300$ dimensions (the dimension of the pre-trained embeddings). A small sample of the multilingual embeddings produced by the winner system is shown in Table 4.

Finally, we did a last experiment in which we use our multilingual embeddings obtained with *Joint w/ Aux* to train monolingual systems for each language. This time, we compare with a bag-of-words naïve Bayes system (reported by Hermann and Blunsom (2014)), a system trained on the Polyglot embeddings from Al-Rfou et al. (2013) (which are multilingual, but not in a shared representation space), and the two systems developed by Hermann and Blunsom (2014). The results are shown in Table 3. We observe that, with the exception of Turkish, our systems consistently outperform all the competitors. Comparing the bottom two rows of Tables 2 and 3 we also observe that, for the ℓ_2-regularized system, there is not much degradation caused by cross-lingual training versus training on the target language directly (in fact, for Spanish, Polish, and Brazilian Portuguese, the former scores are even higher). This suggests that the multilingual embeddings have high quality.

6 Conclusions

We proposed a new formulation which jointly minimizes a combination of a supervised loss function with a multilingual co-regularization term using unlabeled parallel data. This allows learning task-specific multilingual embeddings together with a classifier for the task. Our method achieved state-of-the-art accuracy on the Reuters RCV1/RCV2 cross-lingual classification task in the English to German direction, while being extremely simple and computationally efficient. Our results in the Reuters RCV1/RCV2 task, obtained using Europarl v7 as parallel data, show that our method has no trouble handling different levels of representations simutaneously (document, sentence and word). On the TED Corpus, we obtained the highest reported scores for 10 out of 11 languages, using an auxiliary task with pre-trained English embeddings.

Acknowledgments

We would like to thank the three anonymous reviewers. This work was partially supported by the European Union under H2020 project SUMMA, grant 688139, and by Fundação para a Ciência e Tecnologia (FCT),

[7] Note that, despite the name, the Hermann and Blunsom (2014)'s *joint* systems are not doing joint training as we are.

[8] Note however that, overall, our *Joint w/ Aux* systems have access to more data than our *Joint* systems and also than Hermann and Blunsom (2014)'s systems, since the pre-trained embeddings were trained on a large amount of English monolingual data. Yet, the amount of target language data is the same.

		Ara.	Ger.	Spa.	Fre.	Ita.	Dut.	Pol.	Br. Pt.	Rom.	Rus.	Tur.
MT Baseline	[HB14]	42.9	46.5	51.8	52.6	51.4	50.5	44.5	47.0	49.3	43.2	40.9
DOC/ADD *single*	[HB14]	41.0	42.4	38.3	47.6	48.5	26.4	40.2	35.4	41.8	44.8	45.2
DOC/ADD *joint*	[HB14]	39.2	40.5	44.3	44.7	47.5	45.3	39.4	40.9	44.6	47.6	41.7
Joint, $\mathcal{R}_{\ell_2}$, $K = 15$		41.8	46.6	46.6	46.0	48.7	52.5	39.5	40.8	47.6	44.9	47.2
Joint, $\mathcal{R}_{\ell_1}$, $K = 15$		44.0	44.7	49.4	40.1	46.1	49.4	35.7	43.5	40.5	42.2	43.4
Joint w/ Aux, $\mathcal{R}_{\ell_2}$, $K = 300$		**46.9**	**52.0**	**59.4**	**54.6**	**56.0**	**53.6**	**51.0**	**51.7**	**53.9**	**52.3**	**49.5**
Joint w/ Aux, $\mathcal{R}_{\ell_1}$, $K = 300$		44.0	40.4	40.4	39.5	38.6	38.1	43.2	36.6	35.1	44.3	44.4

Table 2: Cross-lingual experiments on the TED Corpus using English as a source language. Reported are the micro-averaged F_1 scores for a machine translation baseline and the two strongest systems of Hermann and Blunsom (2014), our one-stage joint system (*Joint*), and our two-stage system that trains the multilingual embeddings jointly with the auxiliary task of fitting pre-trained English embeddings (*Joint w/ Aux*), with both ℓ_1 and ℓ_2 regularization. Bold indicates the best result for each target language.

		Ara.	Ger.	Spa.	Fre.	Ita.	Dut.	Pol.	Br. Pt.	Rom.	Rus.	Tur.
BOW baseline	[HB14]	46.9	47.1	52.6	53.2	52.4	52.2	41.5	46.5	50.9	46.5	**51.3**
Polyglot	[HB14]	41.6	27.0	41.8	36.1	33.2	22.8	32.3	19.4	30.0	40.2	29.5
DOC/ADD Single	[HB14]	42.2	42.9	39.4	48.1	45.8	25.2	38.5	36.3	43.1	47.1	43.5
DOC/ADD Joint	[HB14]	37.1	38.6	47.2	45.1	39.8	43.9	30.4	39.4	45.3	40.2	44.1
Joint w/ Aux, $\mathcal{R}_{\ell_2}$, $K = 300$		48.6	**54.4**	57.5	**55.8**	**56.9**	**54.5**	46.1	**51.3**	**56.5**	**53.0**	49.5
Joint w/ Aux, $\mathcal{R}_{\ell_1}$, $K = 300$		**52.4**	47.8	**57.8**	50.0	53.3	52.3	**47.6**	49.0	49.2	51.4	50.9

Table 3: Monolingual experiments on the TED Corpus. Shown are the micro-averaged F_1 scores for a bag-of-words baseline, a system trained on Polyglot embeddings, the two strongest systems of Hermann and Blunsom (2014), and our *Joint w/ Aux* system with ℓ_1 and ℓ_2 regularization.

january_en	**science_en**	**oil_en**	**road_en**	**speak_en**
januari_nl	مولعلا_ar	óleo_pb	route_fr	spreken_nl
şubat_tr	ملعلا_ar	olie_nl	strada_it	fala_pb
gennaio_it	ciência_pb	petrolio_it	weg_nl	ملاكلا_ar
februarie_ro	science_fr	öl_de	drum_ro	gesproken_nl
ريارباف_ar	ştiinţa_ro	pétrole_fr	ريسلا_ar	habla_es
ianuarie_ro	wetenschap_nl	petrol_tr	estrada_pb	konuşma_tr
febrero_es	scienza_it	petróleo_es	drogi_pl	говорить_ru
janvier_fr	ciencia_es	طفنلا_ar	lopen_nl	horen_nl
ريانيد_ar	wissenschaft_de	petróleo_pb	strade_it	mowy_pl
janeiro_pb	científica_pb	petrol_ro	drodze_pl	vorbească_ro
enero_es	nauka_pl	aceite_es	wegen_nl	spreekt_nl
september_nl	bilim_tr	ropę_pl	yol_tr	ثيدحلا_ar
settembre_it	ştiinţa_ro	нефть_ru	camino_es	sprechen_de
septiembre_es	ştiinţă_ro	petrolul_ro	conduce_ro	ii_ro
september_de	nauki_pl	нефти_ru	andar_pb	discours_fr
ekim_tr	наука_ru	طفذ_ar	пути_ru	sentire_it
ربمتبس_ar	ملع_ar	ropy_pl	ريست_ar	contar_pb
febbraio_it	مولع_ar	تيز_ar	далеко_ru	себя_ru
septembrie_ro	scientifica_it	ulei_ro	yolculuk_tr	صخش_ar
setembro_pb	scienze_it	تيزلا	yola_tr	poser_fr

Table 4: Examples of nearest neighbor words for the multilingual embeddings trained with our *Joint w/ Aux* system with ℓ_2 regularization. Shown for each English word are the 20 closest target words in Euclidean distance, regardless of language.

through contracts UID/EEA/50008/2013, through the LearnBig project (PTDC/EEI-SII/7092/2014), and the GoLocal project (grant CMUPERI/TIC/0046/2014).

References

Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2013. Polyglot: Distributed word representations for multilingual NLP. *arXiv preprint arXiv:1307.1662* .

Yasemin Altun, Mikhail Belkin, and David A. McAllester. 2005. Maximum Margin Semi-Supervised Learning for Structured Variables. In *Advances in Neural Information Processing Systems 18*. pages 33–40.

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

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit3: Web inventory of transcribed and translated talks. In *Proc. of the 16th Conference of the European Association for Machine Translation*. pages 261–268.

Sarath Chandar, Stanislas Lauly, Hugo Larochelle, Mitesh M. Khapra, Balaraman Ravindran, Vikas Raykar, and Amrita Saha. 2014. An Autoencoder Approach to Learning Bilingual Word Representations.

Danqi Chen and Christopher D Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of Empirical Methods for Natural Language Processing*. pages 740–750.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of the International Conference on Machine Learning*. ACM, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Jocelyn Coulmance, Jean-Marc Marty, Guillaume Wenzek, and Amine Benhalloum. 2015. Transgram, Fast Cross-lingual Word-embeddings. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* pages 1109–1113.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research* 12:2121–2159.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2015. Retrofitting word vectors to semantic lexicons. In *Proc. of Annual Meeting of the North-American Chapter of the Association for Computational Linguistics*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proc. of Annual Meeting of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Kuzman Ganchev and Dipanjan Das. 2013. Cross-lingual discriminative learning of sequence models with posterior regularization. In *Proc. of Empirical Methods in Natural Language Processing*.

Kuzman Ganchev, Joao Graca, John Blitzer, and Ben Taskar. 2008. Multi-view learning over structured and non-identical outputs. In *Proc. of Conference on Uncertainty in Artificial Intelligence*.

Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. BilBOWA: Fast Bilingual Distributed Representations without Word Alignments. *Proceedings of the 32nd International Conference on Machine Learning (2015)* pages 748–756.

Karl Moritz Hermann and Phil Blunsom. 2014. Multilingual Models for Compositional Distributed Semantics. *Proceedings of ACL* pages 58–68.

Rebecca Hwa, Philip Resnik, Amy Weinberg, Clara Cabezas, and Okan Kolak. 2005. Bootstrapping parsers via syntactic projection across parallel texts. *Natural language engineering* 11(3):311–325.

Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing crosslingual distributed representations of words. *24th International Conference on Computational Linguistics - Proceedings of COLING 2012: Technical Papers (2012)* pages 1459–1474.

Philipp Koehn. 2005. Europarl: A parallel corpus

for statistical machine translation. *MT summit* 11.

Quoc Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014* 32:1188–1196.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research* 5:361–397.

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45:503–528.

Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Bilingual Word Representations with Monolingual Quality in Mind. *Workshop on Vector Modeling for NLP* pages 151–159.

André F. T. Martins. 2015. Transferring coreference resolvers with posterior regularization. In *ACL*.

Ryan McDonald, Slav Petrov, and Keith Hall. 2011. Multi-source transfer of delexicalized dependency parsers. In *Proc. of Empirical Methods in Natural Language Processing*.

Tomas Mikolov, Greg Corrado, Kai Chen, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations (ICLR 2013)* pages 1–12.

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

Kaare Brandt Petersen and Michael Syskind Pedersen. 2012. The Matrix Cookbook.

Hieu Pham, Minh-Thang Luong, and Christopher D. Manning. 2015. Learning Distributed Representations for Multilingual Text Sequences. *Workshop on Vector Modeling for NLP* pages 88–94.

Tianze Shi, Zhiyuan Liu, Yang Liu, and Maosong Sun. 2015. Learning Cross-lingual Word Embeddings via Matrix Co-factorization. *Annual Meeting of the Association for Computational Linguistics* pages 567–572.

Vikas Sindhwani, Partha Niyogi, and Mikhail Belkin. 2005. A co-regularization approach to semi-supervised learning with multiple views. In *Proceedings of ICML workshop on learning with multiple views*. Citeseer, pages 74–79.

Richard Socher, Jeffrey Pennington, and Eh Huang. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Conference on Empirical Methods in Natural Language Processing*. pages 151–161.

Hubert Soyer, Pontus Stenetorp, and Akiko Aizawa. 2015. Leveraging Monolingual Data for Crosslingual Compositional Word Representations. *Proceedings of the 2015 International Conference on Learning Representations (ICLR)* .

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of the Annual Meeting of the Association for Computational Linguistics*.

Daniel Zeman and Philip Resnik. 2008. Cross-language parser adaptation between related languages. In *IJCNLP*. pages 35–42.

Will Y Zou, Richard Socher, Daniel M Cer, and Christopher D Manning. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proc. of Empirical Methods for Natural Language Processing*. pages 1393–1398.

Supersense Embeddings: A Unified Model for Supersense Interpretation, Prediction, and Utilization

Lucie Flekova[†] **and Iryna Gurevych**[†‡]
† Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
‡ Ubiquitous Knowledge Processing Lab (UKP-DIPF)
German Institute for Educational Research
www.ukp.tu-darmstadt.de

Abstract

Coarse-grained semantic categories such as supersenses have proven useful for a range of downstream tasks such as question answering or machine translation. To date, no effort has been put into integrating the supersenses into distributional word representations. We present a novel joint embedding model of words and supersenses, providing insights into the relationship between words and supersenses in the same vector space. Using these embeddings in a deep neural network model, we demonstrate that the supersense enrichment leads to a significant improvement in a range of downstream classification tasks.

1 Introduction

The effort of understanding the meaning of words is central to the NLP community. The word sense disambiguation (WSD) task has therefore received a substantial amount of attention (see Navigli (2009) or Pal and Saha (2015) for an overview). Words in training and evaluation data are usually annotated with senses taken from a particular lexical semantic resource, most commonly WordNet (Miller, 1995). However, WordNet has been criticized to provide too fine-grained distinctions for end level applications. e.g. in machine translation or information retrieval (Izquierdo et al., 2009). Although some researchers report an improvement in sentiment prediction using WSD (Rentoumi et al., 2009; Akkaya et al., 2011; Sumanth and Inkpen, 2015), the publication bias toward positive results (Plank et al., 2014) impedes the comparison to experiments with the opposite conclusion, and the contribution of WSD to downstream document classification tasks remains "mostly speculative"(Ciaramita and Altun, 2006), which can be attributed to the too subtle sense distinctions (Navigli, 2009). This is why *supersenses*, the coarse-grained word labels based on WordNet's (Fellbaum, 1998) lexicographer files, have recently gained attention for text classification tasks. Supersenses contain 26 labels for nouns, such as ANIMAL, PERSON or FEELING and 15 labels for verbs, such as COMMUNICATION, MOTION or COGNITION. Usage of supersense labels has been shown to improve dependency parsing (Agirre et al., 2011), named entity recognition (Marrero et al., 2009; Rüd et al., 2011), non-factoid question answering (Surdeanu et al., 2011), question generation (Heilman, 2011), semantic role labeling (Laparra and Rigau, 2013), personality profiling (Flekova and Gurevych, 2015), semantic similarity (Severyn et al., 2013) and metaphor detection (Tsvetkov et al., 2013).

An alternative path to semantic interpretation follows the distributional hypothesis (Harris, 1954). Recently, word vector representations learned with neural-network based language models have contributed to state-of-the-art results on various linguistic tasks (Bordes et al., 2011; Mikolov et al., 2013b; Pennington et al., 2014; Levy et al., 2015).

In this work, we present a novel approach for incorporating the supersense information into the word embedding space and propose a new methodology for utilizing these to label the text with supersenses and to exploit these joint word and supersense embeddings in a range of applied text classification tasks. Our contributions in this work include the following:

- We are the first to provide a joint word- and supersense-embedding model, which we make publicly available[1] for the research community. This provides an insight into the word and supersense positions in the vector space

[1] https://github.com/UKPLab/acl2016-supersense-embeddings

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2029–2041,
Berlin, Germany, August 7-12, 2016.

through similarity queries and visualizations, and can be readily used in any word embedding application.

- Using this information, we propose a supersense tagging model which achieves competitive performance on recently published social media datasets.
- We demonstrate how these predicted supersenses and their embeddings can be used in a range of text classification tasks. Using a deep neural network architecture, we achieve an improvement of 2-6% in accuracy for the tasks of sentiment polarity classification, subjectivity classification and metaphor prediction.

2 Related Work

2.1 Semantically Enhanced Word Embeddings

An idea of combining the distributional information with the expert knowledge is attractive and has been newly pursued in multiple directions. One of them is creating the word sense or synset embeddings (Iacobacci et al., 2015; Chen et al., 2014; Rothe and Schütze, 2015; Bovi et al., 2015). While the authors demonstrate the utility of these embeddings in tasks such as WSD, knowledge base unification or semantic similarity, the contribution of such vectors to downstream document classification problems can be challenging, given the fine granularity of the WordNet senses (cf. the discussion in Navigli (2009)). As discussed above, supersenses have been shown to be better suited for carrying the relevant amount of semantic information. An alternative approach focuses on altering the objective of the learning mechanism to capture relational and similarity information from knowledge bases (Bordes et al., 2011; Bordes et al., 2012; Yu and Dredze, 2014; Bian et al., 2014; Faruqui and Dyer, 2014; Goikoetxea et al., 2015). While, in principle, supersenses could be seen as a relation between a word and its hypernym, to our knowledge they have not been explicitly employed in these works. Moreover, an important advantage of our explicit supersense embeddings compared to the retrained vectors is their direct interpretability.

2.2 Supersense Tagging

Supersenses, also known as lexicographer files or semantic fields, were originally used to organize lexical-semantic resources (Fellbaum, 1990). The supersense tagging task was introduced by Ciaramita and Johnson (2003) for nouns and later expanded for verbs (Ciaramita and Altun, 2006). Their state-of-the-art system is trained and evaluated on the SemCor data (Miller et al., 1994) with an F-score of 77.18%, using a hidden Markov model. Since then, the system, resp. its reimplementation by Heilman[2], was widely used in applied tasks (Agirre et al., 2011; Surdeanu et al., 2011; Laparra and Rigau, 2013). Supersense taggers have then been built also for Italian (Picca et al., 2008), Chinese (Qiu et al., 2011) and Arabic (Schneider et al., 2013). Tsvetkov et al. (2015) proposes the usage of SemCor supersense frequencies as a way to evaluate word embedding models, showing that a good alignment of embedding dimensions to supersenses correlates with performance of the vectors in word similarity and text classification tasks. Recently, Johannsen et al. (2014) introduced a task of multiword supersense tagging on Twitter. On their newly constructed dataset, they show poor domain adaptation performance of previous systems, achieving a maximum performance with a search-based structured prediction model (Daumé III et al., 2009) trained on both Twitter and SemCor data. In parallel, Schneider and Smith (2015) expanded a multiword expression (MWE) annotated corpus of online reviews with supersense information, following an alternative annotation scheme focused on MWE. Similarly to Johannsen et al. (2014), they find that SemCor may not be a sufficient resource for supersense tagging adaption to different domains. Therefore, in our work, we explore the potential of using an automatically annotated Babelfied Wikipedia corpus (Scozzafava et al., 2015) for this task.

3 Building Supersense Embeddings

To learn our embeddings, we adapt the freely available sample of 500k articles of Babelfied English Wikipedia (Scozzafava et al., 2015). To our knowledge, this is one of the largest published and evaluated sense-annotated corpora, containing over 500 million words, of which over 100 million are annotated with Babel synsets, with an estimated synset annotation accuracy of 77.8%. Few other automatically sense-annotated Wikipedia corpora are available (Jordi Atserias and Attardi, 2008; Reese et

[2] https://github.com/kutschkem/SmithHeilmann_fork/tree/master/MIRATagger

.1	*About 10.9% of families were below the poverty line, including 13.6% of those under age 18.*
2	*About 10.9% of* N.GROUP *were below the* N.POSSESSION V.CHANGE *13.6% of those under* N.ATTRIBUTE *18.*
3	*About 10.9% of* FAMILIES_N.GROUP *were below the* POVERTY_LINE_N.POSSESSION INCLUDING_V.CHANGE *13.6% of those under* AGE_N.ATTRIBUTE *18.*

Table 1: Example of plain (1), generalized (2) and disambiguated (3) Wikipedia

al., 2010). However, their annotation quality was assessed only on the training domain and as Atserias et al. state (p.2316): *"Wikipedia text differs significantly ... from the corpora used to train the taggers ... Therefore the quality of these NLP processors is considerably lower than the results of the evaluation in-domain."*

We map the Babel synsets to WordNet 3.0 synsets (Miller, 1995) using the BabelNet API (Navigli and Ponzetto, 2012), and map these synsets to their corresponding WordNet's supersense categories (Miller, 1990; Fellbaum, 1990). For the nested named entities, only the largest BabelNet span is considered, hence there are no nested supersense labels in our data. In this manner we obtain an alternative Wikipedia corpus, where each word is replaced by its corresponding supersense (see Table 1, second row) and another alternative corpus where each word has its supersense appended (Table 1, third row). Using the Gensim (Řehůřek and Sojka, 2010) implementation of Word2vec (Mikolov et al., 2013a), we applied the skip-gram model with negative sampling on these three Wikipedia corpora jointly (i.e., on the rows 1, 2 and 3 in Table 1) to produce continuous representations of words, supersense-disambiguated words and standalone supersenses in one vector space based on the distributional information obtained from the data. [3] The benefits of learning this information jointly are threefold:

1. Vectorial representations of the original words are altered (compared to training on text only), taking into account the similarity to supersenses in the vector space
2. Standalone supersenses are positioned in the vector space, enabling insightful similarity queries between words and supersenses, esp. for unannotated words
3. Disambiguated word+supersense vectors of annotated words can be employed similarly to sense embeddings (Iacobacci et al., 2015; Chen et al., 2014) to improve downstream tasks and serve as input for supersense disambiguation or contextual similarity systems

In the following, the designation WORDS denotes the experiments with the word embeddings learned on plain Wikipedia text (as in row 1 of Table 1) while the designation SUPER denotes the experiments with the word embeddings learned jointly on the supersense-enriched Wikipedia (i.e., rows 1, 2 and 3 in Table 1 together).

4 Qualitative Analysis

4.1 Verb Supersenses

Table 2 shows the most similar word vectors to each of the verb supersense vectors using cosine similarity. Note that while no explicit part-of-speech information is specified, the most similar words hold both the semantic and syntactic information - most of the assigned words are verbs.

VERBS	
BODY	wearing, injured, worn, wear, wounded, bitten, soaked, healed, cuffed, dressed
CHANGE	changed, started, added, dramatically, expanded drastically, begun, altered, shifted, transformed
COGNITION	known, thought, consider, regarded, remembered attributed, considers, accepted, believed, read
COMMUNICATION	stated, said, argued, jokingly, called, noted, suggested, described, claimed, referred
COMPETITION	won, played, lost, beat, scored defeated, win, competed, winning, playing
CONSUMPTION	feed, fed, employed, based, hosted feeds, utilized, applied, provided, consumed
CONTACT	thrown, set, carried, opened, laid pulled, placed, cut, dragged, broken
CREATION	produced, written, created, designed, developed directed, built, published, penned, constructed
EMOTION	want, felt, loved, wanted, delighted disappointed, feel, like, saddened, thrilled
MOTION	brought, led, headed, returned, followed left, turned, sent, travelled, entered
PERCEPTION	seen, shown, revealed, appeared, appears shows, noticed, see, showing, presented
POSSESSION	received, obtained, awarded, acquired, provided donated, gained, bought, found, sold
SOCIAL	appointed, established, elected, joined, assisted led, succeeded, encouraged, initiated, organized
STATIVE	included, held, includes, featured, served, represented, referred, holds, continued, related
WEATHER	glow, emitted, ignited, flare, emitting smoke, fumes, sunlight, lit, darkened

Table 2: Top 10 most similar word embeddings for verb supersense vectors

[3] The embeddings are learned using skip-gram as training algorithm with downsampling of 0.001 higher-frequency words, negative sampling of 5 noise words, minimal word frequency of 100, window of size 2 and alpha of 0.025, using 10 epochs to produce 300-dimensional vectors. Our experiments with less dimensions and with the CBOW model performed worse.

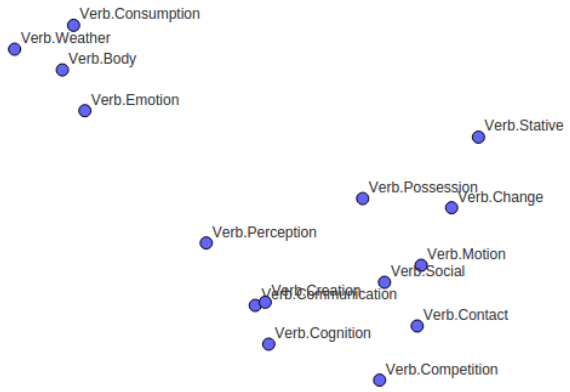


Figure 1: Verb supersense embeddings visualized in the vector space (t-SNE)

Furthermore, using a large corpus such as Wikipedia conveniently reduces the current need of lemmatization for supersense tagging, as the words are sufficiently represented in all their forms. The most frequent error originates from assigning the adverbs to their related verb categories, e.g. *jokingly* to COMMUNICATION and *drastically* to CHANGE. Such information, however, can be beneficial for context analysis in supersense tagging.

Figure 1 displays the verb supersenses using the t-distributed Stochastic Neighbor Embedding (Van der Maaten and Hinton, 2008), a technique designed to visualize structures in high-dimensional data. While many of the distances are probable to be dataset-agnostic, such as the proximity of BODY, CONSUMPTION and EMOTION, other appear emphasized by the nature of Wikipedia corpus, e.g. the proximity of supersenses COMMUNICATION and CREATION or SOCIAL and MOTION, as can be explained by table 2 (see e.g. *led* and *followed*).

Figure 2: Noun supersense embeddings (t-SNE)

4.2 Noun Supersenses

Table 3 displays the most similar word embeddings for noun supersenses. In accordance with previous work on suppersense tagging (Ciaramita and Altun, 2006; Schneider et al., 2012; Johannsen et al., 2014), the assignments of more specific supersenses such as FOOD, PLANT, TIME or PERSON are in general more plausible than those for abstract concepts such as ACT, ARTIFACT or COGNITION. The same is visible in Figure 2, where these supersense embeddings are more central, with closer neighbors. In contrast to the observations by Schneider et al. (2012) and Johannsen et al. (2014), the COMMUNICATION supersense appears well defined, likely due to the character of Wikipedia.

NOUNS	
ACT	participation, activities, involvement, undertaken ongoing, conduct, efforts, large-scale, success
ANIMAL	peccaries, capybaras, frogs, echidnas, birds marmosets, rabits, hatchling, ciconiidae, species
ARTIFACT	wooden, two-floor, purpose-built, installed, wall fittings, turntable, racks, wrought-iron, ceramic, stone
ATTRIBUTE	height, strength, age, versatility, hardness power, fluidity, mastery, brilliance, inherent
BODY	abdomen, bone, femur, anterior, forearm femoral, skin, neck, muscles, thigh
COGNITION	ideas, concepts, empirical, philosophy, knowledge, epistemology, analysis, atomistic, principles
COMMUNICATION	written, excerpts, text, music, excerpted, translation, lyrics, subtitle, transcription, words
EVENT	sudden, death, occurred, event, catastrophic unexpected, accident, victory, final, race
FEELING	sadness, love, sorrow, frustration, disgust anger, affection, feelings, grief, fear
FOOD	cheese, butter, coffee, milk, yogurt dessert, meat, bread, vegetables, sauce
GROUP	members, school, phtheochroa, ypsolophidae pitcairnia, cryptanthus, group, division, schools
LOCATION	northern, southern, northeastern, area, south capital, town, west, region, city
MOTIVE	motivation, reasons, rationale, justification, motive justifications, motives, incentive, desire, why
OBJECT	river, valley, lake, hills, floodplain lakes, rivers, mountain, estuary, ocean
PERSON	greatgrandfather, son, nephew, son-in-law, father halfbrother, brother, who, mentor, fellow
PHENOMENON	wind, forces, self-focusing, radiation, ionizing result, intensity, gravitational, dissipation, energy
PLANT	fruit, fruits, magnifera, sativum, flowers caesalpinia, shrubs, trifoliate, vines, berries
POSSESSION	property, payment, money, payments, taxes tax, cash, fund, pay, $100
PROCESS	growth, decomposition, oxidative, mechanism rapid, reaction, hydrolysis, inhibition, development
QUANTITY	miles, square, meters, kilometer, cubic, ton, number, megabits, volume, kilowatthours
RELATION	southeast, southwest, northeast, northwest, east portion, link, correlation, south, west
SHAPE	semicircles, right-angled, concave, parabola, ellipse, angle, circumcircle, semicircle, lines
STATE	chronic, condition, debilitating, problems, health worsening, illness, illnesses, exacerbation, disease
SUBSTANCE	magnesium, zinc, silica, manganese, sulfur oxide, sulphate, phosphate, salts, phosphorus
TIME	september, december, november, july, april january, august, february, year, days
TOPS	time, group, event, person, groups individuals, events, animals, individual, plant

Table 3: Top 10 most similar word embeddings for noun supersense vectors

4.3 Word Analogy and Word Similarity Tasks

We also assess the changes between the individual word embeddings learned on plain Wikipedia text (WORDS) and jointly with the supersense-enriched Wikipedia (SUPER). With this aim we perform two standard embedding evaluation tasks: word similarity and word analogy.

Mikolov et al. (2013b) introduce a word analogy dataset containing 19544 analogy questions that can be answered with word vector operations (*Paris is to France as Athens are to...?*). The questions are grouped into 13 categories. Table 4 presents our results. Word vectors trained in the SUPER setup achieve better results on groups related to entities, e.g. Family Relations and Citizen to State questions, where the PERSON and LOCATION supersenses can provide additional information to reduce noise. At the same time, performance on questions such as Opposites or Plurals drops, as this information is pushed to the background. Enriching our data with the recently proposed adjective supersenses (Tsvetkov et al., 2014) could be of interest for these categories.

Group/Vectors:	WORDS	SUPER
Capitals - common	91.1	94.7±0.99
Capitals - world	87.6	89.5±0.69
City in state	65.2	65.7±1.03
Nationality to state	94.5	95.2±0.58
Family relations	93.0	94.4±1.28
Opposites	56.7	54.6±3.21
Plurals	89.4	86.4±1.08
Comparatives	90.6	90.4±0.85
Superlatives	79.4	79.6±1.83
Adjective to adverb	20.2	22.2±1.53
Present to participle	64.2	64.6±1.57
Present to past	60.0	59.2±1.30
3rd person verbs	84.3	82.1±1.44
Total	75.0	76.0±0.28

Table 4: Accuracy and standard error on analogy tasks. Tasks related to noun supersense distinctions show the tendency to improve, while syntax-related information is pushed to the background. In most cases, however, the difference is not significant.

Without explicitly exploiting the sense infromation, we compare the performance of our text-trained (WORDS) to our jointly trained (SUPER) word vectors on the following word similarity datasets: WordSim353-Similarity (353-S) and WordSim353-Relatedness (353-R) (Agirre et al., 2009), MEN dataset (Bruni et al., 2014), RG-65 dataset (Rubenstein and Goodenough, 1965) and MC-30 (Miller and Charles, 1991).

Data:	MEN	353-S	353-R	RG-65	MC-30
WORDS	73.18	76.93	62.11	79.13	79.49
SUPER	74.26	78.63	61.22	79.75	80.94

Table 5: Performance of our vectors (Spearman's ρ) on five similarity datasets. Results indicate a trend of better performance of vectors trained jointly with supersenses.

The word embeddings for words trained jointly with supersenses achieve higher performance than those trained solely on the same text without supersenses on 4 out of 5 tasks (Table 5). In addition, the explicit supersense information could be further exploited, similarly to previous sense embedding works (Iacobacci et al., 2015; Rothe and Schütze, 2015; Chen et al., 2014). Furthermore, note that while we report the performance of our embeddings on the word similarity tasks for completeness, there has been a substantial discussion on seeking alternative ways to quantify embedding quality with the focus on their purpose in downstream applications (Li and Jurafsky, 2015; Faruqui et al., 2016). Therefore, in the remainder of this paper we explore the usefulness of supersense embeddings in text classification tasks.

5 Building a Supersense Tagger

The task of predicting supersenses has recently regained its popularity (Johannsen et al., 2014; Schneider and Smith, 2015), since supersenses provide disambiguating information, useful for numerous downstream NLP tasks, without the need of tedious fine-grained WSD. Exploiting our joint embeddings, we build a deep neural network model to predict supersenses on the Twitter supersense corpus created by Johannsen et al. (2014), based on the Twitter NER task (Ritter et al., 2011), using the same training data as the authors. [4] [5] The datasets follow the token-level annotation which combines the B-I-O flags (Ramshaw and Marcus, 1995) with the supersense class labels to represent the multiword expression segmentation and supersense labeling in a sentence.

5.1 Experimental Setup

We implement a window-based approach with a multi-channel multi-layer perceptron model using

[4] https://github.com/kutschkem/SmithHeilmann_fork/tree/master/MIRATagger/data

[5] https://github.com/coastalcph/supersense-data-twitter

the Theano framework (Bastien et al., 2012). With a sliding window of size 5 for the sequence learning setup we extract for each word the following seven feature vectors:

1. 300-dimensional word embedding,
2. 41 cosine similarities of the word to each standalone supersense embedding,
3. 41 cosine similarities of the word to each of its *word*_SUPERSENSE embeddings,
4. fixed vector of frequencies of each supersense in Wikipedia, in order to simulate the MFS backoff strategy,
5. for the given word, the frequency of each *word*_SUPERSENSE in our Wikipedia corpus,
6. part-of-speech information as a unit vector,
7. casing information as a 3-dimensional (upper/lower/mixed) unit vector

After a dropout regularization, the embedding sets are flattened, concatenated and fed into fully connected dense layers with a rectified linear unit (ReLU) activation function and a final softmax.

5.2 Supersense Prediction

We evaluate our system on the same Twitter dataset with provided training and development (`Tw-R-dev`) set and two test sets: `Tw-R-eval`, reported by Johannsen et al. as *RITTER*, and `Tw-J-eval`, reported by Johannsen et al. as *INHOUSE*. Our results are shown in table 6 and compared to results reported in previous work by Johannsen et al. (2014), with two additional baselines: The SemCor system of Ciaramita and Altun (2006) and the most frequent sense. Our system achieves comparable performance to the best previously used supervised systems, without using any explicit gazetteers.

To get an intuition.[6] of how the individual feature vectors contribute to the prediction, we perform an ablation test by removing one feature group at a time. The biggest performance drop in the F-score (2.7–5.4) occurs when removing the the part of speech information, followed by the supersense similarity features and supersense frequency priors (0.2–3.0). The casing information has only a minor contribution to Twitter supersense tagging (0–0.9).

System/Data:	Tw-R-dev	Tw-R-eval	Tw-J-eval
Baseline and upper bound			
Most frequent sense	47.54	44.98	38.65
Inter-annotator agreement		69.15	61.15
SemCor-trained systems			
(Ciaramita and Altun, 2006)†	48.96	45.03	39.65
Searn (Johannsen et al., 2014)	56.59	50.89	40.50
HMM (Johannsen et al., 2014)	57.14	50.98	41.84
Ours Semcor	54.47	50.30	35.61
Twitter-trained systems			
Searn (Johannsen et al., 2014)	67.72	57.14	42.42
HMM (Johannsen et al., 2014)	60.66	51.40	41.60
Ours Twitter (all features)	61.12	57.16	41.97
Ours Twitter no casing	61.06	56.20	41.13
Ours Twitter no similarities	63.47	56.78	39.44
Ours Twitter no frequencies	61.10	57.32	39.02
Ours Twitter no part-of-speech	57.08	54.45	36.50
Ours Twitter no word embed.	57.57	53.43	34.91

Table 6: Weighted F-score performance on supersense prediction for the development set and two test sets provided by Johannsen et al. (2004). Our system performs comparably to state-of-the-art systems.
† For the system of Ciaramita et al, the publicly avaliable reimplementation of Heilman was used

6 Using Supersense Embeddings in Document Classification Tasks

Word sense disambiguation is to some extent an artificial stand-alone task. Despite its popularity, its contribution to downstream document classification tasks remains rather limited, which might be attributed to the complexity of document preprocessing and the errors cumulated along the pipeline. In this section, we demonstrate an alternative, deep learning approach, in which we process the original text in parallel to the supersense information. The model can then flexibly learn the usefulness of provided input. We demonstrate that the model extended with supersense embeddings outperforms the same model using only word-based features on a range of classification tasks.

6.1 Experimental Setup

Both Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) are state-of-the-art semantic composition models for a variety of text classification tasks (Kim, 2014; Li et al., 2015; Johnson and Zhang, 2014). Recently, their combinations have been proposed, achieving an unprecedented performance (Sainath et al., 2015). We extend the CNN-LSTM approach from the publicly available

[6] Intuition, since there are many additional aspects that may affect the performance. For example, we keep the network parameters fixed for the ablation, although the feature vectors are of different lengths. Furthermore, our model performs a concatenation of the feature vectors, hence only the ablation extended to all possible permutations would verify the feature order effect.

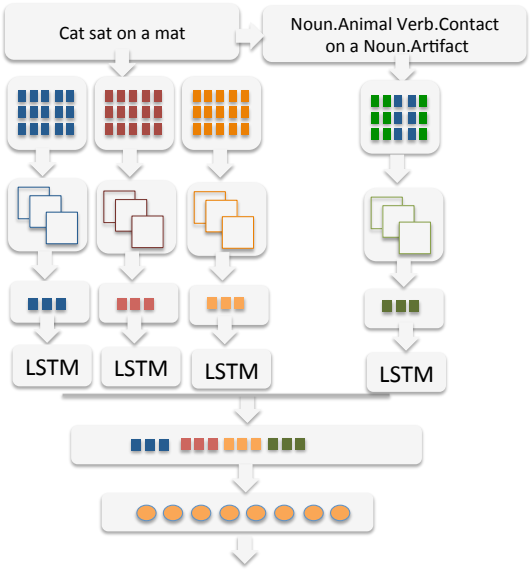


Figure 3: Network architecture. Each of the four different embedding channels serves as input to its CNN layer, followed by an LSTM layer. Afterwards, the outputs are concatenated and fed into a dense layer.

Keras demo[7], into which we incorporate the supersense information. Figure 3 displays our network architecture. First, we use three channels of word embeddings on the plain textual input. The first channel are the 300-dimensional word embeddings obtained from our enriched Wikipedia corpus. The second embedding channel consists of 41-dimensional vectors capturing the cosine similarity of the word to each supersense embedding. The third channel contains the vector of relative frequencies of the word occurring in the enriched Wikipedia together with its supersense, i.e. providing the background supersense distribution for the word. Each of the document embeddings is then convoluted with the filter size of 3, followed by a pooling layer of length 2 and fed into a long-short-term-memory (LSTM) layer. In parallel, we feed as input a processed document text, where the words are replaced by their predicted supersenses. Given that we have the Wikipedia-based supersense embeddings in the same vector space as the word embeddings, we can now proceed to creating the 300-dimensional embedding channel also for the supersense text. As in the plain text channels, we feed also these embeddings into the convolutional and LSTM layers in a similar fashion. Afterwards, we concatenate all LSTM outputs and feed them into a standard fully connected neural network layer, followed by the sigmoid for the binary output. The following subsections discuss our results on a range of classification tasks: subjectivity prediction, sentiment polarity classification and metaphor detection.

6.2 Sentiment Polarity Classification

Sentiment classification has been a widely explored task which received a lot of attention. The Movie Review dataset, published by Pang and Lee (2005)[8], has become a standard machine learning benchmark task for binary sentence classification. Socher et al. (2011) address this task with recursive autoencoders and Wikipedia word embeddings, later improving their score using recursive neural network with parse trees (Socher et al., 2012). Competitive results were achieved also by a sentiment-analysis-specific parser (Dong et al., 2015), with a fast dropout logistic regression (Wang and Manning, 2013), and with convolutional neural networks (Kim, 2014). Table 7 compares these approaches to our results for a 10-fold crossvalidation with 10% of the data withheld for parameter tuning. The line *WORDS* displays the performance using only the leftmost part of our architecture, i.e. only the text input with our word embeddings. The line *SUPER* shows the result of using the full supersense architecture. As it can be seen from the table, the supersense features improve the accuracy by about 2%. Both systems are significantly different ($p < 0.01$), using the McNemar's test.

System	Accuracy
Socher et al. (2011)	77.7
Socher et al. (2012)	79.0
Wang and Manning (2013)	79.1
Dong et al. (2015)	79.5
Kim (2014)	81.5
WORDS	79.4
SUPER	**81.7**±0.37

Table 7: 10-fold cross-validation accuracy and standard error of our system and as reported in previous work for the sentiment classification task on Pang and Lee (2005) movie review data

A detailed analysis of the supersense-tagged data and the classification output revealed that supersenses help to generalize over rare terms. Noun

[7] https://github.com/fchollet/keras/blob/master/examples/imdb_cnn_lstm.py

[8] http://www.cs.uic.edu/liub/FBS/sentiment-analysis.html

Positive reviews	
Text	Supersenses
beating the austin powers film at their own game , this blaxploitation spoof downplays the raunch in favor of gags that rely on the strength of their own cleverness as oppose to the extent of their outrageousness .	verbstative the nounlocation nouncognition nounartifact at their own nouncommunication , this nounact nouncommunication verbstative the nouncognition in nouncommunication of that verbcognition on the nouncognition of their own nouncognition as verbcommunication to the nounevent of their nounattribute .
there is problem with this film that even 3 oscar winner ca n't overcome , but it 's a nice girl-buddy movie once it get rock-n-rolling .	there verbstative nouncognition with this nouncommunication that even 3 nounevent nounperson ca n't verbemotion , but it verbstative a nice girl-buddy nouncommunication once it verbstative rock-n-rolling
godard 's ode to tackle life 's wonderment is a rambling and incoherent manifesto about the vagueness of topical excess . in praise of love remain a ponderous and pretentious endeavor that 's unfocused and tediously exasperating .	nounperson nouncommunication to verbstative nouncognition 's nouncognition verbstative a rambling and incoherent nouncommunication about the nounattribute of topical excess . in nouncognition of nouncognition verbstative a ponderous and pretentious nounact that verbstative unfocused and tediously exasperating
Negative reviews	
Text	Supersenses
the action scene has all the suspense of a 20-car pileup , while the plot hole is big enough for a train car to drive through – if kaos have n't blow them all up .	the nounact nounlocation verbstative all the nouncognition of a 20-car nouncognition , while the nounlocation verbstative big enough for a nounartifact nounartifact to verbmotion through – if nounperson have n't verbcommunication them all up .
the scriptwriter is no less a menace to society than the film 's character .	the nounperson verbstative no less nounstate to noungroup than the nouncommunication nounperson .
a very slow , uneventful ride around a pretty tattered old carousel .	a very slow , uneventful nounact around a pretty tattered old nounartifact .
the milieu is wholly unconvincing . . . and the histrionics reach a truly annoying pitch .	the nouncognition verbstative wholly unconvincing and the nouncommunication verbstative a truly annoying nounattribute .

Table 8: Example of documents classified incorrectly with word embeddings and correctly with word and supersense embeddings on Pang and Lee (2005) movie review data.

concepts such as GROUP, LOCATION, TIME and PERSON appear somewhat more frequently in positive reviews while certain verb supersenses such as PERCEPTION, SOCIAL and COMMUNICATION are more frequent in the negative ones. On the other hand, the supersense tagging introduces additional errors too - for example the director's *cut* is persistently classified into FOOD.

Table 8 shows an example of positive and negative reviews which were consistently (5x in repeated experiments with different random seeds) classified incorrectly with word embeddings and classified correctly with supersense embeddings. Often the wit of unusual expressions is lost for the benefit of generalization. Some improvements appear to be a result of replacing proper names by NOUN.PERSON.

6.3 Subjectivity Classification

Pang and Lee (2004) demonstrate that the subjectivity detection can be a useful input for a sentiment classifier. They compose a publicly available dataset[9] of 5000 subjective and 5000 objective sentences, classifying them with a reported accuracy of 90-92% and further show that predicting this information improves the end-level sentiment classification on a movie review dataset. Kim (2014) and Wang and Manning (2013) further improve the performance through different machine learning methods. Supersenses are a natural candidate for subjectivity prediction, as we hypothesize that the nouns and verbs in the subjective and objective sentences often come from different semantic classes (e.g. VERB.FEELING vs. VERB.COGNITION). We employ the same architecture as in previous task, automatically annotating the words in the documents with their supersenses. Our results are reported in Table 9. The supersenses (SUPER) provide an additional information, improving the model performance by up to 2% over word embeddings (WORDS). The difference between both systems is significant. Based on a manual error analysis, the supersense information contributes here in a similar manner as in the previous case. Subjective sentences contain more verbs of supersense PERCEPTION, while objective ones more frequently feature the supersenses POSSESSION and SOCIAL. Nouns in the subjective category are characterized by supersenses COMMUNICATION and ATTRIBUTE, while in objective ones the PERSON and POSSESSION are more frequent.

System	Accuracy
SVM (Pang and Lee, 2004)	90.0
NB (Pang and Lee, 2004)	92.0
CNN (Kim, 2014)	93.4
F-Dropout (Wang and Manning, 2013)	93.6
MV-CNN (Zhang et al., 2016)	93.9
WORDS	92.1
SUPER	**93.9**$\pm$0.26

Table 9: 10-fold cross-validation accuracy and standard error of our system and as reported in previous work for binary classification on the subjectivity dataset of Pang and Lee (2004)

[9] https://www.cs.cornell.edu/people/pabo/movie-review-data/

6.4 Metaphor Identification

Supersenses have recently been shown to provide improvements in metaphor prediction tasks (Gershman et al., 2014), as they hold the information of coarse semantic concepts. Turney et al. (2011) explore the task of discriminating literal and metaphoric adjective-noun expressions. They report an accuracy of 79% on a small dataset rated by five annotators. Tsvetkov et al. (2013) pursue this work further by constructing and publishing a dataset of 985 literal and 985 methaphorical adjective-noun pairs[10] and classify them. Gershman et al. (2014) further expand on this work using 64-dimensional vector-space word representations constructed by Faruqui and Dyer (2014) for classification. They report a state-of-the-art F-score of 85% with random decision forests, including also abstractness and imageability features (Wilson, 1988) and supersenses from WordNet, averaged across senses.

System	F1-score on test set
(Gershman et al., 2014)	85
WORDS	81.91±2.81
SUPER	**87.23**±2.36

Table 10: F1-score and a standard error on a provided test set for the adjective-noun metaphor prediction task Gershman et al. (2014). WORDS: word embeddings only, SUPER: multi-channel word embeddings with the supersense similarity and frequency vectors added

Since this setup is simpler than the sentence classification tasks, we use only a subset of our architecture, specifically the left half of Figure 3, i.e. our word embeddings, similarity vectors and supersense frequency vectors. Since there are only two words in each document, we leave out the LSTM layer. We merge the similarity and frequency layers by multiplication and concatenate the result to the word embedding convolution, feeding the output of the concatenation directly to the dense layer. Table 10 shows our results on a provided test set. Based on McNemar's test, there is a significant difference ($p < 0.01$) between our system based on words only and the one with supersenses.

7 Discussion

Unlike previous research on supersenses, our work is not based on a manually produced gold standard, but on an automatically annotated large corpus. While Scozzafava et al. (2015) report a high accuracy estimate of 77.8% on sense level, the performance and possible bias on tagged supersenses are yet to be evaluated. We are also aware that some of the previously proposed approaches for building word sense embeddings (Rothe and Schütze, 2015; Chen et al., 2014; Iacobacci et al., 2015) could be eventually extended to supersenses. We strongly encourage the authors to do so and perform a contrastive evaluation comparing these methods. Additionaly, a different level of granularity of the concepts, such as WordNet Domains (Magnini and Cavaglia, 2000) could be explored.

8 Conclusions and Future Work

We have presented a novel joint embedding set of words and supersenses, which provides a new insight into the word and supersense positions in the vector space. We demonstrated the utility of these embeddings for predicting supersenses and manifested that the supersense enrichment can lead to a significant improvement in a range of downstream classification tasks, using our embeddings in a neural network model. The outcomes of this work are available to the research community.[11]. In follow-up work, we aim to apply our embedding method on smaller, yet gold-standard corpora such as SemCor (Miller et al., 1994) and STREUSLE (Schneider and Smith, 2015) to examine the impact of the corpus choice in detail and extend the training data beyond WordNet vocabulary. Moreover, the coarse semantic categorization contained in supersenses was shown to be preserved in translation (Schneider et al., 2013), making them a perfect candidate for a multilingual adaptation of the vector space, e.g. extending Faruqui and Dyer (2014).

Acknowledgments

This work has been supported by the Volkswagen Foundation as part of the Lichtenberg Professorship Program under grant No. I/82806 and by the German Research Foundation under grant No. GU 798/14-1. Additional support was provided by the German Federal Ministry of Education and Research (BMBF) as a part of the Software Campus program under the reference 01-S12054 and by the German Institute for Educational Research (DIPF). We thank the anonymous reviewers for their input.

[10] http://www.cs.cmu.edu/~ytsvetko/metaphor/datasets.zip

[11] https://github.com/UKPLab/acl2016-supersense-embeddings

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and WordNet-based approaches. In *Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 19–27. Association for Computational Linguistics.

Eneko Agirre, Kepa Bengoetxea, Koldo Gojenola, and Joakim Nivre. 2011. Improving dependency parsing with semantic classes. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: short papers-Volume 2*, pages 699–703. Association for Computational Linguistics.

Cem Akkaya, Janyce Wiebe, Alexander Conrad, and Rada Mihalcea. 2011. Improving the impact of subjectivity word sense disambiguation on contextual opinion analysis. In *Proceedings of the 15th Conference on Computational Natural Language Learning*, pages 87–96. Association for Computational Linguistics.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Jiang Bian, Bin Gao, and Tie-Yan Liu. 2014. Knowledge-powered deep learning for word embedding. In *Machine Learning and Knowledge Discovery in Databases*, pages 132–148. Springer.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Claudio Delli Bovi, Luis Espinosa Anke, and Roberto Navigli. 2015. Knowledge base unification via sense embeddings and disambiguation. In *Proceedings of the 2015 Conference on Empirical Methods on Natural Language Processing*, pages 726–736.

Elia Bruni, Nam-Khanh Tran, and Marco Baroni. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research (JAIR)*, 49(1-47).

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *Proceedings of the 2014 Conference on Empirical Methods on Natural Language Processing*, pages 1025–1035.

Massimiliano Ciaramita and Yasemin Altun. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of the 2006 Conference on Empirical Methods on Natural Language Processing*, pages 594–602. Association for Computational Linguistics.

Massimiliano Ciaramita and Mark Johnson. 2003. Supersense tagging of unknown nouns in WordNet. In *Proceedings of the 2003 Conference on Empirical Methods on Natural Language Processing*, pages 168–175. Association for Computational Linguistics.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning*, 75(3):297–325.

Li Dong, Furu Wei, Shujie Liu, Ming Zhou, and Ke Xu. 2015. A statistical parsing framework for sentiment classification. *Computational Linguistics*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. *arXiv preprint*, arXiv:1605.02276.

Christiane Fellbaum. 1990. English verbs as a semantic net. *International Journal of Lexicography*, 3(4):278–301.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Lucie Flekova and Iryna Gurevych. 2015. Personality profiling of fictional characters using sense-level links between lexical resources. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1805–1816.

Anatole Gershman, Yulia Tsvetkov, Leonid Boytsov, Eric Nyberg, and Chris Dyer. 2014. Metaphor detection with cross-lingual model transfer. In *Proceedings of the 52nd annual meeting on Association for Computational Linguistics*.

Josu Goikoetxea, Aitor Soroa, Eneko Agirre, and Basque Country Donostia. 2015. Random walks and neural network language models on knowledge bases. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 1434–1439.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

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

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2015. Sensembed: learning sense embeddings for word and relational similarity. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 95–105.

Rubén Izquierdo, Armando Suárez, and German Rigau. 2009. An empirical study on class-based word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 389–397. Association for Computational Linguistics.

Anders Johannsen, Dirk Hovy, Héctor Martınez Alonso, Barbara Plank, and Anders Søgaard. 2014. More or less supervised supersense tagging of Twitter. *Proceedings of the 3rd Joint Conference on Lexical and Computational Semantics*, pages 1–11.

Rie Johnson and Tong Zhang. 2014. Effective use of word order for text categorization with convolutional neural networks. *arXiv preprint arXiv:1412.1058*.

Massimiliano Ciaramita Jordi Atserias, Hugo Zaragoza and Giuseppe Attardi. 2008. Semantically annotated snapshot of the english wikipedia. In Bente Maegaard Joseph Mariani Jan Odijk Stelios Piperidis Daniel Tapias Nicoletta Calzolari (Conference Chair), Khalid Choukri, editor, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may. European Language Resources Association (ELRA).

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Egoitz Laparra and German Rigau. 2013. Impar: A deterministic algorithm for implicit semantic role labelling. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 1180–1189.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Jiwei Li and Dan Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1722–1732, Lisbon, Portugal, September. Association for Computational Linguistics.

Jiwei Li, Thang Luong, Dan Jurafsky, and Eduard Hovy. 2015. When are tree structures necessary for deep learning of representations? In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2304–2314, Lisbon, Portugal, September. Association for Computational Linguistics.

Bernardo Magnini and Gabriela Cavaglia. 2000. Integrating subject field codes into WordNet. In *Proceedings of the Third International Conference on Language Resources and Evaluation*.

Mónica Marrero, Sonia Sánchez-Cuadrado, Jorge Morato Lara, and George Andreadakis. 2009. Evaluation of named entity extraction systems. *Advances in Computational Linguistics, Research in Computing Science*, 41:47–58.

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

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *North American Chapter of the Association for Computational Linguistics - Human Language Technologies*, pages 746–751.

George A. Miller and Walter G. Charles. 1991. Contextual correlates of semantic similarity. *Language and Cognitive Processes*, 6(1):1–28.

George A Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G Thomas. 1994. Using a semantic concordance for sense identification. In *Proceedings of the workshop on Human Language Technology*, pages 240–243. Association for Computational Linguistics.

George A Miller. 1990. Nouns in WordNet: a lexical inheritance system. *International Journal of Lexicography*, 3(4):245–264.

George A Miller. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Computing Surveys (CSUR)*, 41(2):10.

Alok Ranjan Pal and Diganta Saha. 2015. Word sense disambiguation: a survey. *arXiv preprint arXiv:1508.01346*.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings*

of the 52nd Annual Meeting of the Association for Computational Linguistics, page 271. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 115–124. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing*, volume 14, pages 1532–1543.

Davide Picca, Alfio Massimiliano Gliozzo, and Massimiliano Ciaramita. 2008. Supersense tagger for italian. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*. Citeseer.

Barbara Plank, Anders Johannsen, and Anders Søgaard. 2014. Importance weighting and unsupervised domain adaptation of pos taggers: a negative result. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 968–973, Doha, Qatar, October. Association for Computational Linguistics.

Likun Qiu, Yunfang Wu, and Yanqiu Shao. 2011. Combining contextual and structural information for supersense tagging of Chinese unknown words. In *Computational Linguistics and Intelligent Text Processing*, pages 15–28. Springer.

Lance Ramshaw and Mitch Marcus. 1995. Text chunking using transformation-based learning. In *Third Workshop on Very Large Corpora*, pages 82–94.

Samuel Reese, Gemma Boleda Torrent, Montserrat Cuadros Oller, Lluís Padró, and German Rigau Claramunt. 2010. Word-sense disambiguated multilingual Wikipedia corpus. In *Proceedings of the 7th International Conference on Language Resources and Evaluation*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA.

Vassiliki Rentoumi, George Giannakopoulos, Vangelis Karkaletsis, and George A Vouros. 2009. Sentiment analysis of figurative language using a word sense disambiguation approach. In *Proceedings of the Conference on Recent Advances in Natural Language Processing*, pages 370–375.

Alan Ritter, Sam Clark, Oren Etzioni, et al. 2011. Named entity recognition in tweets: an experimental study. In *Proceedings of the 2011 Conference on Empirical Methods on Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics.

Sascha Rothe and Hinrich Schütze. 2015. Autoextend: Extending word embeddings to embeddings for synsets and lexemes. In *Proceedings of the 53rd Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1793–1803, Beijing, China, July. Association for Computational Linguistics.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8(10):627–633, October.

Stefan Rüd, Massimiliano Ciaramita, Jens Müller, and Hinrich Schütze. 2011. Piggyback: Using search engines for robust cross-domain named entity recognition. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 965–975. Association for Computational Linguistics.

Tara N Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 4580–4584. IEEE.

Nathan Schneider and Noah A Smith. 2015. A corpus and model integrating multiword expressions and supersenses.

Nathan Schneider, Behrang Mohit, Kemal Oflazer, and Noah A Smith. 2012. Coarse lexical semantic annotation with supersenses: an arabic case study. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 253–258. Association for Computational Linguistics.

Nathan Schneider, Behrang Mohit, Chris Dyer, Kemal Olflazer, and Noah A Smith. 2013. Supersense tagging for arabic: the mt-in-the-middle attack. Association for Computational Linguistics.

Federico Scozzafava, Alessandro Raganato, Andrea Moro, and Roberto Navigli. 2015. Automatic identification and disambiguation of concepts and named entities in the multilingual wikipedia. In *AI* IA 2015 Advances in Artificial Intelligence*, pages 357–366. Springer.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning semantic textual similarity with structural representations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 714–718.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods on Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1201–1211. Association for Computational Linguistics.

Chiraag Sumanth and Diana Inkpen. 2015. How much does word sense disambiguation help in sentiment analysis of micropost data? In *6TH Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA 2015)*, page 115.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383.

Yulia Tsvetkov, Anatole Gershman, and Elena Mukomel. 2013. Cross-lingual metaphor detection using common semantic features. In *The First Workshop on Metaphor in NLP*, page 45.

Yulia Tsvetkov, Nathan Schneider, Dirk Hovy, Archna Bhatia, Manaal Faruqui, and Chris Dyer. 2014. Augmenting english adjective senses with supersenses.

Yulia Tsvetkov, Manaal Faruqui, Wang Ling, Guillaume Lample, and Chris Dyer. 2015. Evaluation of word vector representations by subspace alignment. In *Proceedings of the 2015 Conference on Empirical Methods on Natural Language Processing*. Association for Computational Linguistics.

Peter D Turney, Yair Neuman, Dan Assaf, and Yohai Cohen. 2011. Literal and metaphorical sense identification through concrete and abstract context. In *Proceedings of the 2011 Conference on the Empirical Methods in Natural Language Processing*, pages 680–690.

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

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *Proceedings of the 30th International Conference on Machine Learning*, pages 118–126.

Michael Wilson. 1988. MRC psycholinguistic database: Machine-usable dictionary, version 2.00. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–10.

Mo Yu and Mark Dredze. 2014. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics*, pages 545–550.

Ye Zhang, Stephen Roller, and Byron Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. *arXiv preprint arXiv:1603.00968*.

Efficient techniques for parsing with tree automata

Jonas Groschwitz and **Alexander Koller**
Department of Linguistics
University of Potsdam
Germany
groschwi|akoller@uni-potsdam.de

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

Abstract

Parsing for a wide variety of grammar formalisms can be performed by intersecting finite tree automata. However, naive implementations of parsing by intersection are very inefficient. We present techniques that speed up tree-automata-based parsing, to the point that it becomes practically feasible on realistic data when applied to context-free, TAG, and graph parsing. For graph parsing, we obtain the best runtimes in the literature.

1 Introduction

Grammar formalisms that go beyond context-free grammars have recently enjoyed renewed attention throughout computational linguistics. Classical grammar formalisms such as TAG (Joshi and Schabes, 1997) and CCG (Steedman, 2001) have been equipped with expressive statistical models, and high-performance parsers have become available (Clark and Curran, 2007; Lewis and Steedman, 2014; Kallmeyer and Maier, 2013). Synchronous grammar formalisms such as synchronous context-free grammars (Chiang, 2007) and tree-to-string transducers (Galley et al., 2004; Graehl et al., 2008; Seemann et al., 2015) are being used as models that incorporate syntactic information in statistical machine translation. Synchronous string-to-tree (Wong and Mooney, 2006) and string-to-graph grammars (Chiang et al., 2013) have been applied to semantic parsing; and so forth.

Each of these grammar formalisms requires its users to develop new algorithms for parsing and training. This comes with challenges that are both practical and theoretical. From a theoretical perspective, many of these algorithms are basically the same, in that they rest upon a CKY-style parsing algorithm which recursively explores substructures of the input object and assigns them nonterminal symbols, but their exact relationship is rarely made explicit. On the practical side, this parsing algorithm and its extensions (e.g. to EM training) have to be implemented and optimized from scratch for each new grammar formalism. Thus, development time is spent on reinventing wheels that are slightly different from previous ones, and the resulting implementations still tend to underperform.

Koller and Kuhlmann (2011) introduced *Interpreted Regular Tree Grammars (IRTGs)* in order to address this situation. An IRTG represents a language by describing a regular language of derivation trees, each of which is mapped to a term over some algebra and evaluated there. Grammars from a wide range of monolingual and synchronous formalisms can be mapped into IRTGs by using different algebras: Context-free and tree-adjoining grammars use string algebras of different kinds, graph grammars can be captured by using graph algebras, and so on. In addition, IRTGs come with a universal parsing algorithm based on closure results for tree automata. Implementing and optimizing this parsing algorithm once, one could apply it to all grammar formalisms that can be mapped to IRTG. However, while Koller and Kuhlmann show that asymptotically optimal parsing is possible in theory, it is non-trivial to implement their algorithm optimally.

In this paper, we introduce practical algorithms for the two key operations underlying IRTG parsing: computing the intersection of two tree automata and applying an inverse tree homomorphism to a tree automaton. After defining IRTGs (Section 2), we will first illustrate that a naive bottom-up implementation of the intersection algorithm yields asymptotic parsing complexities that are too high (Section 3). We will then

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2042–2051,
Berlin, Germany, August 7-12, 2016.

show how the parsing complexity can be improved by combining algebra-specific index data structures with a generic parsing algorithm (Section 4), and by replacing bottom-up with top-down queries (Section 5). In contrast to the naive algorithm, both of these methods achieve the expected asymptotic complexities, e.g. $O(n^3)$ for context-free parsing, $O(n^6)$ for TAG parsing, etc. Furthermore, an evaluation with realistic grammars shows that our algorithms improve practical parsing times with IRTG grammars encoding context-free grammars, tree-adjoining grammars, and graph grammars by orders of magnitude (Section 6). Thus our algorithms make IRTG parsing practically feasible for the first time; for graph parsing, we obtain the fastest reported runtimes.

2 Interpreted Regular Tree Grammars

We will first define IRTGs and explain how the universal parsing algorithm for IRTGs works.

2.1 Formal foundations

First, we introduce some fundamental theoretical concepts and notation.

A *signature* Σ is a finite set of symbols $r, f, \ldots$, each of which has an *arity* $\mathsf{ar}(r) \geq 0$. A *tree* t over the signature Σ is a term of the form $r(t_1, \ldots, t_n)$, where the t_i are trees and $r \in \Sigma$ has arity n. We identify the nodes of t by their Gorn addresses, i.e. paths $\pi \in \mathbb{N}^*$ from the root to the node, and write $t(\pi)$ for the label of π. We write T_Σ for the set of all trees over Σ, and $T_\Sigma(\mathcal{X}_k)$ for the trees in which each node either has a label from Σ, or is a leaf labeled with one of the variables $\{x_1, \ldots, x_k\}$.

A *(linear, nondeleting) tree homomorphism* h from a signature Σ to a signature Δ is a mapping $h : T_\Sigma \to T_\Delta$. It is defined by specifying, for each symbol $r \in \Sigma$ of arity k, a term $h(r) \in T_\Delta(\mathcal{X}_k)$ in which each variable occurs exactly once. This symbol-wise mapping is lifted to entire trees by letting $h(r(t_1, \ldots, t_k)) = h(r)[h(t_1), \ldots, h(t_k)]$, i.e. by replacing the variable x_i in $h(r)$ by the recursively computed value $h(t_i)$.

Let Δ be a signature. A Δ-*algebra* $\mathcal{A}$ consists of a nonempty set A, called the *domain*, and for each symbol $f \in \Delta$ with arity k, a function $f^{\mathcal{A}} : A^k \to A$, the *operation* associated with f. We can *evaluate* any term $t \in T_\Delta$ to a value $t^{\mathcal{A}} \in A$, by evaluating the operation symbols bottom-up. In this paper, we will be particularly interested in the *string algebra* E^* over the finite alphabet E. Its domain is the set of all strings over E. For each symbol $a \in E$, it has a nullary operation symbol a with $\mathsf{a}^{E^*} = a$. It also has a single binary operation symbol $*$, such that $*^{E^*}(w_1, w_2)$ is the concatenation of the strings w_1 and w_2. Thus the term $*(\mathsf{John}, *(\mathsf{walks}, *(\mathsf{on}, \mathsf{Mars})))$ in Fig. 2b evaluates to the string "John walks on Mars".

automaton rule	homomorphism
$\mathsf{S} \to \mathsf{r}_1(\mathsf{NP}, \mathsf{VP})$	$*(x_1, x_2)$
$\mathsf{NP} \to \mathsf{r}_2$	John
$\mathsf{VP} \to \mathsf{r}_3$	walks
$\mathsf{VP} \to \mathsf{r}_4(\mathsf{VP}, \mathsf{NP})$	$*(x_1, *(\mathsf{on}, x_2))$
$\mathsf{NP} \to \mathsf{r}_5$	Mars

Figure 1: An example IRTG.

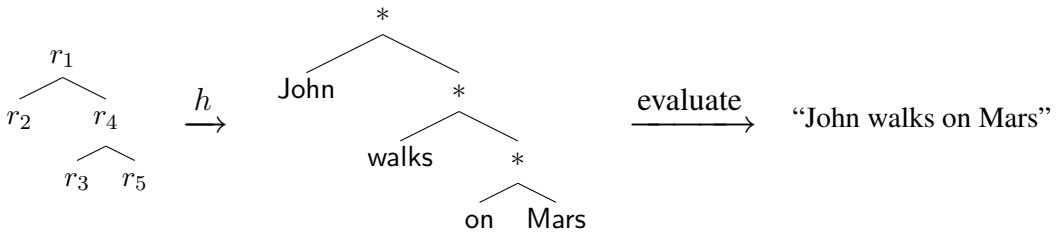
A *finite tree automaton* M over the signature Σ is a structure $M = (\Sigma, Q, R, X_F)$, where Q is a finite set of *states* and $X_F \in Q$ is a *final state*. R is a finite set of *transition rules* of the form $X \to r(X_1, \ldots, X_k)$, where the *terminal symbol* $r \in \Sigma$ is of arity k and $X, X_1, \ldots, X_k \in Q$. A tree automaton can *run* non-deterministically on a tree $t \in T_\Sigma$ by assigning states to the nodes of t bottom-up. If we have $t = r(t_1, \ldots, t_n)$ and M can assign the state X_i to each t_i, written $X_i \to^* t_i$, then we also have $X \to^* t$. We say that M *accepts* t if $X_F \to^* t$, and define the *language* $L(M) \subseteq T_\Sigma$ of M as the (possibly infinite) set of all trees that M accepts. An example of a tree automaton (with states S, NP, etc.) is shown in the "automaton rule" column of Fig. 1. It accepts, among others, the tree τ_1 in Fig. 2a.

Tree automata can be defined top-down or bottom-up, and are equivalent to *regular tree grammars*. The languages that can be accepted by finite tree automata are called the *regular tree languages*. See e.g. Comon et al. (2008) for details.

2.2 Interpreted regular tree grammars

We can combine tree automata, homomorphisms, and algebras into grammars that can describe languages of arbitrary objects, as well as relations between such objects – in a way that inherits many technical properties from context-free grammars, while extending the expressive capacity.

An *interpreted regular tree grammar* (IRTG, Koller and Kuhlmann (2011)) $\mathcal{G} = (M, (h_1, \mathcal{A}_1), \ldots, (h_n, \mathcal{A}_n))$ consists of a tree automaton M over some signature Σ, together with an arbitrary number n of *inter-*



(a) Tree τ_1. (b) Term $h(\tau_1)$. (c) $h(\tau_1)$ evaluated in E^*.

Figure 2: The tree τ_1, evaluated by the homomorphism h and the algebra E^*

pretations $(h_i, \mathcal{A}_i)$, where each $\mathcal{A}_i$ is an algebra over some signature Δ_i and each h_i is a tree homomorphism from Σ to Δ_i. The automaton M describes a language $L(M)$ of *derivation trees* which represent abstract syntactic structures. Each derivation tree τ is then interpreted n ways: we map it to a term $h_i(\tau) \in T_{\Delta_i}$, and then we evaluate $h_i(\tau)$ to a value $a_i = h_i(\tau)^{\mathcal{A}_i} \in A_i$ of the algebra $\mathcal{A}_i$. Thus, the IRTG $\mathcal{G}$ defines a language $L(\mathcal{G}) = \{(h_1(\tau)^{\mathcal{A}_1}, \ldots, h_n(\tau)^{\mathcal{A}_n}) \mid \tau \in L(M)\}$, which is an n-place relation between the domains of the algebras.

Consider the IRTG $\mathcal{G}$ shown in Fig. 1. The "automaton rule" column indicates the five rules of M; the final state is S. We already saw the derivation tree $\tau_1 \in L(M)$. $\mathcal{G}$ has a single interpretation, into a string algebra E^*, and with a homomorphism that is specified by the "homomorphism" column; for instance, $h(\mathsf{r}_1) = *(x_1, x_2)$ and $h(\mathsf{r}_2) = \mathsf{John}$. Applying this homomorphism to τ_1, we obtain the term $h(\tau_1)$ in Fig. 2b. As we saw earlier, this term evaluates in the string algebra to the string "John walks on Mars" (Fig. 2c). Thus this string is an element of $L(\mathcal{G})$.

We assume that no two rules of M use the same terminal symbol; this is generally not required in tree automata, but every IRTG can be brought into this convenient form. Furthermore, we focus (but only for simplicity of presentation) on IRTGs that use a single string-algebra interpretation, as in Fig. 1. Such grammars capture context-free grammars. However, IRTGs can capture a wide variety of grammar formalisms by using different algebras. For instance, an interpretation that uses a TAG string algebra (or TAG derived-tree algebra) models a tree-adjoining grammar (Koller and Kuhlmann, 2012), and an interpretation into an s-graph algebra models a hyperedge replacement graph grammar (HRG, Groschwitz et al. (2015)). By using multiple algebras, IRTGs can also represent synchronous grammars and (bottom-up) tree-to-tree and tree-to-string transducers. In general, any grammar formalism whose grammars describe derivations in terms of a finite set of states can typically be converted into IRTG.

2.3 Parsing IRTGs

Koller and Kuhlmann (2011) present a uniform parsing algorithm for IRTGs based on tree automata. The (monolingual) *parsing problem* of IRTG consists in determining, for an IRTG $\mathcal{G}$ and an input object $a \in A$, a representation of the set $\mathsf{parses}(a) = \{\tau \in L(M) \mid h(\tau)^{\mathcal{A}} = a\}$, i.e. of the derivation trees that are grammatically correct and are mapped to a by the interpretation. In the example, we have $\mathsf{parses}(\text{"John walks on Mars"}) = \{\tau_1\}$, where τ_1 is as above. In general, $\mathsf{parses}(a)$ may be infinite, and thus we aim to represent it using a tree automaton Ch_a with $L(\mathrm{Ch}_a) = \mathsf{parses}(a)$, the *parse chart* of a.

We can compute Ch_a as follows. First, observe that $\mathsf{parses}(a) = L(M) \cap h^{-1}(\mathsf{terms}(a))$, where $h^{-1}(L) = \{\tau \in T_\Sigma \mid h(\tau) \in L\}$ (the *inverse homomorphic image*, or *invhom*, of L) and $\mathsf{terms}(a) = \{t \in T_\Delta \mid t^{\mathcal{A}} = a\}$, i.e. the set of all terms that evaluate to a. Now assume that the algebra $\mathcal{A}$ is *regularly decomposable*, which means that every $a \in A$ has a *decomposition automaton* D_a, i.e. there is a tree automaton D_a such that $L(D_a) = \mathsf{terms}(a)$. Because regular tree languages are closed under invhom and intersection, we can then compute a tree automaton Ch_a by intersecting M with the invhom of D_a.

To illustrate the IRTG parsing algorithm, let us compute a chart for the sentence $s =$ "John walks on Mars" with the example grammar $\mathcal{G}$ of Fig. 1. The states of the decomposition automaton D_s are spans $[i, k]$ of s; the final state is $X_F = [1, 5]$. The automaton contains fourteen rules, including the ones shown in Fig. 3a.

$[1,5] \to *([1,2],[2,5])$
$[2,5] \to *([2,3],[3,5])$
$[3,5] \to *([3,4],[4,5])$
$[3,4] \to \mathsf{on}$
$[4,5] \to \mathsf{Mars}$

(a) Some rules of D_s.

$[1,5] \to \mathsf{r}_1([1,2],[2,5])$
$[2,5] \to \mathsf{r}_4([2,3],[4,5])$
$[2,4] \to \mathsf{r}_1([2,3],[3,4])$
$[1,2] \to \mathsf{r}_2$
$[2,3] \to \mathsf{r}_3$

(b) Some rules of $I = h^{-1}(D_s)$.

$\mathsf{S}[1,5] \to \mathsf{r}_1(\mathsf{NP}[1,2],\mathsf{VP}[2,5])$
$\mathsf{VP}[2,5] \to \mathsf{r}_4(\mathsf{VP}[2,3],\mathsf{NP}[4,5])$
$\mathsf{NP}[1,2] \to \mathsf{r}_2$
$\mathsf{VP}[2,3] \to \mathsf{r}_3$
$\mathsf{NP}[4,5] \to \mathsf{r}_5$

(c) The parse chart Ch_s.

Figure 3: Example rules for the sentence $s =$ "John walks on Mars"

Algorithm 1 Naive bottom-up intersection

```
1: initialize agenda with state pairs for constants
2: initialize P as empty
3: while agenda is not empty do
4:    T'X' ← pop(agenda)
5:    add T'X' to P
6:    for T''X'' ∈ P do
7:       for {T1X1, T2X2} = {T'X', T''X''} do
8:          for T → r(T1, T2) in M_L do
9:             for X → r(X1, X2) in M_R do
10:               store TX → r(T1X1, T2X2)
11:               add TX to agenda if new
```

We can then compute the invhom automaton I, such that $L(I) = h^{-1}(L(D_s))$. I uses the same states as D_s, but uses terminal symbols from Σ instead of Δ. Some rules of the invhom automaton I in the example are shown in Fig. 3b. Notice that I also contains rules that are not consistent with M, i.e. that would not occur in a grammatical parse of the sentence, such as $[2,4] \to \mathsf{r}_1([2,3],[3,4])$. Finally, the chart Ch_s is computed by intersecting M with I (see Fig. 3c). The states of Ch_s are pairs of states from M and states from I. It accepts τ_1, because $\tau_1 \in \mathsf{parses}(s)$. Observe the similarity to a traditional context-free parse chart.

3 Bottom-up intersection

Both the practical efficiency of this algorithm and its asymptotic complexity depend crucially on how we compute intersection and invhom. We illustrate this using an overly naive intersection algorithm as a strawman, and then analyze the problem to lay the foundations for the improved algorithms in Sections 4 and 5.

Let's say that we want to compute a tree automaton C for the intersection of a "left" automaton M_L and a "right" automaton M_R both over the same signature Σ. In the application to IRTG parsing, M_L is typically the derivation tree automaton (called M above) and M_R is the invhom of a decomposition automaton. As in the product construction for finite string automata, the states of C will be pairs TX of states T of M_L and states X of M_R, and the rules of C will all have the form $TX \to r(T_1X_1, \ldots, T_nX_n)$, where $T \to r(T_1, \ldots, T_n)$ is a rule in M_L, and $X \to r(X_1, \ldots, X_n)$ is a rule in M_R.

3.1 Naive intersection

A naive bottom-up algorithm is shown in Alg. 1.[1] This algorithm maintains an agenda of state pairs that have been discovered, but not explored as children of bottom-up rule applications; and a chart-like set P of all state pairs that have ever been popped off the agenda. The algorithm maintains the invariant that if TX is on the agenda or in P, then T and X are *partners* (written $T \approx X$), i.e. there is a tree $t \in T_\Sigma$ such that $T \to^* t$ in M_L and $X \to^* t$ in M_R.

The agenda is initialized with all state pairs TX, for which M_L has a rule $T \to r$ and M_R has a rule $X \to r$ for some nullary symbol $r \in \Sigma$. Then, while there are state pairs left on the agenda, Alg. 1 pops a state pair $T'X'$ off the agenda and adds it to P; iterates over all state pairs $T''X''$ in P; and queries M_L and M_R bottom-up for rules in which these states appear as children.[2] The iteration in line 7 allows T' and X' to be either left or right children in these rules. For each pair of left and right rules, the rules are combined into a rule of C, and the pair of the parent states T and X is added to the agenda.

This naive intersection algorithm yields an asymptotic complexity for IRTG parsing that is higher than expected. Assume, for example,

[1] We assume binary symbols for simplicity; all algorithms generalize to arbitrary arities.

[2] For the invhom automaton this can be done by substituting the variables in the homomorphic image $h(r)$ with the corresponding states X' and X'', and running the decomposition automaton on the resulting tree.

Algorithm 2 Bottom-up intersection with BU

```
1: initialize agenda with state pairs for constants
2: generate new S_r = S(M_R, r) for every r ∈ Σ
3: while agenda is not empty do
4:    T'X' ← pop(agenda)
5:    for T → r(T_1, T_2) in M_L s.t. T_i = T' do
6:       for X → r(X_1, X_2) ∈ BU(S_r, i, X') do
7:          store rule TX → r(T_1X_1, T_2X_2)
8:          add TX to agenda if new
```

that we are parsing with an IRTG encoding of a context-free grammar, i.e. with a string algebra (as in Fig. 1). Then the states of M_R are spans $[i, k]$, i.e. M_R has $O(n^2)$ states. Once line 4 has picked a span $X' = [i, j]$, line 6 iterates over all spans $X'' = [k, l]$ that have been discovered so far – including ones in which $j \neq k$ and $i \neq l$. Thus the bottom-up lookup in line 9 is executed $O(n^4)$ times, most of which will yield no rules. The overall runtime of Alg. 1 is therefore higher than the asymptotic runtime of $O(n^3)$ expected for context-free parsing. Similar problems arise for other algebras; for instance, the runtime of Alg. 1 for TAG parsing is $O(n^8)$ rather than $O(n^6)$.

3.2 Indexing

In context-free parsing algorithms, such as CKY or Earley, this issue is addressed through appropriate index datastructures, which organize P such that the lookup in line 5 only returns state pairs where X'' is of the form $[j, k]$ or $[k, i]$. This reduces the runtime to cubic.

The idea of obtaining optimal asymptotic complexities in IRTG parsing through appropriate indexing was already mentioned from a theoretical perspective by Koller and Kuhlmann (2011). However, they assumed an optimal indexing data structure as given. In practice, indexing requires algebra-specific knowledge about X'': A CKY-style index only works if we assume that the states of the decomposition automaton are spans (this is not the case in other algebras), and that the only binary operation in the string algebra is $*$, which composes spans in a certain way. Furthermore, in IRTG parsing the rules of the invhom automaton do not directly correspond to algebra operations, but to *terms* of operations, which further complicates indexing.

In this paper, we incorporate indexing into the intersection algorithm through *sibling-finders*. A sibling-finder $S = S(M, r)$ for an automaton M and a label r in M's signature is a data structure that supports a single operation, $\mathsf{BU}(S, i, X')$. We require that a call to $\mathsf{BU}(S, i, X')$ returns the set of rules $X \to r(X_1, \ldots, X_n)$ of M such that X' is the i-th child state, and for every $j \neq i$, X_j must be a state for which we previously called $\mathsf{BU}(S, j, X_j)$. Thus a sibling-finder performs a bottom-up rule lookup, changing its state after each call by caching the state and position.

Assume that we have sibling-finders for M_R. Then we can modify the naive Alg. 1 to the closely related algorithm shown as Alg. 2. This algorithm maintains the same agenda as Alg. 1, but instead of iterating over all explored partner states $T''X''$, it first iterates over all rules in M_L that have T' as a child (line 5). In line 6, Alg. 2 then queries M_R sibling-finders – we maintain one for each rule label – for right rules with matching rule label and child positions. Note that because there is only one rule with label r in M_L, the sibling-finders implicitly keep track of the partners of T_2 we have seen so far. Thus they play the role of a more structured variant of P.

There are a number of ways in which sibling-finders can be implemented. First, they could simply maintain sets $\mathsf{chi}(S_r, i)$ where a call to $\mathsf{BU}(S_r, i, X')$ first adds X' to $\mathsf{chi}(S_r, i)$. The query can then iterate over the set $\mathsf{chi}(S_r, 3-i)$, to check for each state X'' in that set whether M_R actually contains a rule with terminal symbol r and children X' and X'' (in the right order). This essentially reimplements the behavior of Alg. 1, and comes with the same complexity issues.

Second, we could theoretically iterate over all rules of M_R to implement the sibling finders via a bottom-up index (e.g., a trie) that supports efficient BU queries. However, in IRTG parsing M_R is the invhom of a decomposition automaton. Because the decomposition automaton represents all the ways in which the input object can be built recursively out of smaller structures, including ones which will later be rejected by the grammar, such automata can be very large in practice. Thus we would like to work with a lazy representation of M_R and avoid iterating over all rules.

4 Efficient bottom-up lookup

Finally, we can exploit the fact that in IRTG parsing, M_R is the invhom of a decomposition automaton. Below, we first show how to define algebra-specific sibling-finders for decompo-

Algorithm 3 passUpwards(Y, π, i, r)

```
rules ← BU(S_{r,π}, i, Y)
if π = π'k ≠ ϵ then
    for X → f(X_1, ..., X_n) ∈ rules do
        passUpwards(X, π', k, r)
```

sition automata. Then we develop an algebra-independent way to generate invhom sibling-finders out of those for the decomposition automata. These can be plugged into Alg. 2 to achieve the expected parsing complexity.

4.1 Sibling-finders for decomposition automata

First, consider the special case of sibling-finders for a decomposition automaton D. The terminal symbols f of D are the operation symbols of an algebra. If we have information about the operations of this algebra, and how they operate on the states of D, a sibling-finder $S = S(D, f)$ can use indexing specific to the operation f to look up potential siblings, and only for them query D to answer $\mathsf{BU}(S, i, X)$

For instance, a sibling-finder for the '$*$' operation of the string algebra may store all states $[k, l]$ for $i = 1$ under the index l. Thus a lookup $\mathsf{BU}(S, 2, [l, m])$ can directly retrieve siblings from the l-bin, just as a traditional parse chart would. Spans which do not end at l are never considered. Different algebras require different index structures. For instance, sibling-finders for the string-wrapping operation in the TAG string algebra might retrieve all pairs of substrings $[k, l, m, o]$ that wrap around $[l, m]$ instead. Analogous data structures can be defined for the s-graph algebra.

4.2 Sibling-finders for invhom

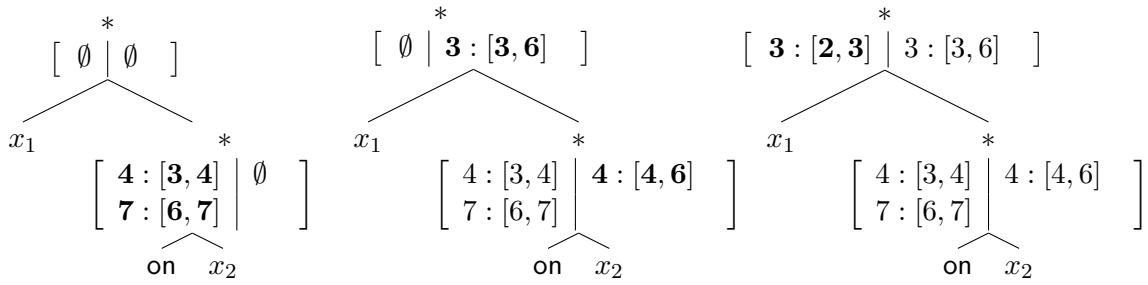
We can build upon the D-sibling-finders to construct sibling-finders for the invhom I of D. The basic idea is as follows. Consider the term $h(\mathsf{r}_1) = *(x_1, x_2)$ from Fig. 1. It contains a single operation symbol $*$ (plus variables); the homomorphism only replaces one symbol with another. Thus a sibling-finder $S(D, *)$ from the decomposition automaton can directly serve as a sibling-finder $S(I, \mathsf{r}_1)$. We only need to replace the $*$ label on the returned rules with r_1.

In general, the situation is more complicated, because $t = h(r)$ may be a complex term consisting of many algebra operations. In such a case, we construct a separate sibling-finder $S_{r,\pi} =$ new $S(D, t(\pi))$ for each node π with at least two children. For instance, consider the term $t = h(\mathsf{r}_4)$ in Fig. 1. It contains three nodes which are labeled by algebra operations, two of which are the concatenation. We decorate these with the sibling-finders $S_{\mathsf{r}_4,\epsilon}$ and $S_{\mathsf{r}_4,1}$. Each of these is a sibling-finder for the algebra's concatenation operation; but they may have different state because they received different queries.

We can then construct an invhom sibling-finder $S_r = S(I, r)$, which answers a query $\mathsf{BU}(S_r, i, X')$ in two steps. First, we substitute the variable x_i by the state X' and percolate it upward through t using the D-sibling-finders on the path from x_i to the root. If $\pi = \pi' k$ is the path to x_i, we do this by calling passUpwards(X', π', k, r), as defined in Alg. 3. If the local sibling-finder returns rules and we are not at the root yet, we recursively call passUpwards at the parent node π' with each parent state of these rules.

As we do this, we let each sibling-finder maintain the set of rules it found, indexed by their parent state. This allows us to perform the second step: we traverse t top-down from the root to extract the rules of the invhom automaton that answer the BU query. Recall that $\mathsf{BU}(S_r, i, X')$ should return only rules $X \to r(X_1, X_2)$ where $\mathsf{BU}(S_r, 3 - i, X_{3-i})$ was called before. Here, this is guaranteed by having distinct D-sibling-finders $S_{\pi,r}$ for every node π at every tree $h(r)$. A final detail is that before the first query to r, we initialize the sibling-finders by calling passUpwards for all the leaves that are labeled by constants.

This process is illustrated in Fig. 4, on the sibling-finder $S = S(I, \mathsf{r}_4)$ and the input string "John walks on a hill on Mars", parsed with a suitable extension of the IRTG in Fig. 1. The decomposition automaton can accept the word "on" from states $[3, 4]$ and $[6, 7]$, which during initialization are entered into position 1 of the lower D-sibling-finder $S_{\mathsf{r}_4,2}$, indexed by their end positions (a). Alg. 2 may then generate the query $\mathsf{BU}(S, 2, [4, 6])$. This enters $[4, 6]$ into the lower D-sibling-finder, and because there is a state with end position 4 on the left side of this sibling-finder, $\mathsf{BU}(S_{\mathsf{r}_4,2}, 2, [4, 6])$ returns a rule with parent $[3, 6]$. The parent is subsequently entered into the upper sibling-finder (b). Finally, the query $\mathsf{BU}(S, 1, [2, 3])$ enters $[2, 3]$ into the upper D-sibling-finder and discovers its sibling $[3, 6]$ (c). This yields a state $X = [2, 6]$ for the whole phrase



(a) After initialization. (b) After $\mathsf{BU}(S, 2, [4, 6])$. (c) After $\mathsf{BU}(S, 1, [2, 3])$.

Figure 4: Three stages of BU on $S(I, \mathsf{r}_4)$ for the sentence "John walks on a hill on Mars".

Algorithm 4 Top-down intersection

```
1: function expand(X):
2: if X ∉ visited then
3:     add X to visited
4:     for X → r(X1, X2) in M_R do
5:         call expand(X_i) for i = 1, 2
6:         for T→r(T1, T2) s.t. T_i ∈ prt(X_i) do
7:             store rule TX→r(T1X1, T2X2)
8:             add T to prt(X)
```

"walks on a hill". The top-down traversal of the sibling-finders reveals that this state is reached by combining $x_1 = [2, 3]$, for which this BU query asked, with $x_2 = [4, 6]$, and thus the BU query yields the rule $[2, 6] \to \mathsf{r}_4([2, 3], [4, 6])$. A subsequent query for $\mathsf{BU}(S, 2, [4, 8])$ would yield the rule $[2, 8] \to \mathsf{r}_4([2, 3], [4, 8])$, and so on.

The overall construction allows us to answer BU queries on invhom automata while making use of algebra-specific index structures. Given suitable index structures, the asymptotic complexity drops down to the expected levels, e.g. $O(n^3)$ for IRTGs using the string algebra, $O(n^6)$ for the TAG string algebra, and so on. This yields a practical algorithm that can be flexibly adapted to new algebras by implementing their sibling-finders.

5 Top-down intersection

Instead of investing into efficient bottom-up queries, we can also explore the use of top-down queries instead. These ask for all rules with parent state X and terminal symbol r. Such queries completely avoid the problem of finding siblings in M_R. An invhom automaton can answer top-down queries for r efficiently by running the decomposition automaton top-down on $h(r)$, collecting child states at the variable nodes. For instance, if we query I from Section 2 top-down for rules with the parent $[1, 5]$ and symbol r_1, it will enumerate the rules $[1, 5] \to \mathsf{r}_1([1, 2], [2, 5])$, $[1, 5] \to \mathsf{r}_1([1, 3], [3, 5])$, and $[1, 5] \to \mathsf{r}_1([1, 4], [4, 5])$, without ever considering any other combination of child states.

This is the idea underlying the intersection algorithm in Alg. 4. It recursively visits states X of M_R, collecting for each X a set $\mathsf{prt}(X)$ of states T of M_L such that $T \approx X$. Line 5 ensures that the prt sets have been computed for both child states of the rule $X \to r(X_1, X_2)$. Line 6 then does a bottom-up lookup of M_L rules with the terminal symbol r and with child states that are partners of X_1 and X_2. Applied to our running example, Alg. 4 parses "John walks on Mars" by recursive calls on $\mathrm{expand}([1, 5])$ and $\mathrm{expand}([2, 5])$, following the rules of I top-down. Recursive calls for $[2, 3]$ and $[4, 5]$ establish $\mathsf{VP} \in \mathsf{prt}([2, 3])$ and $\mathsf{NP} \in \mathsf{prt}([4, 5])$, which enables the recursive call for $[2, 5]$ to apply r_4 in line 6 and consequently add VP to $\mathsf{prt}([2, 5])$ in line 8.

The algorithm mixes top-down queries to M_R with bottom-up queries to M_L. Line 6 implements the core idea of the CKY parser, in that it performs bottom-up queries on sets of nonterminals that are partners of adjacent spans – but generalized to arbitrary IRTGs instead of just the string algebra. The top-down query to M_R in line 4 is bounded by the number of rules that actually exist in M_R, which is $O(n^3)$ for the string algebra, $O(n^6)$ in the TAG string algebra, and $O\left(n^s \cdot 3^{ds} ds\right)$ for graphs of degree d and treewidth $s - 1$ in the graph algebra. Thus Alg. 4 achieves the same asymptotic complexity as native parsing algorithms.

Condensed top-down intersection. One weakness of Alg. 4 is that it iterates over all rules $X \to r(X_1, X_2)$ of M_R individually. This can be extremely wasteful when M_R is the invhom of a decomposition automaton, because it may contain

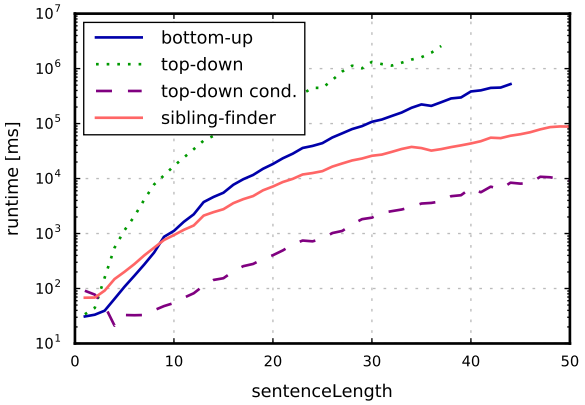


Figure 5: Runtimes for context-free parsing.

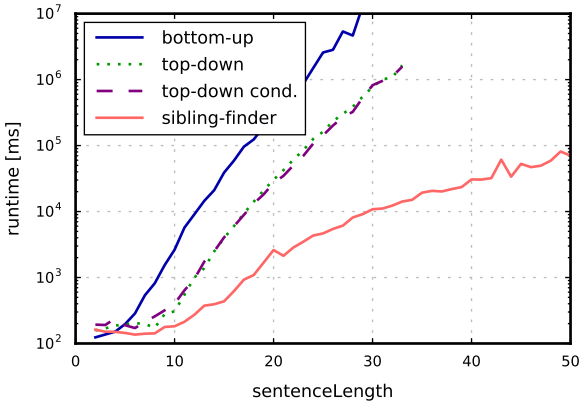


Figure 6: Runtimes for TAG parsing.

a great number of rules that have the same states and only differ in the terminal symbol r. For instance, when we encode a context-free grammar as an IRTG, for every rule r of the form $\mathsf{A} \to \mathsf{B}\ \mathsf{C}$ we have $h(\mathsf{r}) = *(x_1, x_2)$. The rules of the invhom automaton are the same for all terminal symbols r with the same term $h(\mathsf{r})$. But Alg. 4 iterates over rules r and not over different terms $h(\mathsf{r})$, repeating the exact same computation for every binary rule of the context-free grammar.

To solve this, we define *condensed* tree automata, which have rules of the form $X \to \rho(X_1, \ldots, X_n)$, where $\rho \subseteq \Sigma$ is a nonempty set of symbols with arity n. A condensed automaton represents the tree automaton which for all condensed rules $X \to \rho(X_1, \ldots, X_n)$ and all $r \in \rho$ has the rule $X \to r(X_1, \ldots, X_n)$. It is straightforward to represent an invhom automaton as a condensed condensed automaton, by determining for each distinct homomorphic image t the set $\rho_t = \{r_1, \ldots, r_k\}$ of symbols with $h(r_i) = t$.

We can modify Alg. 4 to iterate over condensed rules in line 4, and to iterate in line 6 over the rules $T \to r(T_1, T_2)$ for which $T_i \in \mathsf{prt}(X_i)$ and $r \in \rho$. This bottom-up query to M_L can be answered efficiently from an appropriate index on the rules of M_L. Altogether, this condensed intersection algorithm can be dramatically faster than the original version, if the grammar contains many symbols with the same homomorphic image.

6 Evaluation

We compare the runtime performance of the proposed algorithms on practical grammars and inputs, from three very different grammar formalisms: context-free grammars, TAG, and HRG graph grammars. In each setting, we measure the runtime of four algorithms: the naive bottom-up baseline of Section 3; the sibling-finder algorithm from Section 4; and the non-condensed and the condensed version of the top-down algorithm from Section 5. The results are shown in Figures 5, 6 and 7. We measure the runtimes for computing the complete chart, and plot the geometric mean of runtimes for each input size on a log scale.

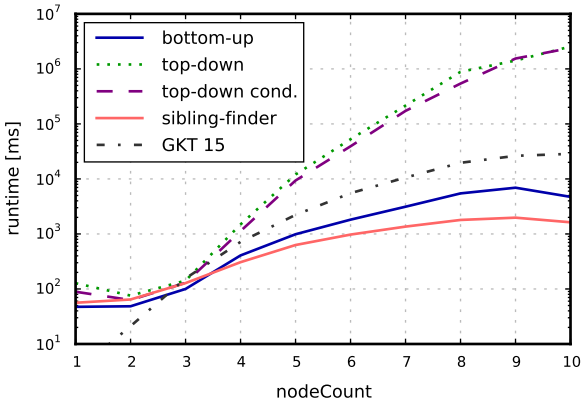


Figure 7: Runtimes for graph parsing.

We measured all runtimes on an Intel Xeon E7-8857 CPU at 3 GHz using Java 8. The JVM was warmed up before the measurements. The parser filtered each grammar automatically, removing all rules whose homomorphic image contained a constant that could not be used for a given input (e.g., a word that did not occur in the sentence).

PCFG. We extracted a binarized context-free grammar with 6929 rules from Section 00 of the Penn Treebank, and parsed the sentences of Section 00 with it. The homorphism in the corresponding IRTG assigns every terminal symbol a constant or the term $*(x_1, x_2)$, as in Fig. 1. As a consequence, the condensed automaton optimization from Section 5 outperforms all other algo-

rithms, achieving a 100x speedup over the naive bottom-up algorithm when it was cancelled.

TAG. We also extracted a tree-adjoining grammar from Section 00 of the PTB as described by Chen and Vijay-Shanker (2000), converted it to an IRTG as described by Koller and Kuhlmann (2012), and binarized it, yielding an IRTG with 26652 rules. Each term $h(r)$ in this grammar represents an entire TAG elementary tree, which means the terms are much more complex than for the PCFG and there are much fewer terminal symbols with the same homomorphic term. As a consequence, condensing the invhom is much less helpful. However, the sibling-finder algorithm excels at maintaining state information within each elementary tree, yielding a 1000x speedup over the naive bottom-up algorithm when it was cancelled.

Graphs. Finally, we parsed a corpus of graphs instead of strings, using the 13681-rule graph grammar of Groschwitz et al. (2015) to parse the 1258 graphs with up to 10 nodes from the "Little Prince" AMR-Bank (Banarescu et al., 2013). The top-down algorithms are slow in this experiment, confirming Groschwitz et al.'s findings. Again, the sibling-finder algorithm outperforms all other algorithms. Note that Groschwitz et al.'s parser ("GKT 15" in Fig. 7) shares much code with our system. It uses the same decomposition automata, but a less mature version of the sibling-finder method which fully computes the invhom automaton. Our new system achieves a 9x speedup for parsing the whole corpus, compared to GKT 15.

7 Related Work

Describing parsing algorithms at a high level of abstraction has a long tradition in computational linguistics, e.g. in deductive parsing with parsing schemata (Shieber et al., 1995). A key challenge under this view is to index chart entries so they can be retrieved efficiently, which parallels the situation in automata intersection discussed here. Gómez-Rodríguez et al. (2009) present an algorithm that automatically establishes index structures that guarantee optimal asymptotic runtime, but also requires algebra-specific extensions for grammar formalisms that go beyond context-free string grammars.

Efficient parsing has also been studied in other generalized grammar formalisms beyond IRTG. Kanazawa (to appear) shows how the parsing problem of Abstract Categorial Grammars (de Groote, 2001) can be translated into Datalog, which enables the use of generic indexing strategies for Datalog to achieve optimal asymptotic complexity. Ranta (2004) discusses parsing for his Grammatical Framework formalism in terms of partial evaluation techniques from functional programming, which are related to the step-by-step evaluation of sibling-finders in Figure 4. Like the approach of Gomez-Rodriguez et al., these methods have not been evaluated for large-scale grammars and realistic evaluation data, which makes it hard to judge their relative practical merits.

Most work in the tree automata community has a theoretical slant, and there is less research on the efficient implementation of algorithms for tree automata than one would expect; Cleophas (2009) and Lengal et al. (2012) are notable exceptions. Even these tend to be motivated by applications such as specification and verification, where the tree automata are much smaller and much less ambiguous than in computational linguistics. This makes these systems hard to apply directly.

8 Conclusion

We have presented novel algorithms for computing the intersection and the inverse homomorphic image of finite tree automata. These can be used to implement a generic algorithm for IRTG parsing, and apply directly to any grammar formalism that can be represented as an IRTG. An evaluation on practical data from three different grammar formalisms shows consistent speed improvements of several orders of magnitude, and our graph parser has the fastest published runtimes.

A Java implementation of our algorithms is available as part of the Alto parser, `http://bitbucket.org/tclup/alto`.

We focused here purely on symbolic parsing, and on computing complete parse charts. In the presence of a probability model (e.g. for IRTG encodings of PCFGs), our algorithms could be made faster through the use of appropriate pruning techniques. It would also be interesting to combine the strengths of the condensed and sibling-finder algorithms for further efficiency gains.

Acknowledgments. We thank the anonymous reviewers for their comments. We are grateful to Johannes Gontrum for an early implementation of Alg. 4, and to Christoph Teichmann for many fruitful discussions. This work was supported by the DFG grant KO 2916/2-1.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the Linguistic Annotation Workshop (LAW VII-ID)*.

John Chen and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proceedings of IWPT*.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Stephen Clark and James Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Loek Cleophas. 2009. Forest FIRE and FIRE Wood: Tools for tree automata and tree algorithms. In *Proceedings of the Conference on Finite-State Methods and Natural Language Processing (FSMNLP)*.

Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. 2008. Tree automata techniques and applications. `http://tata.gforge.inria.fr/`.

Philippe de Groote. 2001. Towards abstract categorial grammars. In *Proceedings of the 39th ACL/10th EACL*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of HLT/NAACL*.

Carlos Gómez-Rodríguez, Jesús Vilares, and Miguel A. Alonso. 2009. A compiler for parsing schemata. *Software: Practice and Experience*, 39(5):441–470.

Jonathan Graehl, Kevin Knight, and Jonathan May. 2008. Training tree transducers. *Computational Linguistics*, 34(3).

Jonas Groschwitz, Alexander Koller, and Christoph Teichmann. 2015. Graph parsing with s-graph grammars. In *Proceedings of the 53rd ACL and 7th IJCNLP*.

Aravind K. Joshi and Yves Schabes. 1997. Tree-Adjoining Grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 69–123. Springer-Verlag, Berlin.

Laura Kallmeyer and Wolfgang Maier. 2013. Data-driven parsing using probabilistic linear context-free rewriting systems. *Computational Linguistics*, 39(1):87–119.

Makoto Kanazawa. to appear. Parsing and generation as datalog query evaluation. *IfCoLog Journal of Logics and Their Applications*.

Alexander Koller and Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th International Conference on Parsing Technologies (IWPT)*.

Alexander Koller and Marco Kuhlmann. 2012. Decomposing TAG algorithms using simple algebraizations. In *Proceedings of the 11th TAG+ Workshop*.

Ondrej Lengal, Jiri Simacek, and Tomas Vojnar. 2012. Vata: A library for efficient manipulation of non-deterministic tree automata. In C. Flanagan and B. König, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 18th International Conference, TACAS 2012*. Springer.

Mike Lewis and Mark Steedman. 2014. A* CCG parsing with a supertag-factored model. In *Proceedings of EMNLP*.

Aarne Ranta. 2004. Grammatical framework: A type-theoretical grammar formalism. *Journal of Functional Programming*, 14(2):145–189.

Nina Seemann, Fabienne Braune, and Andreas Maletti. 2015. String-to-tree multi bottom-up tree transducers. In *Proceedings of the 53rd ACL and 7th IJCNLP*.

Stuart M Shieber, Yves Schabes, and Fernando CN Pereira. 1995. Principles and implementation of deductive parsing. *The Journal of logic programming*, 24(1):3–36.

Mark Steedman. 2001. *The Syntactic Process*. MIT Press, Cambridge, MA.

Yuk Wah Wong and Raymond J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-2006)*.

A Vector Space for Distributional Semantics for Entailment *

James Henderson and **Diana Nicoleta Popa**
Xerox Research Centre Europe
james.henderson@xrce.xerox.com and diana.popa@xrce.xerox.com

Abstract

Distributional semantics creates vector-space representations that capture many forms of semantic similarity, but their relation to semantic entailment has been less clear. We propose a vector-space model which provides a formal foundation for a distributional semantics of entailment. Using a mean-field approximation, we develop approximate inference procedures and entailment operators over vectors of probabilities of features being known (versus unknown). We use this framework to reinterpret an existing distributional-semantic model (Word2Vec) as approximating an entailment-based model of the distributions of words in contexts, thereby predicting lexical entailment relations. In both unsupervised and semi-supervised experiments on hyponymy detection, we get substantial improvements over previous results.

1 Introduction

Modelling entailment is a fundamental issue in computational semantics. It is also important for many applications, for example to produce abstract summaries or to answer questions from text, where we need to ensure that the input text entails the output text. There has been a lot of interest in modelling entailment in a vector-space, but most of this work takes an empirical, often ad-hoc, approach to this problem, and achieving good results has been difficult (Levy et al., 2015). In this work, we propose a new framework for modelling entailment in a vector-space, and illustrate its effectiveness with a distributional-semantic model of hyponymy detection.

$\Rightarrow$	*unk*	f	g	$\neg f$
unk	1	0	0	0
f	1	1	0	0
g	1	0	1	0
$\neg f$	1	0	0	1

Table 1: Pattern of logical entailment between nothing known (*unk*), two different features f and g known, and the complement of f ($\neg f$) known.

Unlike previous vector-space models of entailment, the proposed framework explicitly models what information is unknown. This is a crucial property, because entailment reflects what information is and is not known; a representation y entails a representation x if and only if everything that is known given x is also known given y. Thus, we model entailment in a vector space where each dimension represents something we might know. As illustrated in Table 1, knowing that a feature f is true always entails knowing that same feature, but never entails knowing that a different feature g is true. Also, knowing that a feature is true always entails not knowing anything (*unk*), since strictly less information is still entailment, but the reverse is never true. Table 1 also illustrates that knowing that a feature f is false ($\neg f$) patterns exactly the same way as knowing that an unrelated feature g is true. This illustrates that the relevant dichotomy for entailment is known versus unknown, and not true versus false.

Previous vector-space models have been very successful at modelling semantic similarity, in particular using distributional semantic models (e.g. (Deerwester et al., 1990; Schütze, 1993; Mikolov et al., 2013a)). Distributional semantics uses the distributions of words in contexts to induce vector-space embeddings of words, which have been

*This work was partially supported by French ANR grant CIFRE N 1324/2014.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2052–2062,
Berlin, Germany, August 7-12, 2016.

shown to be useful for a wide variety of tasks. Two words are predicted to be similar if the dot product between their vectors is high. But the dot product is an anti-symmetric operator, which makes it more natural to interpret these vectors as representing whether features are true or false, whereas the dichotomy known versus unknown is asymmetric. We surmise that this is why distributional semantic models have had difficulty modelling lexical entailment (Levy et al., 2015).

To develop a vector-space model of whether features are known or unknown, we start with discrete binary vectors, where 1 means known and 0 means unknown. Entailment between these discrete binary vectors can be calculated by independently checking each dimension. But as soon as we try to do calculations with *distributions* over these vectors, we need to deal with the case where the features are not independent. For example, if feature f has a 50% chance of being true and a 50% chance of being false, we can't assume that there is a 25% chance that both f and $\neg f$ are known. This simple case of mutual exclusion is just one example of a wide range of constraints between features which we need to handle in semantic models. These constraints mean that the different dimensions of our vector space are not independent, and therefore exact models are not factorised. Because the models are not factorised, exact calculations of entailment and exact inference of vectors are intractable.

Mean-field approximations are a popular approach to efficient inference for intractable models. In a mean-field approximation, distributions over binary vectors are represented using a single probability for each dimension. These vectors of real values are the basis of our proposed vector space for entailment.

In this work, we propose a vector-space model which provides a formal foundation for a distributional semantics of entailment. This framework is derived from a mean-field approximation to entailment between binary vectors, and includes operators for measuring entailment between vectors, and procedures for inferring vectors in an entailment graph. We validate this framework by using it to reinterpret existing Word2Vec (Mikolov et al., 2013a) word embedding vectors as approximating an entailment-based model of the distribution of words in contexts. This reinterpretation allows us to use existing word embeddings as an unsupervised model of lexical entailment, successfully predicting hyponymy relations using the proposed entailment operators in both unsupervised and semi-supervised experiments.

2 Modelling Entailment in a Vector Space

To develop a model of entailment in a vector space, we start with the logical definition of entailment in terms of vectors of discrete known features: y entails x if and only if all the known features in x are also included in y. We formalise this relation with binary vectors x, y where 1 means known and 0 means unknown, so this discrete entailment relation $(y \Rightarrow x)$ can be defined with the binary formula:

$$P((y \Rightarrow x) \mid x, y) = \prod_k (1 - (1 - y_k)x_k)$$

Given prior probability distributions $P(x), P(y)$ over these vectors, the exact joint and marginal probabilities for an entailment relation are:

$$P(x, y, (y \Rightarrow x)) = P(x)\ P(y) \prod_k (1 - (1 - y_k)x_k)$$

$$P((y \Rightarrow x)) = E_{P(x)} E_{P(y)} \prod_k (1 - (1 - y_k)x_k) \quad (1)$$

We cannot assume that the priors $P(x)$ and $P(y)$ are factorised, because there are many important correlations between features and therefore we cannot assume that the features are independent. As discussed in Section 1, even just representing both a feature f and its negation $\neg f$ requires two different dimensions k and k' in the vector space, because 0 represents unknown and not false. Given valid feature vectors, calculating entailment can consider these two dimensions separately, but to reason with distributions over vectors we need the prior $P(x)$ to enforce the constraint that x_k and $x_{k'}$ are mutually exclusive. In general, such correlations and anti-correlations exist between many semantic features, which makes inference and calculating the probability of entailment intractable.

To allow for efficient inference in such a model, we propose a mean-field approximation. This in effect assumes that the posterior distribution over vectors is factorised, but in practice this is a much weaker assumption than assuming the prior is factorised. The posterior distribution has less uncertainty and therefore is influenced less by non-factorised prior constraints. By assuming a factorised posterior, we can then represent distributions over feature vectors with simple vectors of

probabilities of individual features (or as below, with their log-odds). These real-valued vectors are the basis of the proposed vector-space model of entailment.

In the next two subsections, we derive a mean-field approximation for inference of real-valued vectors in entailment graphs. This derivation leads to three proposed vector-space operators for approximating the log-probability of entailment, summarised in Table 2. These operators will be used in the evaluation in Section 5. This inference framework will also be used in Section 3 to model how existing word embeddings can be mapped to vectors to which the entailment operators can be applied.

2.1 A Mean-Field Approximation

A mean-field approximation approximates the posterior P using a factorised distribution Q. First of all, this gives us a concise description of the posterior $P(x|\ldots)$ as a vector of continuous values $Q(x{=}1)$, where $Q(x{=}1)_k = Q(x_k{=}1) \approx E_{P(x|\ldots)}x_k = P(x_k{=}1|\ldots)$ (i.e. the marginal probabilities of each bit). Secondly, as is shown below, this gives us efficient methods for doing approximate inference of vectors in a model.

First we consider the simple case where we want to approximate the posterior distribution $P(x, y|y{\Rightarrow}x)$. In a mean-field approximation, we want to find a factorised distribution $Q(x, y)$ which minimises the KL-divergence $D_{KL}(Q(x, y)||P(x, y|y{\Rightarrow}x))$ with the true distribution $P(x, y|y{\Rightarrow}x)$.

$$\begin{aligned}
L &= D_{KL}(Q(x,y)||P(x,y|(y{\Rightarrow}x))) \\
&\propto \sum_x Q(x) \log \frac{Q(x,y)}{P(x,y,(y{\Rightarrow}x))} \\
&= \sum_k E_{Q(x_k)} \log Q(x_k) + \sum_k E_{Q(y_k)} \log Q(y_k) \\
&\quad - E_{Q(x)} \log P(x) - E_{Q(y)} \log P(y) \\
&\quad - \sum_k E_{Q(x_k)} E_{Q(y_k)} \log(1{-}(1{-}y_k)x_k)
\end{aligned}$$

In the final equation, the first two terms are the negative entropy of Q, $-H(Q)$, which acts as a maximum entropy regulariser, the final term enforces the entailment constraint, and the middle two terms represent the prior for x and y. One approach (generalised further in the next subsection) to the prior terms $-E_{Q(x)} \log P(x)$ is to bound them by assuming $P(x)$ is a function in the exponential family, giving us:

$$\begin{aligned}
E_{Q(x)} \log P(x) &\geq E_{Q(x)} \log \frac{\exp(\sum_k \theta^x_k x_k)}{\mathcal{Z}_\theta} \\
&= \sum_k E_{Q(x_k)} \theta^x_k x_k - \log \mathcal{Z}_\theta
\end{aligned}$$

where the $\log \mathcal{Z}_\theta$ is not relevant in any of our inference problems and thus will be dropped below.

As typically in mean-field approximations, inference of $Q(x)$ and $Q(y)$ can't be done efficiently with this exact objective L, because of the non-linear interdependence between x_k and y_k in the last term. Thus, we introduce two approximations to L, one for use in inferring $Q(x)$ given $Q(y)$ (forward inference), and one for the reverse inference problem (backward inference). In both cases, the approximation is done with an application of Jensen's inequality to the log function, which gives us an upper bound on L, as is standard practice in mean-field approximations. For forward inference:

$$\begin{aligned}
L \leq & -H(Q) - Q(x_k{=}1)\theta^x_k - E_{Q(y_k)}\theta^y_k y_k \\
& - Q(x_k{=}1) \log Q(y_k{=}1)\ )
\end{aligned} \quad (2)$$

which we can optimise for $Q(x_k{=}1)$:

$$Q(x_k{=}1) = \sigma(\ \theta^x_k + \log Q(y_k{=}1)\ ) \quad (3)$$

where $\sigma()$ is the sigmoid function. The sigmoid function arises from the entropy regulariser, making this a specific form of maximum entropy model. And for backward inference:

$$\begin{aligned}
L \leq & -H(Q) - E_{Q(x_k)}\theta^x_k x_k - Q(y_k{=}1)\theta^y_k \\
& - (1{-}Q(y_k{=}1)) \log(1{-}Q(x_k{=}1))\ )
\end{aligned} \quad (4)$$

which we can optimise for $Q(y_k{=}1)$:

$$Q(y_k{=}1) = \sigma(\ \theta^y_k - \log(1{-}Q(x_k{=}1))\ ) \quad (5)$$

Note that in equations (2) and (4) the final terms, $Q(x_k{=}1) \log Q(y_k{=}1)$ and $(1{-}Q(y_k{=}1)) \log(1{-}Q(x_k{=}1))$ respectively, are approximations to the log-probability of the entailment. We define two vector-space operators, $\circledcirc$ and $\oslash$, to be these same approximations.

$$\begin{aligned}
&\log Q(y{\Rightarrow}x) \\
&\quad \approx \sum_k E_{Q(x_k)} \log(E_{Q(y_k)}(1 - (1{-}y_k)x_k)) \\
&\quad = Q(x{=}1) \cdot \log Q(y{=}1) \ \equiv X \circledcirc Y
\end{aligned}$$

$$\begin{aligned}
&\log Q(y{\Rightarrow}x) \\
&\quad \approx \sum_k E_{Q(y_k)} \log(E_{Q(x_k)}(1 - (1{-}y_k)x_k)) \\
&\quad = (1{-}Q(y{=}1)) \cdot \log(1{-}Q(x{=}1)) \ \equiv Y \oslash X
\end{aligned}$$

$$
\begin{aligned}
X \circledcirc Y &\equiv \sigma(X) \cdot \log \sigma(Y) \\
Y \circledS X &\equiv \sigma(-Y) \cdot \log \sigma(-X) \\
Y \barwedge X &\equiv \sum_k \log(1 - \sigma(-Y_k)\sigma(X_k))
\end{aligned}
$$

Table 2: The proposed entailment operators, approximating $\log P(y \Rightarrow x)$.

We parametrise these operators with the vectors X, Y of log-odds of $Q(x), Q(y)$, namely $X = \log \frac{Q(x=1)}{Q(x=0)} = \sigma^{-1}(Q(x=1))$. The resulting operator definitions are summarised in Table 2.

Also note that the probability of entailment given in equation (1) becomes factorised when we replace P with Q. We define a third vector-space operator, $\barwedge$, to be this factorised approximation, also shown in Table 2.

2.2 Inference in Entailment Graphs

In general, doing inference for one entailment is not enough; we want to do inference in a graph of entailments between variables. In this section we generalise the above mean-field approximation to entailment graphs.

To represent information about variables that comes from outside the entailment graph, we assume we are given a prior $P(x)$ over all variables x_i in the graph. As above, we do not assume that this prior is factorised. Instead we assume that the prior $P(x)$ is itself a graphical model which can be approximated with a mean-field approximation.

Given a set of variables x_i each representing vectors of binary variables x_{ik}, a set of entailment relations $r = \{(i,j)|(x_i \Rightarrow x_j)\}$, and a set of negated entailment relations $\bar{r} = \{(i,j)|(x_i \not\Rightarrow x_j)\}$, we can write the joint posterior probability as:

$$
\begin{aligned}
P(x, r, \bar{r}) = \frac{1}{\mathcal{Z}} P(x) \prod_i \Big(& \\
& \Big(\prod_{j:r(i,j)} \prod_k P(x_{ik} \Rightarrow x_{jk} | x_{ik}, x_{jk}) \Big) \\
& \Big(\prod_{j:\bar{r}(i,j)} (1 - \prod_k P(x_{ik} \Rightarrow x_{jk} | x_{ik}, x_{jk})) \Big) \Big)
\end{aligned}
$$

We want to find a factorised distribution Q that minimises $L = D_{KL}(Q(x)||P(x|r,\bar{r}))$. As above, we bound this loss for each element $X_{ik} = \sigma^{-1}(Q(x_{ik}=1))$ of each vector we want to infer, using analogous Jensen's inequalities for the terms involving nodes i and j such that $r(i,j)$ or $r(j,i)$. For completeness, we also propose similar inequalities for nodes i and j such that $\bar{r}(i,j)$ or $\bar{r}(j,i)$, and bound them using the constants

$$C_{ijk} \geq \prod_{k' \neq k} (1 - \sigma(-X_{ik'})\sigma(X_{jk'})).$$

To represent the prior $P(x)$, we use the terms

$$\theta_{ik}(X_{\bar{ik}}) \leq \log \frac{E_{Q(x_{\bar{ik}})} P(x_{\bar{ik}}, x_{ik}=1)}{1 - E_{Q(x_{\bar{ik}})} P(x_{\bar{ik}}, x_{ik}=1)}$$

where $x_{\bar{ik}}$ is the set of all $x_{i'k'}$ such that either $i' \neq i$ or $k' \neq k$. These terms can be thought of as the log-odds terms that would be contributed to the loss function by including the prior's graphical model in the mean-field approximation.

Now we can infer the optimal X_{ik} as:

$$
\begin{aligned}
X_{ik} \;=\; & \theta_{ik}(X_{\bar{ik}}) + \sum_{j:r(i,j)} -\log \sigma(-X_{jk}) \qquad (6) \\
& + \sum_{j:r(j,i)} \log \sigma(X_{jk}) + \sum_{j:\bar{r}(j,i)} \log \frac{1 - C_{ijk}\sigma(X_{jk})}{1 - C_{ijk}} \\
& + \sum_{j:\bar{r}(i,j)} -\log \frac{1 - C_{ijk}\sigma(-X_{jk})}{1 - C_{ijk}}
\end{aligned}
$$

In summary, the proposed mean-field approximation does inference in entailment graphs by iteratively re-estimating each X_i as the sum of: the prior log-odds, $-\log \sigma(-X_j)$ for each entailed variable j, and $\log \sigma(X_j)$ for each entailing variable j.[1] This inference optimises $X_i \circledcirc X_j$ for each entailing j plus $X_i \circledS X_j$ for each entailed j, plus a maximum entropy regulariser on X_i. Negative entailment relations, if they exist, can also be incorporated with some additional approximations. Complex priors can also be incorporated through their log-odds, simulating the inclusion of the prior within the mean-field approximation.

Given its dependence on mean-field approximations, it is an empirical question to what extent we should view this model as computing real entailment probabilities and to what extent we should view it as a well-motivated non-linear mapping for which we simply optimise the input-output behaviour (as for neural networks (Henderson and Titov, 2010)). In Sections 3 and 5 we argue for the former (stronger) view.

3 Interpreting Word2Vec Vectors

To evaluate how well the proposed framework provides a formal foundation for the distributional semantics of entailment, we use it to re-interpret an

[1] It is interesting to note that $-\log \sigma(-X_j)$ is a non-negative transform of X_j, similar to the ReLU nonlinearity which is popular in deep neural networks (Glorot et al., 2011). $\log \sigma(X_j)$ is the analogous non-positive transform.

existing model of distributional semantics in terms of semantic entailment. There has been a lot of work on how to use the distribution of contexts in which a word occurs to induce a vector representation of the semantics of words. In this paper, we leverage this previous work on distributional semantics by re-interpreting a previous distributional semantic model and using this understanding to map its vector-space word embeddings to vectors in the proposed framework. We then use the proposed operators to predict entailment between words using these vectors. In Section 5 below, we evaluate these predictions on the task of hyponymy detection. In this section we motivate three different ways to interpret the Word2Vec (Mikolov et al., 2013a; Mikolov et al., 2013b) distributional semantic model as an approximation to an entailment-based model of the semantic relationship between a word and its context.

Distributional semantics learns the semantics of words by looking at the distribution of contexts in which they occur. To model this relationship, we assume that the semantic features of a word are (statistically speaking) redundant with those of its context words, and consistent with those of its context words. We model these properties using a hidden vector which is the consistent unification of the features of the middle word and the context. In other words, there must exist a hidden vector which entails both of these vectors, and is consistent with prior constraints on vectors. We split this into two steps, inference of the hidden vector Y from the middle vector X_m, context vectors X_c and prior, and computing the log-probability (7) that this hidden vector entails the middle and context vectors:

$$\max_{Y}(\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c)) \tag{7}$$

We interpret Word2Vec's Skip-Gram model as learning its context and middle word vectors so that the log-probability of this entailment is high for the observed context words and low for other (sampled) context words. The word embeddings produced by Word2Vec are only related to the vectors X_m assigned to the middle words; context vectors are computed but not output. We model the context vectors X'_c as combining (as in equation (5)) information about a context word itself with information which can be inferred from this word given the prior, $X'_c = \theta_c - \log\sigma(-X_c)$.

The numbers in the vectors output by Word2Vec are real numbers between negative infinity and infinity, so the simplest interpretation of them is as the log-odds of a feature being known. In this case we can treat these vectors directly as the X_m in the model. The inferred hidden vector Y can then be calculated using the model of backward inference from the previous section.

$$\begin{aligned} Y &= \theta_c - \log\sigma(-X_c) - \log\sigma(-X_m) \\ &= X'_c - \log\sigma(-X_m) \end{aligned}$$

Since the unification Y of context and middle word features is computed using backward inference, we use the backward-inference operator $\circledcirc$ to calculate how successful that unification was. This gives us the final score:

$$\begin{aligned} &\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\ &\quad\approx\ Y{\circledcirc}X_m + Y{\circledcirc}X_c + -\sigma(-Y){\cdot}\theta_c \\ &\quad=\ Y{\circledcirc}X_m + -\sigma(-Y){\cdot}X'_c \end{aligned}$$

This is a natural interpretation, but it ignores the equivalence in Word2Vec between pairs of positive values and pairs of negative values, due to its use of the dot product. As a more accurate interpretation, we interpret each Word2Vec dimension as specifying whether its feature is known to be true or known to be false. Translating this Word2Vec vector into a vector in our entailment vector space, we get one copy Y^+ of the vector representing known-to-be-true features and a second negated duplicate Y^- of the vector representing known-to-be-false features, which we concatenate to get our representation Y.

$$\begin{aligned} Y^+ &= X'_c - \log\sigma(-X_m) \\ Y^- &= -X'_c - \log\sigma(X_m) \\ &\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\ &\quad\approx\ Y^+{\circledcirc}X_m + -\sigma(-Y^+){\cdot}X'_c \\ &\qquad + Y^-{\circledcirc}(-X_m) + -\sigma(-Y^-){\cdot}(-X'_c) \end{aligned}$$

As a third alternative, we modify this latter interpretation with some probability mass reserved for unknown in the vicinity of zero. By subtracting 1 from both the original and negated copies of each dimension, we get a probability of unknown of $1{-}\sigma(X_m{-}1) - \sigma(-X_m{-}1)$. This gives us:

$$\begin{aligned} Y^+ &= X'_c - \log\sigma(-(X_m{-}1)) \\ Y^- &= -X'_c - \log\sigma(-(-X_m{-}1)) \\ &\log P(y, y{\Rightarrow}x_m, y{\Rightarrow}x_c) \\ &\quad\approx\ Y^+{\circledcirc}(X_m{-}1) + -\sigma(-Y^+){\cdot}X'_c \\ &\qquad + Y^-{\circledcirc}(-X_m{-}1)) + -\sigma(-Y^-){\cdot}(-X'_c) \end{aligned}$$

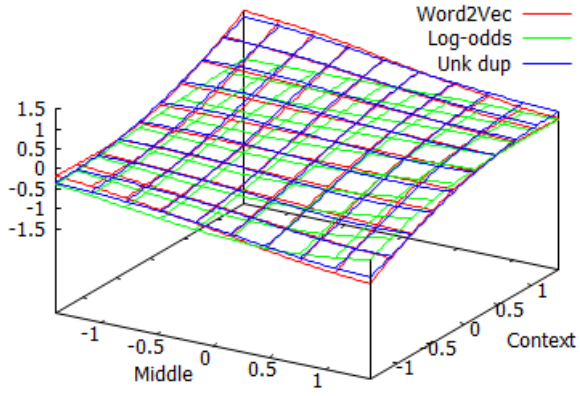


Figure 1: The learning gradients for Word2Vec, the *log-odds* ⊘, and the *unk dup* ⊘ interpretation of its vectors.

To understand better the relative accuracy of these three interpretations, we compared the training gradient which Word2Vec uses to train its middle-word vectors to the training gradient for each of these interpretations. We plotted these gradients for the range of values typically found in Word2Vec vectors for both the middle vector and the context vector. Figure 1 shows three of these plots. As expected, the second interpretation is more accurate than the first because its plot is anti-symmetric around the diagonal, like the Word2Vec gradient. In the third alternative, the constant 1 was chosen to optimise this match, producing a close match to the Word2Vec training gradient, as shown in Figure 1 (*Word2Vec* versus *Unk dup*).

Thus, Word2Vec can be seen as a good approximation to the third model, and a progressively worse approximation to the second and first models. Therefore, if the entailment-based distributional semantic model we propose is accurate, then we would expect the best accuracy in hyponymy detection using the third interpretation of Word2Vec vectors, and progressively worse accuracy for the other two interpretations. As we will see in Section 5, this prediction holds.

4 Related Work

There has been a significant amount of work on using distributional-semantic vectors for hyponymy detection, using supervised, semi-supervised or unsupervised methods (e.g. (Yu et al., 2015; Necsulescu et al., 2015; Vylomova et al., 2015; Weeds et al., 2014; Fu et al., 2015; Rei and Briscoe, 2014)). Because our main concern is modelling entailment within a vector space, we do not do a thorough comparison to models which use measures computed outside the vector space (e.g. symmetric measures (LIN (Lin, 1998)), asymmetric measures (WeedsPrec (Weeds and Weir, 2003; Weeds et al., 2004), balAPinc (Kotlerman et al., 2010), invCL (Lenci and Benotto, 2012)) and entropy-based measures (SLQS (Santus et al., 2014))), nor to models which encode hyponymy in the parameters of a vector-space operator or classifier (Fu et al., 2015; Roller et al., 2014; Baroni et al., 2012)). We also limit our evaluation of lexical entailment to hyponymy, not including other related lexical relations (cf. (Weeds et al., 2014; Vylomova et al., 2015; Turney and Mohammad, 2014; Levy et al., 2014)), leaving more complex cases to future work on compositional semantics. We are also not concerned with models or evaluations which require supervised learning about individual words, instead limiting ourselves to semi-supervised learning where the words in the training and test sets are disjoint.

For these reasons, in our evaluations we replicate the experimental setup of Weeds et al. (2014), for both unsupervised and semi-supervised models. Within this setup, we compare to the results of the models evaluated by Weeds et al. (2014) and to previously proposed vector-space operators. This includes one vector space operator for hyponymy which doesn't have trained parameters, proposed by Rei and Briscoe (2014), called *weighted cosine*. The dimensions of the dot product (normalised to make it a cosine measure) are weighted to put more weight on the larger values in the entailed (hypernym) vector.

We base this evaluation on the Word2Vec (Mikolov et al., 2013a; Mikolov et al., 2013b) distributional semantic model and its publicly available word embeddings. We choose it because it is popular, simple, fast, and its embeddings have been derived from a very large corpus. Levy and Goldberg (2014) showed that it is closely related to the previous PMI-based distributional semantic models (e.g. (Turney and Pantel, 2010)).

The most similar previous work, in terms of motivation and aims, is that of Vilnis and McCallum (2015). They also model entailment directly using a vector space, without training a classifier. But instead of representing words as a point in a vector space (as in this work), they represent words as a Gaussian distribution over points in a vector space. This allows them to represent the extent to which a feature is known versus unknown as the

amount of variance in the distribution for that feature's dimension. While nicely motivated theoretically, the model appears to be more computationally expensive than the one proposed here, particularly for inferring vectors. They do make unsupervised predictions of hyponymy relations with their learned vector distributions, using KL-divergence between the distributions for the two words. They evaluate their models on the hyponymy data from (Baroni et al., 2012). As discussed further in section 5.2, our best models achieve non-significantly better average precision than their best models.

The semi-supervised model of Kruszewski et al. (2015) also models entailment in a vector space, but they use a discrete vector space. They train a mapping from distributional semantic vectors to Boolean vectors such that feature inclusion respects a training set of entailment relations. They then use feature inclusion to predict hyponymy, and other lexical entailment relations. This approach is similar to the one used in our semi-supervised experiments, except that their discrete entailment prediction operator is very different from our proposed entailment operators.

5 Evaluation

To evaluate whether the proposed framework is an effective model of entailment in vector spaces, we apply the interpretations from Section 3 to publicly available word embeddings and use them to predict the hyponymy relations in a benchmark dataset. This framework predicts that the more accurate interpretations of Word2Vec result in more accurate unsupervised models of hyponymy. We evaluate on detecting hyponymy relations between words because hyponymy is the canonical type of lexical entailment; most of the semantic features of a hypernym (e.g. "animal") must be included in the semantic features of the hyponym (e.g. "dog"). We evaluate in both a fully unsupervised setup and a semi-supervised setup.

5.1 Hyponymy with Word2Vec Vectors

For our evaluation on hyponymy detection, we replicate the experimental setup of Weeds et al. (2014), using their selection of word pairs[2] from the BLESS dataset (Baroni and Lenci, 2011).[3] These noun-noun word pairs include positive hyponymy pairs, plus negative pairs consisting of some other hyponymy pairs reversed, some pairs in other semantic relations, and some random pairs. Their selection is balanced between positive and negative examples, so that accuracy can be used as the performance measure. For their semi-supervised experiments, ten-fold cross validation is used, where for each test set, items are removed from the associated training set if they contain any word from the test set. Thus, the vocabulary of the training and testing sets are always disjoint, thereby requiring that the models learn about the vector space and not about the words themselves. We had to perform our own 10-fold split, but apply the same procedure to filter the training set.

We could not replicate the word embeddings used in Weeds et al. (2014), so instead we use publicly available word embeddings.[4] These vectors were trained with the Word2Vec software applied to about 100 billion words of the Google-News dataset, and have 300 dimensions.

The hyponymy detection results are given in Table 3, including both unsupervised (upper box) and semi-supervised (lower box) experiments. We report two measures of performance, hyponymy detection accuracy (*50% Acc*) and direction classification accuracy (*Dir Acc*). Since all the operators only determine a score, we need to choose a threshold to get detection accuracies. Given that the proportion of positive examples in the dataset has been artificially set at 50%, we threshold each model's score at the point where the proportion of positive examples output is 50%, which we call "*50% Acc*". Thus the threshold is set after seeing the testing inputs but not their target labels.

Direction classification accuracy (*Dir Acc*) indicates how well the method distinguishes the relative abstractness of two nouns. Given a pair of nouns which are in a hyponymy relation, it classifies which word is the hypernym and which is the hyponym. This measure only considers positive examples and chooses one of two directions, so it is inherently a balanced binary classification task. Classification is performed by simply comparing the scores in both directions. If both directions produce the same score, the expected random accuracy (50%) is used.

As representative of previous work, we report

[2] https://github.com/SussexCompSem/learninghypernyms

[3] Of the 1667 word pairs in this data, 24 were removed because we do not have an embedding for one of the words.

[4] https://code.google.com/archive/p/word2vec/

operator	supervision	50% Acc	Dir Acc
Weeds et.al.	None	58%	–
log-odds $\oslash$	None	54.0%	55.9%
weighted cos	None	55.5%	57.9%
dot	None	56.3%	50%
dif	None	56.9%	59.6%
log-odds $\vec{\Rightarrow}$	None	57.0%	59.4%
log-odds $\oslash$	None	60.1%*	62.2%
dup $\oslash$	None	61.7%	68.8%
unk dup $\vec{\Rightarrow}$	None	63.4%*	68.8%
unk dup $\oslash$	None	64.5%	68.8%
Weeds et.al.	SVM	75%	–
mapped dif	cross ent	64.3%	72.3%
mapped $\oslash$	cross ent	74.5%	91.0%
mapped $\vec{\Rightarrow}$	cross ent	77.5%	92.3%
mapped $\oslash$	cross ent	80.1%	90.0%

Table 3: Accuracies on the BLESS data from Weeds et al. (2014), for hyponymy detection (*50% Acc*) and hyponymy direction classification (*Dir Acc*), in the unsupervised (upper box) and semi-supervised (lower box) experiments. For unsupervised accuracies, * marks a significant difference with the previous row.

the best results from Weeds et al. (2014), who try a number of unsupervised and semi-supervised models, and use the same testing methodology and hyponymy data. However, note that their word embeddings are different. For the semi-supervised models, Weeds et al. (2014) trains classifiers, which are potentially more powerful than our linear vector mappings. We also compare the proposed operators to the dot product (*dot*),[5] vector differences (*dif*), and the weighted cosine of Rei and Briscoe (2014) (*weighted cos*), all computed with the same word embeddings as for the proposed operators.

In Section 3 we argued for three progressively more accurate interpretations of Word2Vec vectors in the proposed framework, the log-odds interpretation (*log-odds* $\oslash$), the negated duplicate interpretation (*dup* $\oslash$), and the negated duplicate interpretation with unknown around zero (*unk dup* $\oslash$). We also evaluate using the factorised calculation of entailment (*log-odds* $\vec{\Rightarrow}$, *unk dup* $\vec{\Rightarrow}$), and the backward-inference entailment operator (*log-odds* $\oslash$), neither of which match the proposed interpretations. For the semi-supervised case, we train a linear vector-space mapping into a new vector space, in which we apply the operators (*mapped* operators). All these results are discussed in the next two subsections.

[5]We also tested the cosine measure, but results were very slightly worse than *dot*.

5.2 Unsupervised Hyponymy Detection

The first set of experiments evaluate the vector-space operators in unsupervised models of hyponymy detection. The proposed models are compared to the dot product, because this is the standard vector-space operator and has been shown to capture semantic similarity very well. However, because the dot product is a symmetric operator, it always performs at chance for direction classification. Another vector-space operator which has received much attention recently is vector differences. This is used (with vector sum) to perform semantic transforms, such as "king - male + female = queen", and has previously been used for modelling hyponymy (Vylomova et al., 2015; Weeds et al., 2014). For our purposes, we sum the pairwise differences to get a score which we use for hyponymy detection.

For the unsupervised results in the upper box of table 3, the best unsupervised model of Weeds et al. (2014), and the operators *dot*, *dif* and *weighted cos* all perform similarly on accuracy, as does the log-odds factorised entailment calculation (*log-odds* $\vec{\Rightarrow}$). The forward-inference entailment operator (*log-odds* $\oslash$) performs above chance but not well, as expected given the backward-inference-based interpretation of Word2Vec vectors. By definition, *dot* is at chance for direction classification, but the other models all perform better, indicating that all these operators are able to measure relative abstractness. As predicted, the $\oslash$ operator performs significantly better than all these results on accuracy, as well as on direction classification, even assuming the log-odds interpretation of Word2Vec vectors.

When we move to the more accurate interpretation of Word2Vec vectors as specifying both original and negated features (*dup* $\oslash$), we improve (non-significantly) on the log-odds interpretation. Finally, the third and most accurate interpretation, where values around zero can be unknown (*unk dup* $\oslash$), achieves the best results in unsupervised hyponymy detection, as well as for direction classification. Changing to the factorised entailment operator (*unk dup* $\vec{\Rightarrow}$) is worse but also signifi-

cantly better than the other accuracies.

To allow a direct comparison to the model of Vilnis and McCallum (2015), we also evaluated the unsupervised models on the hyponymy data from (Baroni et al., 2012). Our best model achieved 81% average precision on this dataset, non-significantly better than the 80% achieved by the best model of Vilnis and McCallum (2015).

5.3 Semi-supervised Hyponymy Detection

Since the unsupervised learning of word embeddings may reflect many context-word correlations which have nothing to do with hyponymy, we also consider a semi-supervised setting. Adding some supervision helps distinguish features that capture semantic properties from other features which are not relevant to hyponymy detection. But even with supervision, we still want the resulting model to be captured in a vector space, and not in a parametrised scoring function. Thus, we train mappings from the Word2Vec word vectors to new word vectors, and then apply the entailment operators in this new vector space to predict hyponymy. Because the words in the testing set are always disjoint from the words in the training set, this experiment measures how well the original unsupervised vector space captures features that generalise entailment across words, and not how well the mapping can learn about individual words.

Our objective is to learn a mapping to a new vector space in which an operator can be applied to predict hyponymy. We train linear mappings for the $\oslash$ operator (*mapped* $\oslash$) and for vector differences (*mapped dif*), since these were the best performing proposed operator and baseline operator, respectively, in the unsupervised experiments. We do not use the duplicated interpretations because these transforms are subsumed by the ability to learn a linear mapping.[6] Previous work on using vector differences for semi-supervised hyponymy detection has used a linear SVM (Vylomova et al., 2015; Weeds et al., 2014), which is mathematically equivalent to our vector-differences model, except that we use cross entropy loss and they use a large-margin loss and SVM training.

The semi-supervised results in the bottom box of table 3 show a similar pattern to the unsupervised results.[7] The $\oslash$ operator achieves the best generalisation from training word vectors to testing word vectors. The *mapped* $\oslash$ model has the best accuracy, followed by the factorised entailment operator *mapped* $\vec{\Rightarrow}$ and Weeds et al. (2014). Direction accuracies of all the proposed operators (*mapped* $\oslash$, *mapped* $\vec{\Rightarrow}$, *mapped* $\oslash$) reach into the 90's. The *dif* operator performs particularly poorly in this *mapped* setting, perhaps because both the mapping and the operator are linear. These semi-supervised results again support our distributional-semantic interpretations of Word2Vec vectors and their associated entailment operator $\oslash$.

6 Conclusion

In this work, we propose a vector-space model which provides a formal foundation for a distributional semantics of entailment. We developed a mean-field approximation to probabilistic entailment between vectors which represent known versus unknown features. And we used this framework to derive vector operators for entailment and vector inference equations for entailment graphs. This framework allows us to reinterpret Word2Vec as approximating an entailment-based distributional semantic model of words in context, and show that more accurate interpretations result in more accurate unsupervised models of lexical entailment, achieving better accuracies than previous models. Semi-supervised evaluations confirm these results.

A crucial distinction between the semi-supervised models here and much previous work is that they learn a mapping into a vector space which represents entailment, rather than learning a parametrised entailment classifier. Within this new vector space, the entailment operators and inference equations apply, thereby generalising naturally from these lexical representations to the compositional semantics of multi-word expressions and sentences. Further work is needed to explore the full power of these abilities to extract information about entailment from both unlabelled text and labelled entailment data, encode it all in a single vector space, and efficiently perform complex inferences about vectors and entailments. This future work on compositional distributional semantics should further demonstrate the full power of the proposed framework for modelling entailment in a vector space.

[6] Empirical results confirm that this is in practice the case, so we do not include these results in the table.

[7] It is not clear how to measure significance for cross-validation results, so we do not attempt to do so.

References

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, GEMS '11, pages 1–10. Association for Computational Linguistics.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23–32, Avignon, France. Association for Computational Linguistics.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. 2015. Learning semantic hierarchies: A continuous vector space approach. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(3):461–471.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323.

James Henderson and Ivan Titov. 2010. Incremental sigmoid belief networks for grammar learning. *Journal of Machine Learning Research*, 11(Dec):3541–3570.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-geffet. 2010. Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359–389.

Germn Kruszewski, Denis Paperno, and Marco Baroni. 2015. Deriving boolean structures from distributional vectors. *Transactions of the Association for Computational Linguistics*, 3:375–388.

Alessandro Lenci and Giulia Benotto. 2012. Identifying hypernyms in distributional semantic spaces. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics*, SemEval '12, pages 75–79. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Omer Levy, Ido Dagan, and Jacob Goldberger. 2014. Focused entailment graphs for open ie propositions. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 87–97, Ann Arbor, Michigan. Association for Computational Linguistics.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, May–June. Association for Computational Linguistics.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th International Conference on Computational Linguistics - Volume 2*, COLING '98, pages 768–774. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Silvia Necsulescu, Sara Mendes, David Jurgens, Núria Bel, and Roberto Navigli. 2015. Reading between the lines: Overcoming data sparsity for accurate classification of lexical relationships. In *Proceedings of the Fourth Joint Conference on Lexical and Computational Semantics*, pages 182–192, Denver, Colorado. Association for Computational Linguistics.

Marek Rei and Ted Briscoe. 2014. Looking for hyponyms in vector space. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 68–77, Ann Arbor, Michigan. Association for Computational Linguistics.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1025–1036, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2014, April 26-30, 2014, Gothenburg, Sweden*, pages 38–42.

Hinrich Schütze. 1993. Word space. In *Advances in Neural Information Processing Systems 5*, pages 895–902. Morgan Kaufmann.

Peter D. Turney and Saif M. Mohammad. 2014. Experiments with three approaches to recognizing lexical entailment. *CoRR*, abs/1401.8269.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188.

Luke Vilnis and Andrew McCallum. 2015. Word representations via Gaussian embedding. In *Proceedings of the International Conference on Learning Representations 2015 (ICLR)*.

Ekaterina Vylomova, Laura Rimell, Trevor Cohn, and Timothy Baldwin. 2015. Take and took, gaggle and goose, book and read: Evaluating the utility of vector differences for lexical relation learning. In *CoRR 2015*.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 81–88.

Julie Weeds, David Weir, and Diana McCarthy. 2004. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, pages 1015–1021. Association for Computational Linguistics.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2249–2259, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Zheng Yu, Haixun Wang, Xuemin Lin, and Min Wang. 2015. Learning term embeddings for hypernymy identification. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, IJCAI 2015. AAAI Press / International Joint Conferences on Artificial Intelligence.

Hidden Softmax Sequence Model for Dialogue Structure Analysis

Zhiyang He[1], **Xien Liu**[2], **Ping Lv**[2], **Ji Wu**[1]
[1]Department of Electronic Engineering, Tsinghua University, Beijing, China
[2]Tsinghua-iFlytek Joint Laboratory for Speech Technology, Beijing, China
{zyhe_ts, xeliu, luping_ts, wuji_ee}@mail.tsinghua.edu.cn

Abstract

We propose a new unsupervised learning model, hidden softmax sequence model (HSSM), based on Boltzmann machine for dialogue structure analysis. The model employs three types of units in the hidden layer to discovery dialogue latent structures: softmax units which represent latent states of utterances; binary units which represent latent topics specified by dialogues; and a binary unit that represents the global general topic shared across the whole dialogue corpus. In addition, the model contains extra connections between adjacent hidden softmax units to formulate the dependency between latent states. Two different kinds of real world dialogue corpora, Twitter-Post and AirTicketBooking, are utilized for extensive comparing experiments, and the results illustrate that the proposed model outperforms sate-of-the-art popular approaches.

1 Introduction

Dialogue structure analysis is an important and fundamental task in the natural language processing domain. The technology provides essential clues for solving real-world problems, such as producing dialogue summaries (Murray et al., 2006; Liu et al., 2010), controlling conversational agents (Wilks, 2006), and designing interactive dialogue systems (Young, 2006; Allen et al., 2007) etc. The study of modeling dialogues always assumes that for each dialogue there exists an unique latent structure (namely dialogue structure), which consists of a series of latent states.[1]

[1]Also called dialogue acts or speech acts in some past work. In this paper, for simplicity we will only use the term "latent state" to describe the sequential dialogue structure.

Some past works mainly rely on supervised or semi-supervised learning, which always involve extensive human efforts to manually construct latent state inventory and to label training samples. Cohen et al. (2004) developed an inventory of latent states specific to E-mail in an office domain by inspecting a large corpus of e-mail. Jeong et al. (2009) employed semi-supervised learning to transfer latent states from labeled speech corpora to the Internet media and e-mail. Involving extensive human efforts constrains scaling the training sample size (which is essential to supervised learning) and application domains.

In recent years, there has been some work on modeling dialogues with unsupervised learning methods which operate only on unlabeled observed data. Crook et al. (2009) employed Dirichlet process mixture clustering models to recognize latent states for each utterance in dialogues from a travel-planning domain, but they do not inspect dialogues' sequential structure. Chotimongkol (2008) proposed a hidden Markov model (HMM) based dialogue analysis model to study structures of task-oriented conversations from in-domain dialogue corpus. More recently, Ritter et al. (2010) extended the HMM based conversation model by introducing additional word sources for topic learning process. Zhai et al. (2014) assumed words in an utterance are emitted from topic models under HMM framework, and topics were shared across all latent states. All these dialogue structure analysis models are directed generative models, in which the HMMs, language models and topic models are combined together.

In this study, we attempt to develop a Boltzmann machine based **undirected generative model** for dialogue structure analysis. As for the document modeling using undirected generative model, Hinton and Salakhutdinov (2009) proposed a general framework, replicated soft-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2063–2072,
Berlin, Germany, August 7-12, 2016.

max model (RSM), for topic modeling based on restricted Boltzmann machine (RBM). The model focuses on the document-level topic analysis, it cannot be applied for the structure analysis. We propose a hidden softmax sequence model (HSSM) for the dialogue modeling and structure analysis. HSSM is a two-layer special Boltzmann machine. The visible layer contains softmax units used to model words in a dialogue, which are the same with the visible layer in RSM (Hinton and Salakhutdinov, 2009). However, the hidden layer has completely different design. There are three kinds of hidden units: softmax hidden units, which is utilized for representing latent states of dialogues; binary units used for representing dialogue specific topics; and a special binary unit used for representing the general topic of the dialogue corpus. Moreover, unlike RSM whose hidden binary units are conditionally independent when visible units are given, HSSM has extra connections utilized to formulate the dependency between adjacent softmax units in the hidden layer. The connections are the latent states of two adjacent utterances. Therefore, HSSM can be considered as a special Boltzmann machine.

The remainder of this paper is organized as follows. Section 2 introduces two real world dialogue corpora utilized in our experiments. Section 3 describes the proposed hidden softmax sequence model. Experimental results and discussions are presented in Section 4. Finally, Section 5 presents our conclusions.

2 Data Set

Two different datasets are utilized to test the effectiveness of our proposed model: a corpus of post conversations drawn from Twitter (Twitter-Post), and a corpus of task-oriented human-human dialogues in the airline ticket booking domain (AirTicketBooking).

2.1 Twitter-Post

Conversations in Twitter are carried out by replying or responding to specific posts with short 140-character messages. The post length restriction makes Twitter keep more chat-like interactions than blog posts. The style of writing used on Twitter is widely varied, highly ungrammatical, and often with spelling errors. For example, the terms "be4", "b4", and "bef4" are always appeared in the Twitter posts to represent the word "before". Here, we totally collected about 900, 000 raw Twitter dialogue sessions. The majority of conversation sessions are very short; and the frequencies of conversation session lengths follow a power law relationship as described in (Ritter et al., 2010). For simplicity , in the data preprocessing stage non-English sentences were dropped; and non-English characters, punctuation marks, and some non-meaning tokens (such as "&") were also filtered from dialogues. We filtered short Twitter dialogue sessions and randomly sampled 5,000 dialogues (the numbers of utterances in dialogues rang from 5 to 25) to build the Twitter-Post dataset.

2.2 AirTicketBooking

The AirTicketBooking corpus consists of a set of task-oriented human-human mandarin dialogues from an airline ticket booking service center. The manual transcripts of the speech dialogues are utilized in our experiments. In the dataset, there is always a relative clear structure underlying each dialogue. A dialogue often begins with a customer's request about airline ticket issues. And the service agent always firstly checks the client's personal information, such as name, phone number and credit card numberm, etc. Then the agent starts to deal with the client's request. We totally collected 1,890 text-based dialogue sessions obtaining about 40,000 conversation utterances with length ranging from 15 to 100.

3 Dialogue Structure Analysis

3.1 Model Design

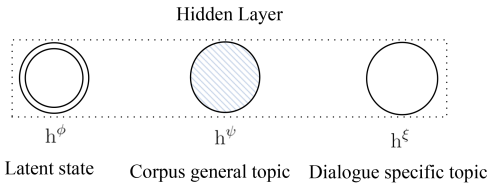


Figure 1: Hidden layer that consists of different types of latent variables

We design an undirected generative model based on Boltzmann machine. As we known, dialogue structure analysis models are always based on an underlying assumption: each utterance in the dialogues is generated from one latent state, which has a causal effect on the words. For instance, an utterance in AirTicketBooking dataset, "Tomorrow afternoon, about 3 o'clock" corre-

sponds to the latent state "Time Information". However, by carefully examining words in dialogues we can observe that not all words are generated from the latent states (Ritter et al., 2010; Zhai and Williams, 2014). There are some words relevant to a global or background topic shared across dialogues. For example, "about" and "that" belong to a global (general English) topic. Some other words in a dialogue may be strongly related to the dialogue specific topic. For example, "cake", "toast" and "pizza" may appear in a Twitter dialogue with respect to a specific topic, "food". From the perspective of generative model, we can also consider that words in a dialogue are generated by the mixture model of latent states, a global/background topic, and a dialogue specific topic. Therefore, there are three kinds of units in the hidden layer of our proposed model, which are displayed in Figure 1. $\mathbf{h}^{\phi}$ is a softmax unit, which indicates the latent state for a utterance. $\mathbf{h}^{\psi}$ and $\mathbf{h}^{\xi}$ represent the general topic, and the dialogue specific topic, respectively. For the visible layer, we utilize the softmax units to model words in each utterance, which is the same with the approach in RSM (Hinton and Salakhutdinov, 2009). In Section 3.2, We propose a basic model based on Boltzmann machine to formulate each word in utterances of dialogues.

A dialogue can be abstractly viewed as a sequence of latent states in a certain reasonable order. Therefore, formulating the dependency between latent states is another import issue for dialogue structure analysis. In our model, we assume that each utterance's latent state is dependent on its two neighbours. So there exist connections between each pair of adjacent hidden softmax units in the hidden layer. The details of the model will be presented in Section 3.3.

3.2 HSM: Hidden Softmax Model

Notation	Explanation
K	dictionary size
J	number of latent states
$\mathbf{V}$	observed visibles representing words in dialogues
$\mathbf{b}$	bias term of $\mathbf{V}$
$\mathbf{h}^{\phi}$	latent variables representing latent states
$\mathbf{h}^{\psi}$	latent variable representing corpus general topic
$\mathbf{h}^{\xi}$	latent variables representing dialogue specific topics
$\mathbf{a}^{\phi}$	bias terms of $\mathbf{h}^{\phi}$
$\mathbf{a}^{\psi}$	bias term of $\mathbf{h}^{\psi}$
$\mathbf{a}^{\xi}$	bias terms of $\mathbf{h}^{\xi}$
$\mathbf{W}^{\phi}$	weights connecting $\mathbf{h}^{\phi}$ to $\mathbf{V}$
$\mathbf{W}^{\psi}$	weights connecting $\mathbf{h}^{\psi}$ to $\mathbf{V}$
$\mathbf{W}^{\xi}$	weights connecting $\mathbf{h}^{\xi}$ to $\mathbf{V}$
$\mathbf{F}, \mathbf{F}^{s}, \mathbf{F}^{e}$	weights between hidden softmax units

Table 1: Definition of notations.

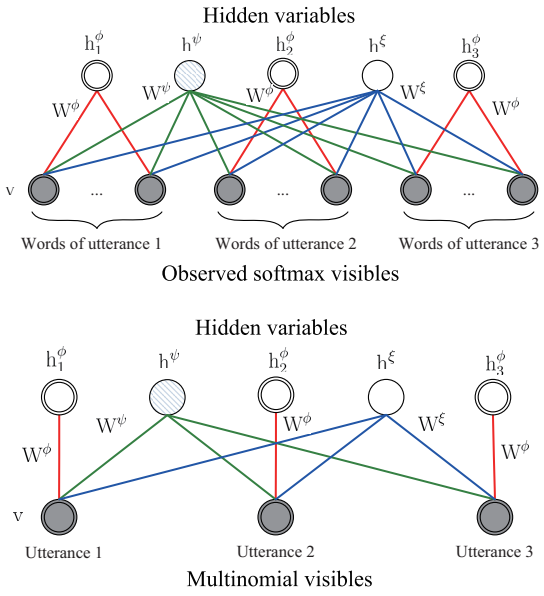


Figure 2: Hidden Softmax Model. The bottom layer are softmax visible units and the top layer consists of three types of hidden units: softmax hidden units used for representing latent states, a binary stochastic hidden unit used for representing the dialogue specific topic, and a special binary stochastic hidden unit used for representing corpus general topic. **Upper**: The model for a dialogue session containing three utterances. Connection lines in the same color related to a latent state represent the same weight matrix. **Lower**: A different interpretation of the Hidden Softmax Model, in which D_r visible softmax units in the r^{th} utterance are replaced by one single multinomial unit which is sampled D_r times.

Table 1 summarizes important notations utilized in this paper. Before introducing the ultimate learning model for dialogue structure analysis, we firstly discuss a simplified version, Hidden Softmax Model (HSM), which is based on Boltzmann machine and assumes that the latent variables are independent given visible units. HSM has a two-layer architecture as shown in Figure 2. The energy of the state $\{\mathbf{V}, \mathbf{h}^{\phi}, \mathbf{h}^{\psi}, \mathbf{h}^{\xi}\}$ is defined as follows:

$$E(\mathbf{V}, \mathbf{h}^{\phi}, \mathbf{h}^{\psi}, \mathbf{h}^{\xi}) = \bar{E}_{\phi}(\mathbf{V}, \mathbf{h}^{\phi}) + \bar{E}_{\psi}(\mathbf{V}, \mathbf{h}^{\psi}) + \bar{E}_{\xi}(\mathbf{V}, \mathbf{h}^{\xi}) + C(\mathbf{V}), \quad (1)$$

where $\bar{E}_{\phi}(\mathbf{V}, \mathbf{h}^{\phi})$, $\bar{E}_{\psi}(\mathbf{V}, \mathbf{h}^{\psi})$ and $\bar{E}_{\xi}(\mathbf{V}, \mathbf{h}^{\xi})$ are sub-energy functions related to hidden variables $\mathbf{h}^{\phi}$, $\mathbf{h}^{\psi}$, and $\mathbf{h}^{\xi}$, respectively. $C(\mathbf{V})$ is the shared visible units bias term. Suppose K is the dictionary size, D_r is the r^{th} utterance size (i.e. the

number of words in the r^{th} utterance), and R is the number of utterances in the a dialogue.

For each utterance $v_r(r = 1, .., R)$ in the dialogue session we have a hidden variable vector h_r^ϕ (with size of J) as a latent state of the utterance, the sub-energy function $\bar{E}_\phi(\mathbf{V}, \mathbf{h}^\phi)$ is defined by

$$\begin{aligned}\bar{E}_\phi(\mathbf{V}, \mathbf{h}^\phi) = &- \sum_{r=1}^{R}\sum_{j=1}^{J}\sum_{i=1}^{D_r}\sum_{k=1}^{K} h_{rj}^\phi W_{rjik}^\phi v_{rik} \\ &- \sum_{r=1}^{R}\sum_{j=1}^{J} h_{rj}^\phi a_{rj}^\phi,\end{aligned} \tag{2}$$

where $v_{rik} = 1$ means the i^{th} visible unit v_{ri} in the r^{th} utterance takes on k^{th} value, $h_{rj}^\phi = 1$ means the r^{th} softmax hidden units takes on j^{th} value, and a_{rj}^ϕ is the corresponding bias. W_{rjik}^ϕ is a symmetric interaction term between visible unit v_{ri} that takes on k^{th} value and hidden variable h_r^ϕ that takes on j^{th} value.

The sub-energy function $\bar{E}_\psi(\mathbf{V}, \mathbf{h}^\psi)$, related to the global general topic of the corpus, is defined by

$$\bar{E}_\psi(\mathbf{V}, \mathbf{h}^\psi) = -\sum_{r=1}^{R}\sum_{i=1}^{D_r}\sum_{k=1}^{K} h^\psi W_{rik}^\psi v_{rik} - h^\psi a^\psi. \tag{3}$$

The sub-energy function $\bar{E}_\xi(\mathbf{V}, \mathbf{h}^\xi)$ corresponds to the dialogue specific topic, and is defined by

$$\bar{E}_\xi(\mathbf{V}, \mathbf{h}^\xi) = -\sum_{r=1}^{R}\sum_{i=1}^{D_r}\sum_{k=1}^{K} h^\xi W_{rik}^\xi v_{rik} - h^\xi a^\xi. \tag{4}$$

W_{rik}^ψ in Eq. (3) and W_{rik}^ξ in Eq. (4) are two symmetric interaction terms between visible units and the corresponding hidden units, which are similar to W_{rjik}^ϕ in (2); a^ψ and a^ξ are the corresponding biases. $C(\mathbf{V})$ is defined by

$$C(\mathbf{V}) = -\sum_{r=1}^{R}\sum_{i=1}^{D_r}\sum_{k=1}^{K} v_{rik} b_{rik}, \tag{5}$$

where b_{rik} is the corresponding bias.

The probability that the model assigns to a visible binary matrix $\mathbf{V} = \{v_1, v_2, ..., v_D\}$ (where $D = \sum_{r=1}^{R} D_r$ is the dialogue session size) is

$$\begin{aligned}P(\mathbf{V}) &= \frac{1}{\mathcal{Z}} \sum_{\mathbf{h}^\phi,\, \mathbf{h}^\psi, \mathbf{h}^\xi} \exp(-E(\mathbf{V}, \mathbf{h}^\phi,\, \mathbf{h}^\psi, \mathbf{h}^\xi)) \\ \mathcal{Z} &= \sum_{\mathbf{V}} \sum_{\mathbf{h}^\phi,\, \mathbf{h}^\psi, \mathbf{h}^\xi} \exp(-E(\mathbf{V}, \mathbf{h}^\phi,\, \mathbf{h}^\psi,\, \mathbf{h}^\xi),\end{aligned} \tag{6}$$

where $\mathcal{Z}$ is known as the partition function or normalizing constant.

In our proposed model, for each word in the document we use a softmax unit to represent it. For the sake of simplicity, assume that the order of words in an utterance is ignored. Therefore, all of these softmax units can share the same set of weights that connect them to hidden units, thus the visible bias term $C(\mathbf{V})$ and the sub-energy functions $\bar{E}_\phi(\mathbf{V}, \mathbf{h}^\phi)$, $\bar{E}_\psi(\mathbf{V}, \mathbf{h}^\psi)$ and $\bar{E}_\xi(\mathbf{V}, \mathbf{h}^\xi)$ in Eq. (1) can be redefined as follows:

$$\begin{aligned}\bar{E}_\phi(\mathbf{V}, \mathbf{h}^\phi) = &- \sum_{r=1}^{R}\sum_{j=1}^{J}\sum_{k=1}^{K} h_{rj}^\phi W_{jk}^\phi \hat{v}_{rk} \\ &- \sum_{r=1}^{R}(D_r \sum_{j=1}^{J} h_{rj}^\phi a_j^\phi)\end{aligned} \tag{7}$$

$$\bar{E}_\psi(\mathbf{V}, \mathbf{h}^\psi) = -\sum_{k=1}^{K} h^\psi W_k^\psi \hat{v}_k - D h^\psi a^\psi \tag{8}$$

$$\bar{E}_\xi(\mathbf{V}, \mathbf{h}^\xi) = -\sum_{k=1}^{K} h^\xi W_k^\xi \hat{v}_k - D h^\xi a^\xi \tag{9}$$

$$C(\mathbf{V}) = -\sum_{k=1}^{K} \hat{v}_k b_k, \tag{10}$$

where $\hat{v}_{rk} = \sum_{i=1}^{D_r} v_{rik}$ denotes the count for the k^{th} word in the r^{th} utterance of the dialogue, $\hat{v}_k = \sum_{r=1}^{R} \hat{v}_{rk}$ is the count for the k^{th} word in whole dialogue session. D_r and D ($D = \sum_{r=1}^{R} D_r$) are employed as the scaling parameters, which can make hidden units behave sensibly when dealing with dialogues of different lengths (Hinton and Salakhutdinov, 2009).

The conditional distributions are given by softmax and logistic functions:

$$P(h_{rj}^\phi = 1|\mathbf{V}) = \frac{exp(\sum_{k=1}^{K} W_{jk}^\phi \hat{v}_{rk} + D_r a_j^\phi)}{\sum_{j'=1}^{J} exp(\sum_{k=1}^{K} W_{j'k}^\phi \hat{v}_{rk} + D_r a_{j'}^\phi)} \tag{11}$$

$$P(h^\psi = 1|\mathbf{V}) = \sigma(\sum_{k=1}^{K} W_k^\psi \hat{v}_k + D a^\psi) \tag{12}$$

$$P(h^\xi = 1|\mathbf{V}) = \sigma(\sum_{k=1}^{K} W_k^\xi \hat{v}_k + D a^\xi) \tag{13}$$

$$\begin{aligned}&P(v_{rik} = 1|\mathbf{h}^\phi, \mathbf{h}^\psi, \mathbf{h}^\xi) = \\ &\frac{exp(\sum_{j=1}^{J} h_{rj}^\phi W_{jk}^\phi + h^\psi W_k^\psi + h^\xi W_k^\xi + b_k)}{\sum_{k'=1}^{K} exp(\sum_{j=1}^{J} h_{rj}^\phi W_{jk'}^\phi + h^\psi W_{k'}^\psi + h^\xi W_{k'}^\xi + b_{k'})},\end{aligned} \tag{14}$$

where $\sigma(x) = 1/(1 + exp(-x))$ is the logistic function.

3.3 HSSM: Hidden Softmax Sequence Model

In this section, we consider the dependency between the adjacent latent states of utterances, and extend the HSM to hidden softmax sequence model (HSSM), which is displayed in Figure 3. We define the energy of the state $\{\mathbf{V}, \mathbf{h}^\phi, \mathbf{h}^\psi, \mathbf{h}^\xi\}$ in HSSM as follows:

$$\begin{aligned} E(\mathbf{V}, \mathbf{h}^\phi, \mathbf{h}^\psi, \mathbf{h}^\xi) =& \bar{E}_\phi(\mathbf{V}, \mathbf{h}^\phi) + \bar{E}_\psi(\mathbf{V}, \mathbf{h}^\psi) + \bar{E}_\xi(\mathbf{V}, \mathbf{h}^\xi) \\ &+ C(\mathbf{V}) + \bar{E}_\Phi(\mathbf{h}^\phi, \mathbf{h}^\phi), \end{aligned} \tag{15}$$

where $C(\mathbf{V})$, $\bar{E}_\phi(\mathbf{V}, \mathbf{h}^\phi)$, $\bar{E}_\psi(\mathbf{V}, \mathbf{h}^\psi)$ and $\bar{E}_\xi(\mathbf{V}, \mathbf{h}^\xi)$ are the same with that in HSM. The last term $\bar{E}_\Phi(\mathbf{h}^\phi, \mathbf{h}^\phi)$ is utilized to formulate the dependency between latent variables $\mathbf{h}^\phi$, which is defined as follows:

$$\begin{aligned} \bar{E}_\Phi(\mathbf{h}^\phi, \mathbf{h}^\phi) = &- \sum_{q=1}^{J} h_s^\phi F_q^s h_{1q}^\phi - \sum_{q=1}^{J} h_{Rq}^\phi F_q^e h_e^\phi \\ &- \sum_{r=1}^{R-1} \sum_{j=1}^{J} \sum_{q=1}^{J} h_{rj}^\phi F_{jq} h_{r+1,q}^\phi, \end{aligned} \tag{16}$$

where h_s^ϕ and h_e^ϕ are two constant scalar variables ($h_s^\phi \equiv 1$, $h_e^\phi \equiv 1$), which represent the virtual beginning state unit and ending state unit of a dialogue. F^s is a vector with size J, and its elements measure the dependency between h_s^ϕ and the latent softmax units of the first utterance. F^e also contains J elements, and in contrast to F^s, F^e represents the dependency measure between h_e^ϕ and the latent softmax units of the last utterance. F is a symmetric matrix for formulating dependency between each two adjacent hidden units pair (h_r^ϕ, h_{r+1}^ϕ), $r = 1, ..., R-1$.

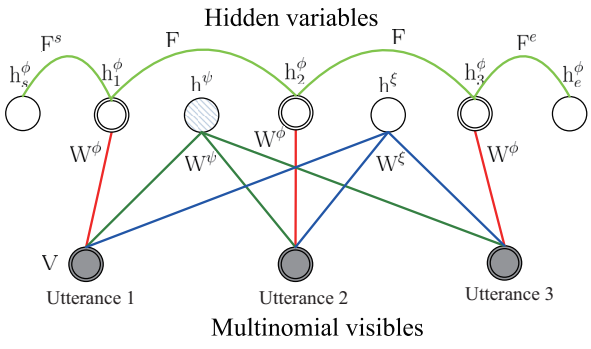


Figure 3: Hidden softmax sequence model. A connection between each pair of adjacent hidden softmax units is added to formulate the dependency between the two corresponding latent states.

3.4 Parameter Learning

Exact maximum likelihood learning in the proposed model is intractable. "Contrastive Divergence" (Hinton, 2002) can be used for HSM's learning, however, it can not be utilized for HSSM, because the hidden-to-hidden interaction term, $\{F, F^s, F^e\}$, result in the intractability when obtaining exact samples from the conditional distribution $P(h_{rj}^\phi = 1|\mathbf{V})$, $r = [1, R]$, $j \in [1, J]$. We use the mean-field variational inference (Hinton and Zemel, 1994; Neal and Hinton, 1998; Jordan et al., 1999) and a stochastic approximation procedure (SAP) (Tieleman, 2008) to estimate HSSM's parameters. The variational learning is utilized to get the data-dependent expectations, and SAP is utilized to estimate the model's expectation. The log-likelihood of the HSSM has the following variational lower bound:

$$\log P(\mathbf{V}; \theta) \geq \sum_{\mathbf{h}} Q(\mathbf{h}) \log P(\mathbf{V}, \mathbf{h}; \theta) + H(Q). \tag{17}$$

$Q(\mathbf{h})$ can be any distribution of $\mathbf{h}$ in theory. $\theta = \{W^\phi, W^\psi, W^\xi, F, F^s, F^e\}$ (the bias terms are omitted for clarity) are the model parameters. $\mathbf{h} = \{\mathbf{h}^\phi, \mathbf{h}^\psi, \mathbf{h}^\xi\}$ represent all the hidden variables. $H(\cdot)$ is the entropy functional. In variational learning, we try to find parameters that minimize the Kullback-Leibler divergences between $Q(\mathbf{h})$ and the true posterior $P(\mathbf{h}|\mathbf{V}; \theta)$. A naive mean-field approach can be chosen to obtain a fully factorized distribution for $Q(\mathbf{h})$:

$$Q(\mathbf{h}) = \left[\prod_{r=1}^{R} q(h^\phi)\right] q(h^\psi)\, q(h^\xi), \tag{18}$$

where $q(h_{rj}^\phi = 1) = \mu_{rj}^\phi$, $q(h^\psi = 1) = \mu^\psi$, $q(h^\xi = 1) = \mu^\xi$. $\boldsymbol{\mu} = \{\mu^\phi, \mu^\psi, \mu^\xi\}$ are the parameters of $Q(\mathbf{h})$. Then the lower bound on the log-probability $\log P(\mathbf{V}; \theta)$ has the form:

$$\begin{aligned} \log P(\mathbf{V}; \theta) \geq &- \bar{E}_\phi(\mathbf{V}, \mu^\phi) - \bar{\mathbf{E}}_\psi(\mathbf{V}, \mu^\psi) - \bar{\mathbf{E}}_\xi(\mathbf{V}, \mu^\xi) \\ &- C(\mathbf{V}) - \bar{E}_\Phi(\mu^\phi, \mu^\phi) - \log \mathcal{Z}, \end{aligned} \tag{19}$$

where $\bar{E}_\phi(\mathbf{V}, \mu^\phi)$, $\bar{E}_\psi(\mathbf{V}, \mu^\psi)$, $\bar{E}_\xi(\mathbf{V}, \mu^\xi)$, and $\bar{E}_\Phi(\mu^\phi, \mu^\phi)$ have the same forms, by replacing $\boldsymbol{\mu}$ with $\mathbf{h}$, as Eqs. (7), (8), (9), and (16), respectively.

We can maximize this lower bound with respect to parameters $\boldsymbol{\mu}$ for fixed θ, and obtain the mean-field fixed-point equations:

$$\mu_{rj}^\phi = \frac{exp(\sum_{k=1}^{K} W_{jk}^\phi \hat{v}_{rk} + D_r a_j^\phi + D_{prev}^j + D_{next}^j - 1)}{\sum_{j'=1}^{J} exp(\sum_{k=1}^{K} W_{j'k}^\phi \hat{v}_{rk} + D_r a_{j'}^\phi + D_{prev}^{j'} + D_{next}^{j'} - 1)}, \tag{20}$$

$$\mu^{\psi} = \sigma(\sum_{k=1}^{K} W_k^{\psi} \hat{v}_k + Da^{\psi}) \quad (21)$$

$$\mu^{\xi} = \sigma(\sum_{k=1}^{K} W_k^{\xi} \hat{v}_k + Da^{\xi}), \quad (22)$$

where D_{prev}^j and D_{next}^j are two terms relevant to the derivative of the RHS of Eq. (19) with respect to μ_{rj}^{ϕ}, defined by

$$D_{prev}^j = \begin{cases} F_j^s, & r = 1 \\ \sum_{q=1}^{J} \mu_{r-1,q}^{\phi} F_{qj}, & r > 1 \end{cases}$$

$$D_{next}^j = \begin{cases} \sum_{q=1}^{J} F_{jq} \mu_{r+1,q}^{\phi}, & r < R. \\ F_j^e, & r = R \end{cases}$$

The updating of $\boldsymbol{\mu}$ can be carried out iteratively until convergence. Then, $(\mathbf{V}, \boldsymbol{\mu})$ can be considered as a special "state" of HSSM, thus the SAP can be applied to update the model's parameters, θ, for fixed $(\mathbf{V}, \boldsymbol{\mu})$.

4 Experiments and Discussions

It's not easy to evaluate the performance of a dialogue structure analysis model. In this study, we examined our model via qualitative visualization and quantitative analysis as done in (Ritter et al., 2010; Zhai and Williams, 2014). We implemented five conventional models to conduct an extensive comparing study on the two corpora: Twitter-Post and AirTicketBooking. Conventional models include: LMHMM (Chotimongkol, 2008), LMHMMS (Ritter et al., 2010), TMHMM, TMHMMS, and TMHMMSS (Zhai and Williams, 2014). In our experiments, for each corpus we randomly select 80% dialogues for training, and use the rest 20% for testing. We select three different number (10, 20 and 30) of latent states to evaluate all the models. In TMHMM, TMHMMS and TMHMMSS, the number of "topics" in the latent states and a dialogue is a hyper-parameter. We conducted a series of experiments with varying numbers of topics, and the results illustrated that 20 is the best choice on the two corpora. So, for all the following experimental results of TMHMM, TMHMMS and TMHMMSS, the corresponding topic configurations are set to 20.

The number of estimation iterations for all the models on training sets is set to 10,000; and on held-out test sets, the numver of iterations for inference is set to 1000. In order to speed-up the learning of HSSM, datasets are divided into mini-batches, each has 15 dialogues. In addition, the learning rate and momentum are set to 0.1 and 0.9, respectively.

4.1 Qualitative Evaluation

Dialogues in Twitter-Post always begin with three latent states: broadcasting what they (Twitter users) are doing now ("Status"), broadcasting an interesting link or quote to their followers ("Reference Broadcast"), or asking a question to their followers ("Question to Followers").[2] We find that structures discoverd by HSSM and LMHMMS with 10 latent states are most reasonable to interpret. For example, after the initiating state ("Status", "Reference Broadcast", or "Question to Followers"), it was often followed a "Reaction" to "Reference Broadcast" (or "Status"), or a "Comment" to "Status", or a "Question" to "Status" ("Reference Broadcast", or "Question to Followers"') etc. Compared with LMHMMS, besides obtaining similar latent states, HSSM exhibits powerful ability in learning sequential dependency relationship between latent states. Take the following simple Twitter dialogue session as an example:

: *rt i like katy perry lt lt we see tht lol*
: *lol gd morning*
: *lol gd morning how u*
: *i'm gr8 n urself*
: *i'm good gettin ready to head out*
: *oh ok well ur day n up its cold out here*
...

LMHMMS labelled the second utterance ("*lol gd morning* ") and the third utterance ("*lol good morning how u* ") into the same latent state, while HSSM treats them as two different latent states (Though they both have almost the same words). The result is reasonable: the first "*gd morning*" is a greeting, while the second "*gd morning*" is a response.

For AirTicketBooking dataset, the state-transition diagram generated with our model under the setting of 10 latent states is presented in Figure 4. And several utterance examples corresponding to the latent staes are also showed in Table 2. In general, conversations begin with sever agent's short greeting, such as "Hi, very glad to be of service.", and then transit to checking the passenger's identity information or

[2]For simplicity and readability in consistent, we follow the same latent state names used in (Ritter et al., 2010)

inquiring the passenger's air ticket demand; or it's directly interrupted by the passenger with booking demand which is always associated with place information. After that, conversations are carried out with other booking related issues, such as checking ticket price or flight time.

The flowchart produced by HSSM can be reasonably interpreted with knowledge of air ticket booking domain, and it most consistent with the agent's real workflow of the Ticket Booking Corporation[3] compared with other models. We notice that conventional models can not clearly distinguish some relevant latent states from each other. For example, these baseline models always confound the latent state "Price Info" with the latent state "Reservation", due to certain words assigned large weights in the two states, such as "打折 (discount)", and "信用卡 (credit card)" etc. Furthermore, Only HSSM and LMHMMS have dialogue specific topics, and experimental results illustrate that HSSM can learn much better than LMHMMS which always mis-recognize corpus general words as belonging to dialogue specific topic (An example is presented in Table 3).

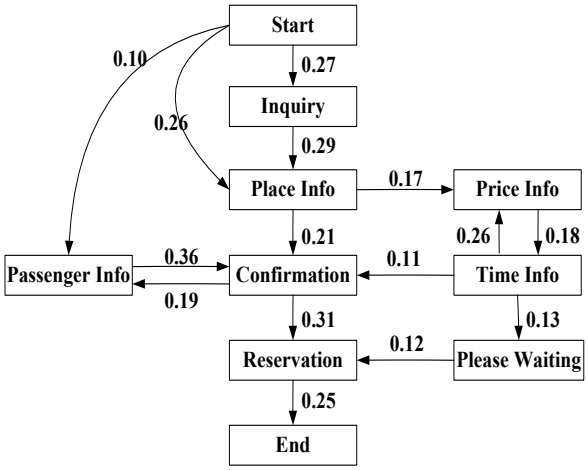


Figure 4: Transitions between latent states on AirTicketBooking generated by our HSSM model under the setting of $J = 10$ latent states. Transition probability cut-off is 0.10.

4.2 Quantitative Evaluation

For quantitative evaluation, we examine HSSM and traditional models with log likelihood and an ordering task on the held-out test set of Twitter-Post and AirTicketBooking.

[3]We hide the corporation's real name for privacy reasons.

Latent States	Utterance Examples (Chinese)	Utterance Examples (English Translation)
Start	您好，很高兴为您服务。	Hello, very glad to be of service.
Inquiry	您想预定机票吗?	Do you want to make a flight reservation?
Place Info	我想预定一张北京到上海的机票。	I want to book an air ticket from Beijing to Shanghai.
Time Info	明天上午10点左右。	Tomorrow morning, about 10 o'clock.
Price Info	成人机票1300元一张。	The adult ticket is 1300 Yuan.
Passenger Info	姓名李东，身份证号12345。	My name is Li Dong, and my ID number is 12345.
Confirmation	好的，可以。	Yes, that's OK.
Please Waiting	请稍等，我帮您查询。	Please wait a moment, I'll check for you.
Reservation	请预定一张，我想用信用卡支付。	Please make a reservation, I want to use a credit card to pay.
End	欢迎下次来电，再见。	Welcome to call next time. Bye.

Table 2: Utterance examples of latent states discovered by our model.

Model	Top Words
HSSM	十点, 李东, 福州, 厦门, 上航, ... ten o'clock, Dong Li (name), Fuzhou (city), Xiamen (city), Shanghai Airlines, ...
LMHMMS	有, 十点, 额, 李东, 预留, ... have, ten o'clock, er, Dong Li (name), reserve, ...

Table 3: One example of dialogue specific topic learned on the same dialogue session with HSSM and LMHMMS, respectively.

Log Likelihood The likelihood metric measures the probability of generating the test set using a specified model. The likelihood of LMHMM and TMHMM can be directed computed with the forward algorithm. However, since likelihoods of LMHMMS, TMHMMS and TMHMMSS are intractable to compute due to the local dependencies with respect to certain latent variables, Chib-style estimating algorithms (Wallach et al., 2009) are employed in our experiments. For HSSM, the partition function is a key problem for calculating the likelihood, and it can be effectively estimated by Annealed Importance Sampling (AIS) (Neal, 2001; Salakhutdinov and Murray, 2008).

Figure 5 presents the likelihood of different models on the two held-out datasets. We can observe that HSSM achieves better performance on likelihood than all the other models under different number of latent states. On Twitter-Post dataset our model slightly surpasses LMHMMS, and it performs much better than all traditional models on AirTicketBooking dataset.

Ordering Test Following previous work (Barzilay and Lee, 2004; Ritter et al., 2010; Zhai and Williams, 2014), we utilize Kendall's τ (Kendall, 1938) as evaluation metric, which measures the similarity between any two sequential data and ranges from -1 (indicating a reverse ordering) to $+1$ (indicating an identical

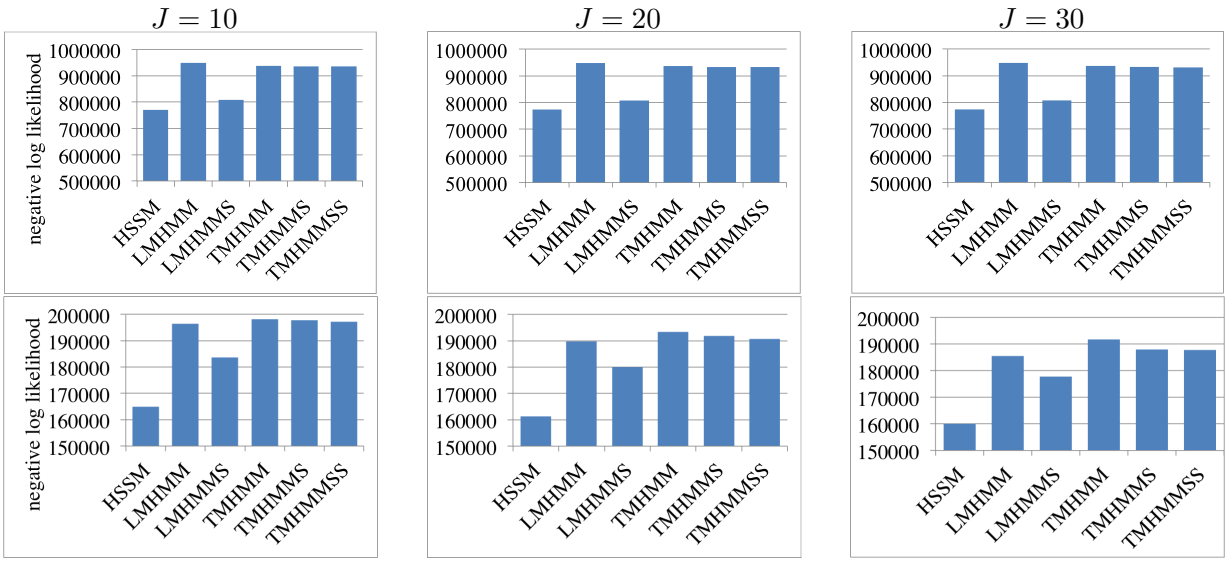


Figure 5: Negative log likelihood (smaller is better) on held-out datasets of Twitter-Post (upper) and AirTicketBooking (lower) under different number of latent states J.

ordering). This is the basic idea: for each dialogue session with n utterances in the test set, we firstly generate all $n!$ permutations of the utterances; then evaluate the probability of each permutation, and measure the similarity, i.e. Kendall's τ, between the max-probability permutation and the original order; finally, we average τ values for all dialogue sessions as the model's ordering test score. As pointed out by Zhai et al. (2014), it's however infeasible to enumerate all possible permutations of dialogue sessions when the number of utterances in large. In experiments, we employ the incrementally adding permutation strategy, as used by Zhai et al. (2014), to build up the permutation set. The results of ordering test are presented in Figure 6. We can see that HSSM exhibits better performance than all the other models. For the conventional models, it is interesting that LMHMMS, TMHMMS and TMHMMSS achieve worse performances than LMHMM and TMHMM. This is likely because the latter two models allow words to be emitted only from latent states (Zhai and Williams, 2014), while the former three models allow words to be generated from additional sources. This also implies HSSM's effectiveness of modeling distinct information uderlying dialogues.

4.3 Discussion

The expermental results illustrate the effectiveness of the proposed undirected dialogue structure analysis model based on Boltzmann machine. The conducted experiments also demonstrate that undirected models have three main merits for text modeling, which are also demonstrated by Hinton and Salakhutdinov (2009), Srivastava et al. (2013) through other tasks. Boltzmann machine based undirected models are able to generalize much better than traditional directed generative model; and model learning is more stable. Besides, an undirected model is more suitable for describing complex dependencies between different kinds of variables.

We also notice that all the models can, to some degree, capture the sequential structure in the dialogues, however, each model has a special characteristic which makes itself fit a certain kind of dataset better. HSSM and LMHMMS are more appropriate for modeling the open domain dataset, such as Twitter-Post used in this paper, and the task-oriented domain dataset with one relatively concentrated topic in the corpus and special information for each dialogue, such as AirTicketBooking. As we known, dialogue specific topics in HSSM or LMHMMS are used and trained only within corresponding dialogues. They are crucial for absorbing certain words that have important meaning but do not belongs to latent states. In addition, for differet dataset, dialogue specific topics may have different effect to the modeling. Take the Twitter-Post for an example, dialogue specific topics formulate actual themes of dialogues, such as a pop song, a sport news. As for the AirTicketBooking dataset, dialogue specific

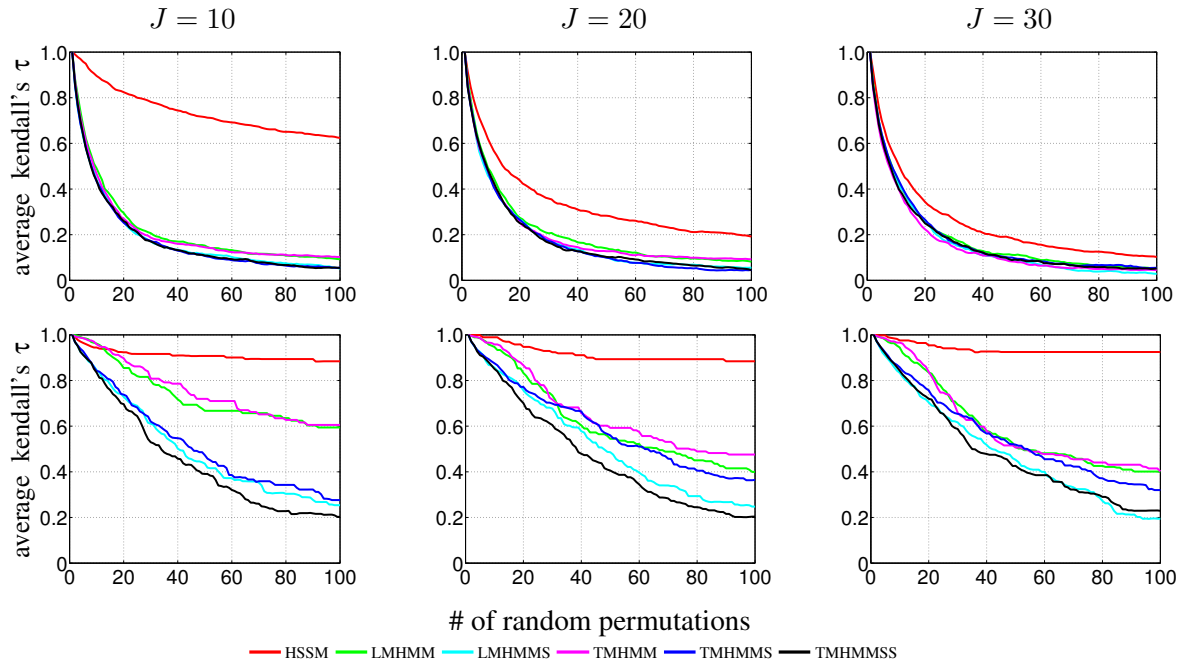


Figure 6: Average Kendall's τ measure (larger is better) on held-out datasets of Twitter-Post (upper) and AirTicketBooking (lower) under different number of latent states J.

topics always represent some special information, such as the personal information, including name, phone number, birthday, etc. In summary, each dialogue specific topic reflects special information which is different from other dialogues.

The three models, TMHMM, TMHMMS and TMHMMSS, which do not include dialogue specific topics, should be utilized on the task-oriented domain dataset, in which each dialogue has little special or personnal information. For example, the three models perform well on the the BusTime and TechSupport datasets (Zhai and Williams, 2014), in which name entities are all replaced by different semantic types (e.g. phone numbers are replaced by "<phone>", E-mail addresses are replaced by "<email>", etc).

5 Conclusions

We develope an undirected generative model, HSSM, for dialogue structure analysis, and examine the effectiveness of our model on two different datasets, Twitter posts occurred in open-domain and task-oriented dialogues from airline ticket booking domain. Qualitative evaluations and quantitative experimental results demonstrate that the proposed model achieves better performance than state-of-the-art approaches. Compared with traditional models, the proposed HSSM has more powerful ability of discovering structures of latent states and modeling different word sources, including latent states, dialogue specific topics and global general topic.

According to recent study (Srivastava et al., 2013), a deep network model exhibits much benefits for latent variable learning. A dialogue may actually have a hierarchy structure of latent states, therefore the proposed model can be extended to a deep model to capture more complex structures. Another possible way to extend the model is to consider modeling long distance dependency between latent states. This may further improve the model's performance.

Acknowledgments

We are grateful to anonymous reviewers for their helpful comments and suggestions. We would like to thank Alan Ritter for kindly providing the raw Twitter dataset.

This work is supported in part by the National Natural Science Funds of China under Grant 61170197 and 61571266, and in part by the Electronic Information Industry Development Fund under project "The R&D and Industrialization on Information Retrieval System Based on Man-Machine Interaction with Natural Speech".

References

James Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary Swift, and William Taysom. 2007. Plow: A collaborative task learning agent. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1514. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

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

Ananlada Chotimongkol. 2008. *Learning the structure of task-oriented conversations from the corpus of in-domain dialogs*. Ph.D. thesis, SRI International.

William W Cohen, Vitor R Carvalho, and Tom M Mitchell. 2004. Learning to classify email into"speech acts". In *EMNLP*, pages 309–316.

Nigel Crook, Ramon Granell, and Stephen Pulman. 2009. Unsupervised classification of dialogue acts using a dirichlet process mixture model. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 341–348. Association for Computational Linguistics.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2009. Replicated softmax: an undirected topic model. In *Advances in neural information processing systems*, pages 1607–1614.

Geoffrey E Hinton and Richard S Zemel. 1994. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pages 3–3.

Geoffrey E Hinton. 2002. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

Minwoo Jeong, Chin-Yew Lin, and Gary Geunbae Lee. 2009. Semi-supervised speech act recognition in emails and forums. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pages 1250–1259. Association for Computational Linguistics.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. 1999. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233.

Maurice G Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Jingjing Liu, Stephanie Seneff, and Victor Zue. 2010. Dialogue-oriented review summary generation for spoken dialogue recommendation systems. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 64–72. Association for Computational Linguistics.

Gabriel Murray, Steve Renals, Jean Carletta, and Johanna Moore. 2006. Incorporating speaker and discourse features into speech summarization. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pages 367–374. Association for Computational Linguistics.

Radford M Neal and Geoffrey E Hinton. 1998. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer.

Radford M Neal. 2001. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139.

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

Ruslan Salakhutdinov and Iain Murray. 2008. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning*, pages 872–879. ACM.

Nitish Srivastava, Ruslan R Salakhutdinov, and Geoffrey E Hinton. 2013. Modeling documents with deep boltzmann machines. *UAI*.

Tijmen Tieleman. 2008. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM.

Hanna M Wallach, Iain Murray, Ruslan Salakhutdinov, and David Mimno. 2009. Evaluation methods for topic models. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1105–1112. ACM.

Yorick Wilks. 2006. Artificial companions as a new kind of interface to the future internet.

Steve J Young. 2006. Using pomdps for dialog management. In *SLT*, pages 8–13.

Ke Zhai and Jason D Williams. 2014. Discovering latent structure in task-oriented dialogues. In *ACL (1)*, pages 36–46.

Summarizing Source Code using a Neural Attention Model

Srinivasan Iyer **Ioannis Konstas** **Alvin Cheung** **Luke Zettlemoyer**
Computer Science & Engineering
University of Washington
Seattle, WA 98195
{sviyer,ikonstas,akcheung,lsz}@cs.washington.edu

Abstract

High quality source code is often paired with high level summaries of the computation it performs, for example in code documentation or in descriptions posted in online forums. Such summaries are extremely useful for applications such as code search but are expensive to manually author, hence only done for a small fraction of all code that is produced. In this paper, we present the first completely data-driven approach for generating high level summaries of source code. Our model, CODE-NN , uses Long Short Term Memory (LSTM) networks with attention to produce sentences that describe C# code snippets and SQL queries. CODE-NN is trained on a new corpus that is automatically collected from StackOverflow, which we release. Experiments demonstrate strong performance on two tasks: (1) code summarization, where we establish the first end-to-end learning results and outperform strong baselines, and (2) code retrieval, where our learned model improves the state of the art on a recently introduced C# benchmark by a large margin.

1 Introduction

Billions of lines of source code reside in online repositories (Dyer et al., 2013), and high quality code is often coupled with natural language (NL) in the form of instructions, comments, and documentation. Short summaries of the overall computation the code performs provide a particularly useful form of documentation for a range of applications, such as code search or tutorials. However, such summaries are expensive to manually author.

1. Source Code (C#):

```
public int TextWidth(string text) {
  TextBlock t = new TextBlock();
  t.Text = text;
  return
    (int)Math.Ceiling(t.ActualWidth);
}
```

Descriptions:
a. Get rendered width of string rounded up to the nearest integer
b. Compute the actual textwidth inside a textblock

2. Source Code (C#):

```
  var input = "Hello";
  var regEx = new Regex("World");
  return !regEx.IsMatch(input);
```

Descriptions:
a. Return if the input doesn't contain a particular word in it
b. Lookup a substring in a string using regex

3. Source Code (SQL):

```
SELECT Max(marks) FROM stud_records
  WHERE marks <
  (SELECT Max(marks) FROM stud_records);
```

Descriptions:
a. Get the second largest value of a column
b. Retrieve the next max record in a table

Figure 1: Code snippets in C# and SQL and their summaries in NL, from StackOverflow. Our goal is to automatically generate summaries from code snippets.

As a result, this laborious process is only done for a small fraction of all code that is produced.

In this paper, we present the first completely data-driven approach for generating short high-level summaries of source code snippets in natural language. We focus on C#, a general-purpose imperative language, and SQL, a declarative language for querying databases. Figure 1 shows example code snippets with descriptions that summarize the overall function of the code, with the goal to generate high level descriptions, such as

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2073–2083,
Berlin, Germany, August 7-12, 2016.

lookup a substring in a string. Generating such a summary is often challenging because the text can include complex, non-local aspects of the code (e.g., consider the phrase 'second largest' in Example 3 in Figure 1). In addition to being directly useful for interpreting uncommented code, high-quality generation models can also be used for code retrieval, and in turn, for natural language programming by applying nearest neighbor techniques to a large corpus of automatically summarized code.

Natural language generation has traditionally been addressed as a pipeline of modules that decide 'what to say' (content selection) and 'how to say it' (realization) separately (Reiter and Dale, 2000; Wong and Mooney, 2007; Chen et al., 2010; Lu and Ng, 2011). Such approaches require supervision at each stage and do not scale well to large domains. We instead propose an end-to-end neural network called CODE-NN that jointly performs content selection using an attention mechanism, and surface realization using Long Short Term Memory (LSTM) networks. The system generates a summary one word at a time, guided by an attention mechanism over embeddings of the source code, and by context from previously generated words provided by a LSTM network (Hochreiter and Schmidhuber, 1997). The simplicity of the model allows it to be learned from the training data without the burden of feature engineering (Angeli et al., 2010) or the use of an expensive approximate decoding algorithm (Konstas and Lapata, 2013).

Our model is trained on a new dataset of code snippets with short descriptions, created using data gathered from Stackoverflow,[1] a popular programming help website. Since access is open and unrestricted, the content is inherently noisy (ungrammatical, non-parsable, lacking content), but as we will see, it still provides strong signal for learning. To reliably evaluate our model, we also collect a clean, human-annotated test set.[2]

We evaluate CODE-NN on two tasks: code summarization and code retrieval (Section 2). For summarization, we evaluate using automatic metrics such as METEOR and BLEU-4, together with a human study for naturalness and informativeness of the output. The results show that CODE-NN outperforms a number of strong baselines and, to the best of our knowledge, CODE-NN is the first approach that learns to generate summaries of source code from easily gathered online data. We further use CODE-NN for code retrieval for programming related questions on a recent C# benchmark, and results show that CODE-NN improves the state of the art (Allamanis et al. (2015b)) for mean reciprocal rank (MRR) by a wide margin.

[1] `http://stackoverflow.com`

[2] Data and code are available at `https://github.com/sriniiyer/codenn`.

2 Tasks

CODE-NN generates a NL summary of source code snippets (GEN task). We have also used CODE-NN on the inverse task to retrieve source code given a question in NL (RET task).

Formally, let U_C be the set of all code snippets and U_N be the set of all summaries in NL. For a training corpus with J code snippet and summary pairs $(c_j, n_j), 1 \leq j \leq J, c_j \in U_C, n_j \in U_N$, we define the following two tasks:

GEN For a given code snippet $c \in U_C$, the goal is to produce a NL sentence $n^* \in U_N$ that maximizes some scoring function $s \in (U_C \times U_N \rightarrow \mathbb{R})$:

$$n^* = \underset{n}{\operatorname{argmax}}\ s(c, n) \tag{1}$$

RET We also use the scoring function s to retrieve the highest scoring code snippet c_j^* from our training corpus, given a NL question $n \in U_N$:

$$c_j^* = \underset{c_j}{\operatorname{argmax}}\ s(c_j, n), 1 \leq j \leq J \tag{2}$$

In this work, s is computed using an LSTM neural attention model, to be described in Section 5.

3 Related Work

Although we focus on generating high-level summaries of source code snippets, there has been work on producing code descriptions at other levels of abstraction. Movshovitz-Attias and Cohen (2013) study the task of predicting class-level comments by learning n-gram and topic models from open source Java projects and testing it using a character-saving metric on existing comments. Allamanis et al. (2015a) create models for suggesting method and class names by embedding them in a high dimensional continuous space. Sridhara et al. (2010) present a pipeline that generates summaries of Java methods by selecting relevant content and generating phrases using templates to describe them. There is also work on improving program comprehension (Haiduc et al.,

2010), identifying cross-cutting source code concerns (Rastkar et al., 2011), and summarizing software bug reports (Rastkar et al., 2010). To the best of our knowledge, we are the first to use learning techniques to construct completely new sentences from arbitrary code snippets.

Source code summarization is also related to generation from formal meaning representations. Wong and Mooney (2007) present a system that learns to generate sentences from lambda calculus expressions by inverting a semantic parser. Mei et al. (2016), Konstas and Lapata (2013), and Angeli et al. (2010) create learning algorithms for text generation from database records, again assuming data that pairs sentences with formal meaning representations. In contrast, we present algorithms for learning from easily gathered web data.

In the database community, Simitsis and Ioannidis (2009) recognize the need for SQL database systems to talk back to users. Koutrika et al. (2010) built an interactive system (LOGOS) that translates SQL queries to text using NL templates and database schemas. Similarly there has been work on translating SPARQL queries to natural language using rules to create dependency trees for each section of the query, followed by a transformation step to make the output more natural (Ngonga Ngomo et al., 2013). These approaches are not learning based, and require significant manual template-engineering efforts.

We use recurrent neural networks (RNN) based on LSTMs and neural attention to jointly model source code and NL. Recently, RNN-based approaches have gained popularity for text generation and have been used in machine translation (Sutskever et al., 2011), image and video description (Karpathy and Li, 2015; Venugopalan et al., 2015; Devlin et al., 2015), sentence summarization (Rush et al., 2015), and Chinese poetry generation (Zhang and Lapata, 2014). Perhaps most closely related, Wen et al. (2015) generate text for spoken dialogue systems with a two-stage approach, comprising an LSTM decoder semantically conditioned on the logical representation of speech acts, and a reranker to generate the final output. In contrast, we design an end-to-end attention-based model for source code.

For code retrieval, Allamanis et al. (2015b) proposed a system that uses Stackoverflow data and web search logs to create models for retrieving C# code snippets given NL questions and vice versa. They construct distributional representations of code structure and language and combine them using additive and multiplicative models to score (code, language) pairs, an approach that could work well for retrieval but cannot be used for generation. We learn a neural generation model without using search logs and show that it can also be used to score code for retrieval, with much higher accuracy.

Synthesizing code from language is an alternative to code retrieval and has been studied in both the Systems and NLP research communities. Giordani and Moschitti (2012), Li and Jagadish (2014), and Gulwani and Marron (2014) synthesize source code from NL queries for database and spreadsheet applications. Similarly, Lei et al. (2013) interpret NL instructions to machine-executable code, and Kushman and Barzilay (2013) convert language to regular expressions. Unlike most synthesis methods, CODE-NN is domain agnostic, as we demonstrate its applications on both C# and SQL.

4 Dataset

We collected data from StackOverflow (SO), a popular website for posting programming-related questions. Anonymized versions of all the posts can be freely downloaded.[3] Each post can have multiple tags. Using the *C#* tag for C# and the *sql*, *database* and *oracle* tags for SQL, we were able to collect 934,464 and 977,623 posts respectively.[4] Each post comprises a short title, a detailed question, and one or more responses, of which one can be marked as accepted. We found that the text in the question and responses is domain-specific and verbose, mixed with details that are irrelevant for our tasks. Also, code snippets in responses that were not accepted were frequently incorrect or tangential to the question asked. Thus, we extracted only the title from the post and use the code snippet from those accepted answers that contain exactly one code snippet (using <code> tags). We add the resulting (title, query) pairs to our corpus, resulting in a total of 145,841 pairs for C# and 41,340 pairs for SQL.

Cleaning We train a semi-supervised classifier to filter titles like *'Difficult C# if then logic'* or *'How can I make this query easier to write?'* that bear no relation to the corresponding code snippet.

[3] http://archive.org/details/stackexchange

[4] The data was downloaded in Dec 2014.

To do so, we annotate 100 titles as being *clean* or *not clean* for each language and use them to bootstrap the algorithm. We then use the remaining titles in our training set as an unsupervised signal, and obtain a classification accuracy of over 73% on a manually labeled test set for both languages. For the final dataset, we retain 66,015 C# (title, query) pairs and 32,337 SQL pairs that are classified as clean, and use 80% of these datasets for training, 10% for validation and 10% for testing.

Parsing Given the informal nature of StackOverflow, the code snippets are approximate answers that are usually incomplete. For example, we observe that only 12% of the SQL queries parse without any syntactic errors (using zql[5]). We therefore aim to perform a best-effort parse of the code snippet, using modified versions of an ANTLR parser for C# (Parr, 2013) and *python-sqlparse* (Albrecht, 2015) for SQL. We strip out all comments and to avoid being context specific, we replace literals with tokens denoting their types. In addition, for SQL, we replace table and column names with numbered placeholder tokens while preserving any dependencies in the query. For example, the SQL query in Figure 1 is represented as `SELECT MAX(col0) FROM tab0 WHERE col0 < (SELECT MAX(col0) FROM tab0)`.

Data Statistics The structural complexity and size of the code snippets in our dataset makes our tasks challenging. More than 40% of our C# corpus comprises snippets with three or more statements and functions, and 20% contains loops and conditionals. Also, over a third of our SQL queries contain one or more subqueries and multiple tables, columns and functions (like `MIN`, `MAX`, `SUM`). On average, our C# snippets are 38 tokens long and the queries in our corpus are 46 tokens long, while titles are 9-12 words long. Table 2 shows the complete data statistics.

Human Annotation For the GEN task, we use n-gram based metrics (see Section 6.1.2) of the summary generated by our model with respect to the actual title in our corpus. Titles can be short, and a given code snippet can be described in many different ways with little overlapping content between them. For example, the descriptions for the second code snippet in Figure 1 share very few words with each other. To address these limitations, we extend our test set by asking human annotators to provide two additional titles for 200 snippets chosen at random from the test set, making a total of three reference titles for each code snippet. To collect this data, annotators were shown only the code snippets and were asked to write a short summary after looking at a few example summaries. They were also asked to "think of a question that they could ask on a programming help website, to get the code snippet as a response." This encouraged them to briefly describe the key feature that the code is trying to demonstrate. We use half of this test set for model tuning (DEV, see Section 5) and the rest for evaluation (EVAL).

[5] `http://zql.sourceforge.net`

C#	**# Statements**		**# Functions**	
	≥ 3	23,611 (44.7%)	≥ 3	26,541 (51.0%)
	≥ 4	17,822 (33.7%)	≥ 4	20,221 (38.2%)
	# Loops		**# Conditionals**	
	≥ 1	10,676 (20.0%)	≥ 1	11,819 (22.3%)
SQL	**# Subqueries**		**# Tables**	
	≥ 1	11,418 (35%)	≥ 3	14,695 (44%)
	≥ 2	3,625 (11%)	≥ 4	10,377 (31%)
	# Columns		**# Functions**	
	≥ 5	12,366 (37%)	≥ 3	6,290 (19%)
	≥ 6	9,050 (27%)	≥ 4	3,973 (12%)

Table 1: Statistics for code snippets in our dataset.

C#	**Avg. code length**	38 tokens	**# tokens**	91,156
	Avg. title length	12 words	**# words**	24,857
SQL	**Avg. query length**	46 tokens	**# tokens**	1,287
	Avg. title length	9 words	**# words**	10,086

Table 2: Average code and title lengths together with vocabulary sizes for C# and SQL after post-processing.

5 The CODE-NN Model

Description We present an end-to-end generation system that performs content selection and surface realization jointly. Our approach uses an attention-based neural network to model the conditional distribution of a NL summary n given a code snippet c. Specifically, we use an LSTM model that is guided by attention on the source code snippet to generate a summary one word at a time, as shown in Figure 2.[6]

Formally, we represent a NL summary $n = n_1, \ldots, n_l$ as a sequence of 1-hot vectors

[6] We experimented with other sequence (Sutskever et al., 2014) and tree based architectures (Tai et al., 2015) as well. None of these models significantly improved performance, however, this is an important area for future work.

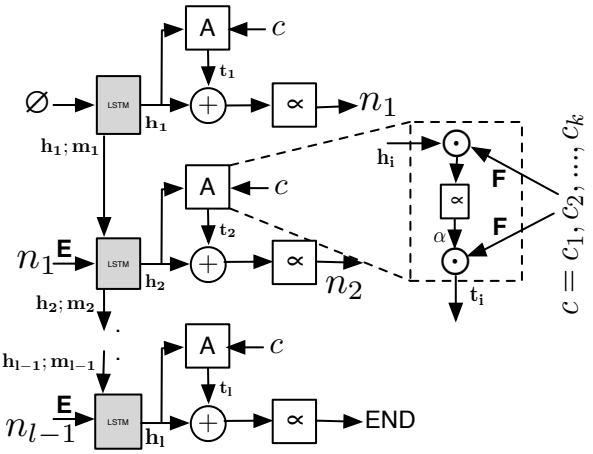


Figure 2: Generation of a title $n = n_1, \ldots,$ END given code snippet $c_1, ..., c_k$. The attention cell computes a distributional representation t_i of the code snippet based on the current LSTM hidden state h_i. A combination of t_i and h_i is used to generate the next word, n_i, which feeds back into the next LSTM cell. This is repeated until a fixed number of words or END is generated. $\propto$ blocks denote softmax operations.

$\mathbf{n_1}, \ldots, \mathbf{n_l} \in \{0,1\}^{|N|}$, where N is the vocabulary of the summaries. Our model computes the probability of n (scoring function s in Eq. 1) as a product of the conditional next-word probabilities

$$s(c, n) = \prod_{i=1}^{l} p(n_i | n_1, \ldots, n_{i-1})$$

with,

$$p(n_i | n_1, \ldots, n_{i-1}) \propto \mathbf{W} \tanh(\mathbf{W_1 h_i} + \mathbf{W_2 t_i})$$

where, $\mathbf{W} \in \mathbb{R}^{|N| \times H}$ and $\mathbf{W_1}, \mathbf{W_2} \in \mathbb{R}^{H \times H}$, H being the embedding dimensionality of the summaries. $\mathbf{t_i}$ is the contribution from the attention model on the source code (see below). $\mathbf{h_i}$ represents the hidden state of the LSTM cell at the current time step and is computed based on the previously generated word, the previous LSTM cell state $\mathbf{m_{i-1}}$ and the previous LSTM hidden state $\mathbf{h_{i-1}}$ as

$$\mathbf{m_i}; \mathbf{h_i} = f(\mathbf{n_{i-1}E}, \mathbf{m_{i-1}}, \mathbf{h_{i-1}}; \theta)$$

where $\mathbf{E} \in \mathbb{R}^{|N| \times H}$ is a word embedding matrix for the summaries. We compute f using the LSTM cell architecture used by Zaremba et al. (2014).

Attention The generation of each word is guided by a global attention model (Luong et al., 2015), which computes a weighted sum of the embeddings of the code snippet tokens based on the current LSTM state (see right part in Figure 2). Formally, we represent c as a set of 1-hot vectors $\mathbf{c_1}, \ldots, \mathbf{c_k} \in \{0,1\}^{|C|}$ for each source code token; C is the vocabulary of all tokens in our code snippets. Our attention model computes,

$$\mathbf{t_i} = \sum_{j=1}^{k} \alpha_{i,j} \cdot \mathbf{c_j F}$$

where $\mathbf{F} \in \mathbb{R}^{|C| \times H}$ is a token embedding matrix and each $\alpha_{i,j}$ is proportional to the dot product between the current internal LSTM hidden state $\mathbf{h_i}$ and the corresponding token embedding $\mathbf{c_j}$:

$$\alpha_{i,j} = \frac{\exp(\mathbf{h_i}^{\mathrm{T}} \mathbf{c_j F})}{\sum_{j=1}^{k} \exp(\mathbf{h_i}^{\mathrm{T}} \mathbf{c_j F})}$$

Training We perform supervised end-to-end training using backpropagation (Werbos, 1990) to learn the parameters of the embedding matrices $\mathbf{F}$ and $\mathbf{E}$, transformation matrices $\mathbf{W}$, $\mathbf{W_1}$ and $\mathbf{W_2}$, and parameters θ of the LSTM cell that computes f. We use multiple epochs of minibatch stochastic gradient descent and update all parameters to minimize the negative log likelihood (NLL) of our training set. To prevent over-fitting we make use of dropout layers (Srivastava et al., 2014) at the summary embeddings and the output softmax layer. Using pre-trained embeddings (Mikolov et al., (2013)) for the summary embedding matrix or adding additional LSTM layers did not improve performance for the GEN task. Since the NLL training objective does not directly optimize for our evaluation metric (METEOR), we compute METEOR (see Section 6.1.2) on a small development set (DEV) after every epoch and save the intermediate model that gives the maximum score, as the final model.

Decoding Given a trained model and an input code snippet c, finding the most optimal title entails generating the title n^* that maximizes $s(c, n)$ (see Eq. 1). We approximate n^* by performing beam search on the space of all possible summaries using the model output.

Implementation Details We add special START and END tokens to our training sequences and replace all tokens and output words occurring

with a frequency of less than 3 with an UNK token, making $|C| = 31,667$ and $|N| = 7,470$ for C# and $|C| = 747$ and $|N| = 2,506$ for SQL. Our hyper-parameters are set based on performance on the validation set. We use a minibatch size of 100 and set the dimensionality of the LSTM hidden states, token embeddings, and summary embeddings (H) to 400. We initialize all model parameters uniformly between -0.35 and 0.35. We start with a learning rate of 0.5 and start decaying it by a factor of 0.8 after 60 epochs if accuracy on the validation set goes down, and terminate training when the learning rate goes below 0.001. We cap the parameter gradients to 5 and use a dropout rate of 0.5.

We use the Torch framework[7] to train our models on GPUs. Training runs for about 80 epochs and takes approximately 7 hours. We compute METEOR score at every epoch on the development set (DEV) to choose the best final model, with the best results obtained between 60 and 70 epochs. For decoding, we set the beam size to 10, and the maximum summary length to 20 words.

6 Experimental Setup

6.1 GEN Task

6.1.1 Baselines

For the GEN task, we compare CODE-NN with a number of competitive systems, none of which had been previously applied to generate text from source code, and hence we adapt them slightly for this task, as explained below.

IR is an information retrieval baseline that outputs the title associated with the code c_j in the training set that is closest to the input code c in terms of token Levenshtein distance. In this case s from Eq.1 becomes,

$$s(c, n_j) = -1 \times \text{lev}(c_j, c), 1 \leq j \leq J$$

MOSES (Koehn et al., 2007) is a popular phrase-based machine translation system. We perform generation by treating the tokenized code snippet as the source language, and the title as the target. We train a 3-gram language model using KenLM (Heafield, 2011) to use with MOSES, and perform MIRA-based tuning (Cherry and Foster, 2012) of hyper-parameters using DEV.

SUM-NN is the neural attention-based abstractive summarization model of Rush et al. (2015). It uses an encoder-decoder architecture with an attention mechanism based on a fixed context window of previously generated words. The decoder is a feed-forward neural language model that generates the next word based on previous words in a context window of size k. In contrast, we decode using an LSTM network that can model long range dependencies and our attention weights are tied to the LSTM hidden states. We set the embedding and hidden state dimensions and context window size by tuning on our validation set. We found this model to generate overly short titles like *'sql server 2008'* when a length restriction was not imposed on the output text. Therefore, we fix the output length to be the average title length in the training set while decoding.

[7] http://torch.ch

6.1.2 Evaluation Metrics

We evaluate the GEN task using automatic metrics, and also perform a human study.

Automatic Evaluation We report METEOR (Banerjee and Lavie, 2005) and sentence level BLEU-4 (Papineni et al., 2002) scores. METEOR is recall-oriented and measures how well our model captures content from the references in our output. BLEU-4 measures the average n-gram precision on a set of reference sentences, with a penalty for overly short sentences. Since the generated summaries are short and there are multiple alternate summaries for a given code snippet, higher order n-grams may not overlap. We remedy this problem by using +1 smoothing (Lin and Och, 2004). We compute these metrics on the tuning set DEV and the held-out evaluation set EVAL.

Human Evaluation Since automatic metrics do not always agree with the actual quality of the results (Stent et al., 2005), we perform human evaluation studies to measure the output of our system and baselines across two modalities, namely naturalness and informativeness. For the former, we asked 5 native English speakers to rate each title against grammaticality and fluency, on a scale between 1 and 5. For *informativeness* (i.e., the amount of content carried over from the input code to the NL summary, ignoring fluency of the text), we asked 5 human evaluators familiar with C# and SQL to evaluate the system output by rating the factual overlap of the summary with the reference titles, on a scale between 1 and 5.

6.2 RET task

6.2.1 Model and Baselines

CODE-NN As described in Section 2, for a given NL question n in the RET task, we rank all code snippets c_j in our corpus by computing the scoring function $s(c_j, n)$, and return the query c_j^* that maximizes it (Eq. 2).

RET-IR is an information retrieval baseline that ranks the candidate code snippets using cosine similarity between the given NL question n and all summaries n_j in the retrieval set, based on their vector representations using TF-IDF weights over unigrams. The scoring function s in Eq. 2 becomes:

$$s(c_j, n) = \frac{\text{tf-idf}(n_j) \cdot \text{tf-idf}(n)}{\|\text{tf-idf}(n_j)\|\|\text{tf-idf}(n)\|}, 1 \leq j \leq J$$

6.2.2 Evaluation Metrics

We assess ranking quality by computing the Mean Reciprocal Rank (MRR) of c_j^*. For every snippet c_j in EVAL (and DEV), we use two of the three references (title and human annotation), namely $n_{j,1}, n_{j,2}$. We then build a retrieval set comprising $(c_j, n_{j,1})$ together with 49 random distractor pairs $(c', n'), c' \neq c_j$ from the test set. Using $n_{j,2}$ as the natural language question, we rank all 50 items in this retrieval set and use the rank of query c_j^* to compute MRR. We average MRR over all returned queries c_j^* in the test set, and repeat this experiment for several different random sets of distractors.

6.3 Tasks from Allamanis et al. (2015b)

Allamanis et al. (2015b) take a retrieval approach to answer C# related natural language questions (L to C), similar to our RET task. In addition, they also use retrieval to summarize C# source code (C to L) and evaluate both tasks using the MRR metric. Although they also use data from Stackoverflow, their dataset preparation and cleaning methods differs significantly from ours. For example, they filter out posts where the question has fewer than 2 votes, the answer has fewer than 3 votes, or the post has fewer than 1000 views. Additionally, they also filter code snippets that cannot be parsed by Roslyn (.NET compiler) or are longer than 300 characters. Thus, to directly compare with their model, we re-train our generation model on their dataset and use our model score for retrieval of both code and summaries.

	Model	METEOR	BLEU-4
C#	IR	7.9 (6.1)	13.7 (12.6)
	MOSES	9.1 (9.7)	11.6 (11.5)
	SUM-NN	10.6 (10.3)	19.3 (18.2)
	CODE-NN	**12.3 (13.4)**	**20.5 (20.4)**
SQL	IR	6.3 (8.0)	13.5 (13.0)
	MOSES	8.3 (9.7)	15.4 (15.9)
	SUM-NN	6.4 (8.7)	13.3 (14.2)
	CODE-NN	**10.9 (14.0)**	**18.4 (17.0)**

Table 3: Performance on EVAL for the GEN task. Performance on DEV is indicated in parentheses.

	Model	Naturalness	Informativeness
C#	IR	3.42	2.25
	MOSES	1.41	2.42
	SUM-NN	**4.61***	1.99
	CODE-NN	**4.48**	**2.83**
SQL	IR	3.21	2.58
	MOSES	2.80	2.54
	SUM-NN	4.44	2.75
	CODE-NN	**4.54**	**3.12**

Table 4: Naturalness and Informativeness measures of model outputs. Stat. sig. between CODE-NN and others is computed with a 2-tailed Student's t-test; $p < 0.05$ except for *.

7 Results

7.1 GEN Task

Table 3 shows automatic evaluation metrics for our model and baselines. CODE-NN outperforms all the other methods in terms of METEOR and BLEU-4 score. We attribute this to its ability to perform better content selection, focusing on the more salient parts of the code by using its attention mechanism jointly with its LSTM memory cells. The neural models have better performance on C# than SQL. This is in part because, unlike SQL, C# code contains informative intermediate variable names that are directly related to the objective of the code. On the other hand, SQL is more challenging in that it only has a handful of keywords and functions, and summarization models need to rely on other structural aspects of the code.

Informativeness and naturalness scores for each model from our human evaluation study are presented in Table 4. In general, CODE-NN performs well across both dimensions. Its superior performance in terms of informativeness further supports our claim that it manages to select content more effectively. Although SUM-NN performs similar to CODE-NN on naturalness, its output lacks content and has very little variation (see Section 7.4), which also explains its surprisingly low

	Model	MRR
C#	RET-IR	0.42 ± 0.02 (0.44 ± 0.01)
	CODE-NN	$\mathbf{0.58 \pm 0.01}$ $(\mathbf{0.66 \pm 0.02})$
SQL	RET-IR	0.28 ± 0.01 (0.4 ± 0.01)
	CODE-NN	$\mathbf{0.44 \pm 0.01}$ $(\mathbf{0.54 \pm 0.02})$

Table 5: MRR for the RET task. Dev set results in parentheses.

	Model	MRR
L to C	Allamanis	0.182 ± 0.009
	CODE-NN	$\mathbf{0.590 \pm 0.044}$
C to L	Allamanis	0.434 ± 0.003
	CODE-NN	$\mathbf{0.461 \pm 0.046}$

Table 6: MRR values for the Language to Code (L to C) and the Code to Language (C to L) tasks using the C# dataset of Allamanis et al. (2015b)

score on informativeness.

7.2 RET Task

Table 5 shows the MRR on the RET task for CODE-NN and RET-IR, averaged over 20 runs for C# and SQL. CODE-NN outperforms the baseline by about 16% for C# and SQL. RET-IR can only output code snippets that are annotated with NL as potential matches. On the other hand, CODE-NN can rank even unannotated code snippets and nominate them as potential candidates. Hence, it can leverage vast amounts of such code available in online repositories like Github. To speed up retrieval when using CODE-NN , it could be one of the later stages in a multi-stage retrieval system and candidates may also be ranked in parallel.

7.3 Comparison with Allamanis et al.

We train CODE-NN on their dataset and evaluate using the same MRR testing framework (see Table 6). Our model performs significantly better for the Language to Code task (L to C) and slightly better for the Code to Language task (C to L). The attention mechanism together with the LSTM network is able to generate better scores for (language, code) pairs.

7.4 Qualitative Analysis

Figure 3 shows the relative magnitudes of the attention weights ($\alpha_{i,j}$) for example C# and SQL code snippets while generating their corresponding summaries. Darker regions represent stronger weights. CODE-NN automatically learns to do high-quality content selection by aligning key summary words with informative tokens in the code snippet.

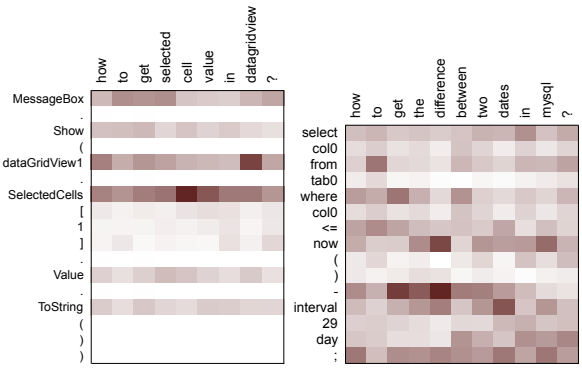


Figure 3: Heatmap of attention weights $\alpha_{i,j}$ for example C# (left) and SQL (right) code snippets. The model learns to align key summary words (like cell) with the corresponding tokens in the input (SelectedCells).

Table 8 shows examples of the output generated by our model and baselines for code snippets in DEV. Most of the models produce meaningful output for simple code snippets (first example) but degrade on longer, compositional inputs. For example, the last SQL query listed in Table 8 includes a subquery, where a complete description should include both summing and concatenation. CODE-NN describes the summation (but not concatenation), while others return non-relevant descriptions.

Finally, we performed manual error analysis on 50 randomly selected examples from DEV (Table 7) for each language. Redundancy is a major source of error, i.e., generation of extraneous content-bearing phrases, along with missing content, e.g., in the last example of Table 8 there is no reference to the concatenation operations present in the beginning of the query. Sometimes the output from our model can be out of context, in the sense that it does not match the input code. This often happens for low frequency tokens (7% of cases), for which CODE-NN realizes them with generic phrases. This also happens when there are very long range dependencies or compositional structures in the input, such as nested queries (13% of the cases).

8 Conclusion

In this paper, we presented CODE-NN , an end-to-end neural attention model using LSTMs to

Error	% Cases
Correct	37%
Redundancy	17%
Missing Info	26%
Out of context	20%

Table 7: Error analysis on 50 examples in DEV

generate summaries of C# and SQL code by learning from noisy online programming websites. Our model outperforms competitive baselines and achieves state of the art performance on automatic metrics, namely METEOR and BLEU, as well as on a human evaluation study. We also used CODE-NN to answer programming questions by retrieving the most appropriate code snippets from a corpus, and beat previous baselines for this task in terms of MRR. We have published our C# and SQL datasets, the accompanying human annotated test sets, and our code for the tasks described in this paper.

In future work, we plan to develop better models for capturing the structure of the input, as well as extend the use of our system to other applications such as automatic documentation of source code.

Acknowledgements

We thank Mike Lewis, Chloé Kiddon, Kenton Lee, Eunsol Choi and the anonymous reviewers for comments on an earlier version. We also thank Bill Howe, Dan Halperin and Mark Yatskar for helpful discussions and Miltiadis Allamanis for providing the dataset for the comparison study. This research was supported in part by the NSF (IIS-1252835), an Allen Distinguished Investigator Award, and a gift from Amazon.

References

Andi Albrecht. 2015. python-sqlparse.

Miltiadis Allamanis, Earl T Barr, Christian Bird, and Charles Sutton. 2015a. Suggesting accurate method and class names. In *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, pages 38–49.

Miltiadis Allamanis, Daniel Tarlow, Andrew Gordon, and Yi Wei. 2015b. Bimodal modelling of source code and natural language. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 2123–2132.

Gabor Angeli, Percy Liang, and Dan Klein. 2010. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512.

Method	Output

C# code	`var x = "FundList[10].Amount";`
	`int xIndex = Convert.ToInt32(`
	`  Regex.Match(x,@"\d+").Value);`

Gold	Identify the number in given string
IR	Convert string number to integer
MOSES	How to xIndex numbers in C#?
SUM-NN	How can I get the value of a string?
CODE-NN	How to convert string to int?

C# code	`foreach (string pTxt in xml.parent) {`
	`  TreeNode parent = new TreeNode();`
	`  foreach (string cTxt in xml.child) {`
	`  TreeNode child = new TreeNode();`
	`  parent.Nodes.Add(child);`
	`  }`
	`}`

Gold	Adding childs to a treenode dynamically in C#
IR	How to set the name of a tabPage programmatically
MOSES	How can TreeView nodes from XML parentText string to a treeview node
SUM-NN	How to get data from xml file in C#
CODE-NN	How to get all child nodes in TreeView?

C# code	`string url = baseUrl +`
	`  "/api/Entry/SendEmail?emailId="`
	`  + emailId;`
	`WebRequest req =`
	`  WebRequest.Create(url);`
	`req.Method = "GET";`
	`req.BeginGetResponse(null, null);`

Gold	Execute a get request on a web server and receive the response asynchronously
IR	How to download a file from another Sharepoint Domain
MOSES	How baseUrl emailId C how to a page in BeginGetResponse to
SUM-NN	How to get data from a file in C
CODE-NN	How to call a URL from a web api post ?
SQL Query	`SELECT * FROM table` `ORDER BY Rand() LIMIT 10`
Gold	Select random rows from mysql table
IR	How to select a random record from a mysql database?
MOSES	How to select all records in mysql ?
SUM-NN	How can I select random rows from a table
CODE-NN	How to get random rows from a mysql database?

SQL Query	`SELECT Group_concat(`
	`  Concat_ws(',', playerid, r1, r2)`
	`  SEPARATOR ';')`
	`FROM (SELECT playerid,`
	`  Sum(rank = 1) r1, Sum(rank < 5) r2`
	`  FROM result GROUP BY playerid) t;`

Gold	Get sum of group values based on condition and concatenate them into a string
IR	Mysql: counting occurences in a table, return as a single row
MOSES	Mysql query to get this result of the result of one column value in mysql
SUM-NN	How do i combine these two queries into one?
CODE-NN	How to get the sum of a column in a single query?

Table 8: Examples of outputs generated by each model for code snippets in DEV

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, volume 29, pages 65–72.

David L Chen, Joohyun Kim, and Raymond J Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, pages 397–435.

Colin Cherry and George Foster. 2012. Batch tuning strategies for statistical machine translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 427–436.

Jacob Devlin, Hao Cheng, Hao Fang, Saurabh Gupta, Li Deng, Xiaodong He, Geoffrey Zweig, and Margaret Mitchell. 2015. Language models for image captioning: The quirks and what works. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 100–105.

Robert Dyer, Hoan Anh Nguyen, Hridesh Rajan, and Tien N Nguyen. 2013. Boa: A language and infrastructure for analyzing ultra-large-scale software repositories. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 422–431.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to SQL queries with generative parsers discriminatively reranked. In *Proceedings of COLING 2012: Posters*, pages 401–410.

Sumit Gulwani and Mark Marron. 2014. Nlyze: Interactive programming by natural language for spreadsheet data analysis and manipulation. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 803–814.

Sonia Haiduc, Jairo Aponte, and Andrian Marcus. 2010. Supporting program comprehension with source code summarization. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 2*, pages 223–226.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197.

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

Andrej Karpathy and Fei-Fei Li. 2015. Deep visual-semantic alignments for generating image descriptions. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pages 3128–3137.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180.

Ioannis Konstas and Mirella Lapata. 2013. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48(1):305–346.

Georgia Koutrika, Alkis Simitsis, and Yannis E Ioannidis. 2010. Explaining structured queries in natural language. In *Data Engineering (ICDE), 2010 IEEE 26th International Conference on*, pages 333–344.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 826–836.

Tao Lei, Fan Long, Regina Barzilay, and Martin Rinard. 2013. From natural language specifications to program input parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1294–1303.

Fei Li and Hosagrahar V Jagadish. 2014. Nalir: An interactive natural language interface for querying relational databases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 709–712.

Chin-Yew Lin and Franz Josef Och. 2004. Orange: a method for evaluating automatic evaluation metrics for machine translation. In *Proceedings of the 20th international conference on Computational Linguistics*, page 501.

Wei Lu and Hwee Tou Ng. 2011. A probabilistic forest-to-string model for language generation from typed lambda calculus expressions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1611–1622.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2016. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations*.

Dana Movshovitz-Attias and William W. Cohen. 2013. Natural language models for predicting programming comments. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 35–40.

Axel-Cyrille Ngonga Ngomo, Lorenz Bühmann, Christina Unger, Jens Lehmann, and Daniel Gerber. 2013. Sorry, i don't speak sparql: Translating sparql queries into natural language. In *Proceedings of the 22Nd International Conference on World Wide Web*, pages 977–988.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Terence Parr. 2013. *The definitive ANTLR 4 reference*. Pragmatic Bookshelf.

Sarah Rastkar, Gail C Murphy, and Gabriel Murray. 2010. Summarizing software artifacts: a case study of bug reports. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 505–514.

Sarah Rastkar, Gail C Murphy, and Alexander WJ Bradley. 2011. Generating natural language summaries for crosscutting source code concerns. In *Software Maintenance (ICSM), 2011 27th IEEE International Conference on*, pages 103–112.

Ehud Reiter and Robert Dale. 2000. *Building natural language generation systems*. Cambridge University Press, New York, NY.

Alexander M. Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

Alkis Simitsis and Yannis E. Ioannidis. 2009. Dbmss should talk back too. In *CIDR 2009, Fourth Biennial Conference on Innovative Data Systems Research, Online Proceedings*.

Giriprasad Sridhara, Emily Hill, Divya Muppaneni, Lori Pollock, and K Vijay-Shanker. 2010. Towards automatically generating summary comments for java methods. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, pages 43–52.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating evaluation methods for generation in the presence of variation. In *Computational Linguistics and Intelligent Text Processing*, pages 341–351.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

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

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015*, pages 1556–1566.

Subhashini Venugopalan, Huijuan Xu, Jeff Donahue, Marcus Rohrbach, Raymond J. Mooney, and Kate Saenko. 2015. Translating videos to natural language using deep recurrent neural networks. In *In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1494–1504.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721.

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

Yuk Wah Wong and Raymond J Mooney. 2007. Generation by inverting a semantic parser that uses statistical machine translation. In *In Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 172–179.

Wojciech Zaremba and Ilya Sutskever. 2014. Learning to execute. *CoRR*, abs/1410.4615.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 670–680.

Continuous Profile Models in ASL Syntactic Facial Expression Synthesis

Hernisa Kacorri
Carnegie Mellon University
Human-Computer Interaction Institute
5000 Forbes Avenue
Pittsburgh, PA 15213, USA
hkacorri@andrew.cmu.edu

Matt Huenerfauth
Rochester Institute of Technology
B. Thomas Golisano College of
Computing and Information Sciences
152 Lomb Memorial Drive
Rochester, NY 14623, USA
matt.huenerfauth@rit.edu

Abstract

To create accessible content for deaf users, we investigate automatically synthesizing animations of American Sign Language (ASL), including grammatically important facial expressions and head movements. Based on recordings of humans performing various types of syntactic face and head movements (which include idiosyncratic variation), we evaluate the efficacy of Continuous Profile Models (CPMs) at identifying an essential "latent trace" of the performance, for use in producing ASL animations. A metric-based evaluation and a study with deaf users indicated that this approach was more effective than a prior method for producing animations.

1 Introduction and Motivation

While there is much written content online, many people who are deaf have difficulty reading text or may prefer sign language. For example, in the U.S., standardized testing indicates that a majority of deaf high school graduates (age 18+) have a fourth-grade reading level or below (Traxler, 2000) (U.S. fourth-grade students are typically age 9). While it is possible to create video-recordings of a human performing American Sign Language (ASL) for use on websites, updating such material is expensive (i.e., re-recording). Thus, researchers investigate technology to automate the synthesis of animations of a signing virtual human, to make it more cost-effective for organizations to provide sign language content online that is easily updated and maintained. Animations can be automatically synthesized from a symbolic specification of the message authored by a human or perhaps by machine translation, e.g. (Ebling and Glauert, 2013; Filhol et al., 2013; Stein et al., 2012).

1.1 ASL Syntactic Facial Expressions

Facial expressions are essential in ASL, conveying emotion, semantic variations, and syntactic structure. Prior research has verified that ASL animations with missing or poor facial expressions are significantly less understandable for deaf users (Kacorri et al., 2014; Kacorri et al., 2013b; Kacorri et al., 2013a). While artists can produce individual animations with beautiful expressions, such work is time-consuming. For efficiently maintainable online content, we need automatic synthesis of ASL from a sparse script representing the lexical items and basic elements of the sentence.

Specifically, we are studying how to model and generate ASL animations that include syntactic facial expressions, conveying grammatical information during entire phrases and therefore constrained by the timing of the manual signs in a phrase (Baker-Shenk, 1983). Generally speaking, in ASL, upper face movements (examined in this paper) convey syntactic information across entire phrases, with the mouth movements conveying lexical or adverbial information.

The meaning of a sequence of signs performed with the hands depends on the co-occuring facial expression. (While we use the term "facial expressions," these phenomena also include movements of the head.) For instance, the ASL sentence "BOB LIKE CHOCOLATE" (English: "Bob likes chocolate.") becomes a yes/no question (English: "Does Bob like chocolate?"), with the addition of a YesNo facial expression during the sentence. The addition of a Negative facial expression during the verb phrase "LIKE CHOCOLATE" changes the meaning of the sentence to "Bob doesn't like chocolate." (The lexical item NOT may optionally be used.) For interrogative questions, a WhQuestion facial expression must occur during the sentence, e.g., "BOB LIKE

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2084–2093,
Berlin, Germany, August 7-12, 2016.

WHAT." The five types of ASL facial expressions investigated in this paper include:

- YesNo: The signer raises his eyebrows while tilting the head forward to indicate that the sentence is a polar question.
- WhQuestion: The signer furrows his eyebrows and tilts his head forward during a sentence to indicate an interrogative question, typically with a "WH" word such as what, who, where, when, how, which, etc.
- Rhetorical: The signer raises his eyebrows and tilts his head backward and to the side to indicate a rhetorical question.
- Topic: The signer raises his eyebrows and tilts his head backward during a clause-initial phrase that should be interpreted as a topic.
- Negative: The signer shakes his head left and right during the verb phrase to indicate negated meaning, often with the sign NOT.

1.2 Prior Work

A survey of recent work of several researchers on producing animations of sign language with facial expressions appears in (Kacorri, 2015). There is recent interest in data-driven approaches using facial motion-capture of human performances to generate sign language animations: For example, (Schmidt et al., 2013) used clustering techniques to select facial expressions that co-occur with individual lexical items, and (Gibet et al., 2011) studied how to map facial motion-capture data to animation controls.

In the most closely related prior work, we had investigated how to generate a face animation based on a set of video recordings of a human signer performing facial expressions (Kacorri et al., 2016), with head and face movement data automatically extracted from the video, and with individual recordings labeled as each of the five syntactic types, as listed in section 1.1. We wanted to identify a single exemplar recording in our dataset, for each of the syntactic types, that could be used as the basis for generating the movements of virtual human character. (In a collection of recordings of face and head movement, there will naturally be non-essential individual variation in the movements; thus, it may be desirable to select a recording that is maximally stereotypical of a set of recordings.) To do so, we made use of a variant of Dynamic Time Warping (DTW) as a distance metric to select the recording with minimal pairwise normalized DTW distance from all of the examples of each syntactic type. We had used this "centroid" recording as the basis for producing a novel animation of the face and head movements for a sign language sentence.

2 Method

In this paper, we present a new methodology for generating face and head movements for sign language animations, given a set of human recordings of various syntactic types of facial expressions. Whereas we had previously selected a single exemplar recording of a human performance to serve as a basis for producing an animation (Kacorri et al., 2016), in this work, we investigate how to construct a model that generalizes across the entire set of recordings, to produce an "average" of the face and head movements, which can serve as a basis for generating an animation. To enable comparison of our new methodology to our prior technique, we make use of an identical training dataset as in (Kacorri et al., 2016) and an identical animation rendering pipeline, described in (Huenerfauth and Kacorri, 2015a). Briefly, the animation pipeline accepts a script of the hand location, hand orientation, and hand-shape information to pose and move the arms of the character over time, and it also accepts a file containing a stream of face movement information in MPEG4 Facial Animation Parameters format (ISO/IEC, 1999) to produce a virtual human animation.

2.1 Dataset and Feature Extraction

ASL is a low-resource language, and it does not have a writing system in common use. Therefore, ASL corpora are generally small in size and in limited supply; they are usually produced through manual annotation of video recordings. Thus, researchers generally work with relatively small datasets. In this work, we make use of two datasets that consist of video recordings of humans performing ASL with annotation labeling the times in the video when each of the five types of syntactic facial expressions listed in section 1.1 occur.

The training dataset used in this study was described in (Kacorri et al., 2016), and consists of 199 examples of facial expressions performed by a female signer recorded at Boston University. While the Training dataset can naturally be partitioned into five subsets, based on each of the five syntactic facial expression types, because adjacent

Type	Subgroup "_A" (Num. of Videos)	Subgroup "_B" (Num. of Videos)
YesNo	Immediately preceded by a facial expression with raised eyebrows, e.g. Topic. (9)	Not immediately preceded by an eyebrow-raising expression. (10)
WhQuestion	Performed during a single word, namely the wh-word (e.g., what, where, when). (4)	Performed during a phrase consisting of multiple words. (8)
Rhetorical	Performed during a single word, namely the wh-word (e.g., what, where, when). (2)	Performed during a phrase consisting of multiple words. (8)
Topic	Performed during a single word. (29)	Performed during a phrase consisting of multiple words. (15)
Negative	Immediately preceded by a facial expression with raised eyebrows, e.g. Topic. (16)	Not immediately preceded by eyebrow-raising expression. (25)

Table 1: Ten subgroups of the training dataset.

facial expressions or phrase durations may affect the performance of ASL facial expressions, in this work, we sub-divide the dataset further, into ten sub-groups, as summarized in Table 1.

The "gold-standard" dataset used in this study was shared with the research community by (Huenerfauth and Kacorri, 2014); we use 10 examples of ASL facial expressions (one for each sub-group listed in Table 1) performed by a male signer who was recorded at the Linguistic and Assistive Technologies laboratory.

To extract face and head movement information from the video, a face-tracker (Visage, 2016) was used to produce a set of MPEG4 facial animation parameters for each frame of video: These values represent face-landmark or head movements of the human appearing in the video, including 14 features used in this study: head_x, head_y, head_z, head_pitch, head_yaw, head_roll, raise_l_i_brow, raise_r_i_brow, raise_l_m_brow, raise_r_m_brow, raise_l_o_brow, raise_r_o_brow, squeeze_l_brow, squeeze_r_brow. The first six values represent head location and orientation. The next six values represent vertical movement of the outer ("o_"), middle ("m_"), or inner ("i_") portion of the right ("r_") or left ("l_") eyebrows. The final values represent horizontal movement of the eyebrows.

2.2 Continuous Profile Models (CPM)

Continuous Profile Model (CPM) aligns a set of related time series data while accounting for changes in amplitude. This model has been previously evaluated on speech signals and on other biological time-series data (Listgarten et al., 2004). With the assumption that a noisy, stochastic process generates the observed time series data, the approach automatically infers the underlying noiseless representation of the data, the so-called "latent trace." Figure 6 (on the last page of this paper) shows an example of multiple time series in unaligned and aligned space, with CPM identifying the the latent trace.

Given a set K of observed time series $\vec{x}^k = (x_1^k, x_2^k, ..., x_N^k)$, CPM assumes there is a latent trace $\vec{z} = (z_1, z_2, ..., z_M)$. While not a requirement of the model, the length of the time series data is assumed to be the same (N) and the length of the latent trace used in practice is $M = (2+\varepsilon)N$, where an ideal M would be large relative to N to allow precise mapping between observed data and an underlying point on the latent trace. Higher temporal resolution of the latent trace also accommodates flexible alignments by allowing an observational series to advance along the latent trace in small or large jumps (Listgarten, 2007).

Continuous Profile Models (CPMs) build on Hidden Markov Models (HMMs) (Poritz, 1988) and share similarities with Profile HMMs which augment HMMs by two constrained-transition states: 'Insert' and 'Delete' (emitting no observations). Similar to the Profile HMM, the CPM has strict left-to-right transition rules, constrained to only move forward along a sequence. Figure 1 includes a visualization we created, which illustrates the graphical model of a CPM.

2.3 Obtaining the CPM Latent Trace

We applied the CPM model to time align and coherently integrate time series data from multiple ASL facial expression performances of a particular type, e.g., Topic_A as listed in section 2.1, with the goal of using the inferred 'latent traces' to drive ASL animations with facial expressions of that type. This section describes our work to train the CPM and to obtain the latent traces; implementation details appear in Appendix A.

The input time-series data for each CPM model is the face and head movement data extracted from ASL videos of one of the facial expression types,

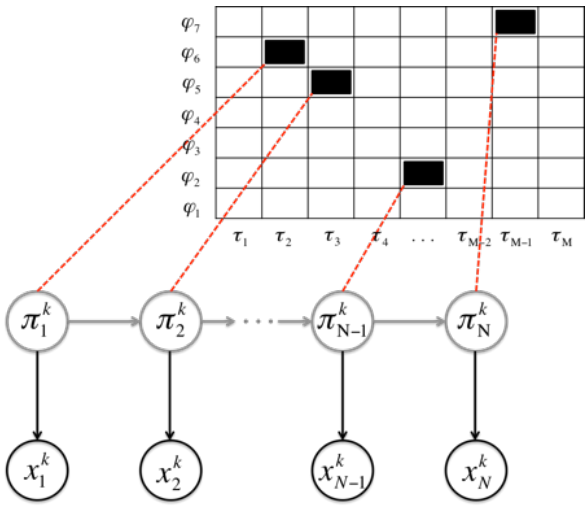


Figure 1: Depiction of a CPM for series x^k, with hidden state variables π_i^k underlying each observation x_i^k. The table illustrates the state-space: time-state/scale-state pairs mapped to the hidden variables, where time states belong to the integer set $(1...M)$ and scale states belong to an ordered set, here with 7 evenly spaced scales in logarithmic space as in (Listgarten et al., 2004).

as shown in Table 2. For each dataset, all the training examples are stretched (resampled using cubic interpolation) to meet the length of the longest example in the set. The length of time series, N, corresponds to the duration in video frames of the longest example in the data set. The recordings in the training set have 14 dimensions, corresponding to the 14 facial features listed in Section 2.1. As discussed above, the latent trace has a time axis of length M, which is approximately double the temporal resolution of the original training examples.

CPM Models	**Training Data** $\#Examples \times N \times \#Features$	**Latent Trace** $M \times \#Features$ where $M = (2+\varepsilon)N$
YesNo_A	9 x 51 x 14	105 x 14
YesNo_B	10 x 78 x 14	160 x 14
WhQuestion_A	4 x 24 x 14	50 x 14
WhQuestion_B	8 x 41 x 14	84 x 14
Rhetorical_A	2 x 16 x 14	33 x 14
Rhetorical_B	8 x 55 x 14	113 x 14
Topic_A	29 x 29 x 14	60 x 14
Topic_B	15 x 45 x 14	93 x 14
Negative_A	16 x 67 x 14	138 x 14
Negative_B	25 x 76 x 14	156 x 14

Table 2: Training data and the obtained latent traces for each of the CPM models on ASL facial expression subcategories.

To demonstrate our experiments, Figure 6 illustrates one of the subcategories, Rhetorical_B. (This figure appears at the end of the paper, due to its large size.) We illustrate the training set, before and after the alignment and amplitude normalization with the CPM, and the obtained latent trace for this subcategory. Figure 6a and Figure 6b illustrate each of the 8 training examples with a subplot extending from $[0, N]$ in the x-axis, which is the observed time axis in video frames. Each of the 14 plots represents one of the head or face features. Figure 6c illustrates the learned latent trace with a subplot extending from $[0, M]$ in the x-axis, which is the latent time axis. While the training set for this subcategory is very small and has high variability, upon visual inspection of Figure 6, we can observe that the learned latent trace shares similarities with most of the time series in the training set without being identical to any of them.

We expect that during the Rhetorical facial expression (Section 2.1), the signer's eyebrows will rise and the head will be tilted back and to the side. In the latent trace, the inner, middle, and outer portions of the left eyebrow rise (Figure 6c, plots 7, 9, 11), and so do the inner, middle, and outer portions of the right eyebrow (Figure 6c, plots 8, 10, 12). Note how the height of the lines in those plots rise, which indicates increased eyebrow height. For the Rhetorical facial expression, we would also expect symmetry in the horizontal displacement of the eyebrows, and we see such mirroring in the latent-trace: In (Figure 6c, plots 13-14), note the tendency for the line in plot 13 (left eyebrow) to increase in height as the line in plot 14 (right eyebrow) decreases in height, and vice versa.

3 Evaluation

This section presents two forms of evaluation of the CPM latent trace model for ASL facial expression synthesis. In Section 3.1, the CPM model will be compared to a "gold-standard" performance of each sub-category of ASL facial expression using a distance-metric-based evaluation, and in Section 3.2, the results of a user-study will be presented, in which ASL signers evaluated animations of ASL based upon the CPM model.

To provide a basis of comparison, in this section, we evaluate the CPM approach in comparison to an alternative approach that we call 'Centroid', which we described in prior work in (Ka-

corri et al., 2016), where we used a multivariate DTW to select one of the time series in the training set as a representative performance of the facial expression. The centroid examples are actual recordings of human ASL signers that are used to drive an animation. Appendix A lists the codenames of the videos from the training dataset selected as centroids and the codenames of the videos used in the gold-standard dataset (Huenerfauth and Kacorri, 2014).

3.1 Metric Evaluation

The gold-standard recordings of a male ASL signer were described in Section 2.1. In addition to the video recordings (which were processed to extract face and head movement data), we have annotation of the timing of the facial expressions and the sequence of signs performed on the hands. To compare the quality of our CPM model and that of the Centroid approach, we used each method to produce a candidate sequence of face and head movements for the sentence performed by the human in the gold-standard recording. Thus, the extracted facial expressions from the human recording can serve as a gold standard for how the face and head should move. In this section, we compare: (a) the distance of the CPM latent trace from the gold standard to (b) the distance of the centroid form the gold standard. It is notable that these gold-standard recordings were previously "unseen" during the creation of the CPM or Centroid models, that is, they were not used in the training data set during the creation of either model.

Since there was variability in the length of the latent trace, centroid, and gold-standard videos, for a fairer comparison, we first resampled these time series, using cubic interpolation, to match the duration (in milliseconds) of the gold-standard ASL sentence, and then we used multivariate DTW to estimate their distance, following the methodology of (Kacorri et al., 2016) and (Kacorri and Huenerfauth, 2015). In prior work (Kacorri and Huenerfauth, 2015), we had shown that a scoring algorithm based on DTW had moderate (yet significant) correlation with scores that participants assigned to ASL animation with facial expressions.

Figure 2 shows an example of a DTW distance scoring between the gold standard and each of the latent trace and the centroid, for one face feature

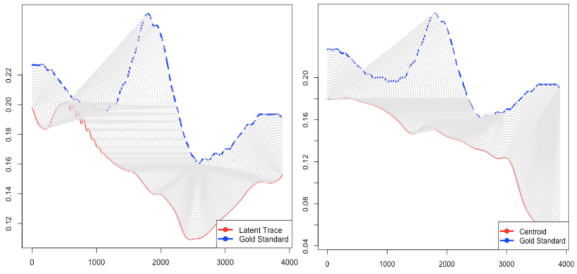


Figure 2: DTW distances on the squeeze_l_brow feature (left eyebrow horizontal movement), during a Negative_A facial expression: (left) between the CPM latent trace and gold standard and (right) between the centroid and gold standard. The timeline is given in milliseconds.

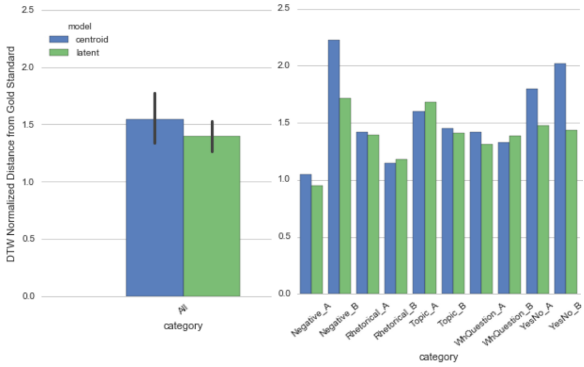


Figure 3: Overall normalized DTW distances for latent trace and centroid (left) and per each subcategory of ASL facial expression (right).

(horizontal movement of the left eyebrow) during a Negative_A facial expression. Given that the centroid and the training data for the latent trace are driven by recordings of a (female) signer and the gold standard is a different (male) signer, there are differences between these facial expressions due to idiosyncratic aspects of individual signers. Thus the metric evaluation in this section is challenging because it is an inter-signer evaluation.

Figure 3 illustrates the overall calculated DTW distances, including a graph with the results broken down per subcategory of ASL facial expression. The results indicate that the CPM latent trace is closer to the gold standard than the centroid is. Note that the distance values are not zero since the latent trace and the centroid are being compared to a recording from a different signer on novel, previously unseen, ASL sentences. The results in these graphs suggest that the latent trace model out-performed the centroid approach.

Figure 4: Screenshots of YesNo_A stimuli of three types: a) neutral, b) centroid, and c) latent trace.

3.2 User Evaluation

To further assess our ASL synthesis approach, we conducted a user study where ASL signers watched short animations of ASL sentences with identical hand movements but differing in their face, head, and torso movements. There were three conditions in this between-subjects study: a) animations with a static neutral face throughout the animation (as a lower baseline), b) animations with facial expressions driven by the centroid human recording, and c) animations with facial expressions driven by the CPM latent trace based on multiple recordings of a human performing that type of facial expression. Figure 4 illustrates screenshots of each stimulus type for a YesNo_A facial expression. The specific sentences used for this study were drawn from a standard test set of stimuli released to the research community by (Huenerfauth and Kacorri, 2014) for evaluating animations of sign language with facial expressions.

All three types of stimuli (neutral, centroid and latent trace), shared identical animation-control scripts specifying the hand and arm movements; these scripts were hand-crafted by ASL signers in a pose-by-pose manner. For the neutral animations, we did not specify any torso, head, nor face movements; rather, we left them in their neutral pose throughout the sentences. As for the centroid and latent trace animations, we applied the head and face movements (as specified by the centroid model or by the latent trace model) only to the portion of the animation where the facial expression of interest occurs, leaving the head and face for the rest of the animation to a neutral pose. For instance, during a stimulus that contains a Wh-question, the face and head are animated only during the Wh-question, but they are left in a neutral pose for the rest of the stimulus (which may include other sentences). The period of time when the facial expression occurred was time-aligned with the subset of words (the sequence of signs performed on the hands) for the appropriate syntactic domain; the phrase-beginning and phrase-ending was aligned with the performance of the facial expression. Thus, the difference in appearance between our animation stimuli was subtle: The only portion of the animations that differed between the three conditions (neutral, centroid, and latent-trace) was the face and the head movements during the span of time when the syntactic facial expression should occur (e.g., during the Wh-question).

We resampled the centroid and CPM time series, using cubic interpolation, to match the duration (in milliseconds) of the animation they would be applied to. To convert the centroid and latent trace time series into the input for the animation-generation system, we used the MPEG4-features-to-animation pipeline described in (Kacorri et al., 2016). That platform is based upon the open-source EMBR animation system for producing human animation (Heloir and Kipp, 2009); specifically, the facial expressions were represented as an EMBR PoseSequence with a pose defined every 133 milliseconds.

In prior work (Huenerfauth and Kacorri, 2015b), we investigated key methodological considerations in conducting a study to evaluate sign language animations with deaf users, including the use of appropriate baselines for comparison, appropriate presentation of questions and instructions, demographic and technology experience factors influencing acceptance of signing avatars, and other factors that we have considered in the design of this current study. Our recent work (Kacorri et al., 2015) has established a set of demographic and technology experience questions which can be used to screen for the most critical participants in a user study of ASL signers to evaluate animation. Specifically, we screened for participants that identified themselves as "deaf/Deaf" or "hard-of-hearing," who had grown up using ASL at home or had attended an ASL-based school as a young child, such as a residential or daytime school.

Deaf researchers (all fluent ASL signers) recruited and collected data from participants, during meetings conducted in ASL. Initial advertise-

ments were sent to local email distribution lists and Facebook groups. A total of 17 participants met the above criteria, where 14 participants self-identified as deaf/Deaf and 3 as hard-of-hearing. Of our participants in the study, 10 had attended a residential school for deaf students, and 7, a daytime school for deaf students. 14 participants had learned ASL prior to age 5, and the remaining 3 had been using ASL for over 7 years. There were 8 men and 9 women of ages 19-29 (average age 22.8). In prior work, we (Kacorri et al., 2015) have advocated that participants in studies evaluating sign language animation complete a two standardized surveys about their technology experience (MediaSharing and AnimationAttitude) and that researchers report these values for participants, to enable comparison across studies. In our study, participant scores for MediaSharing varied between 3 and 6, with a mean score of 4.3, and scores for AnimationAttitude varied from 2 to 6, with a mean score of 3.8.

At the beginning of the study, participants viewed a sample animation, to familiarize them with the experiment and the questions they would be asked about each animation. (This sample used a different stimulus than the other ten animations shown in the study.) Next, they responded to a set of questions that measured their subjective impression of each animation, using a 1-to-10 scalar response. Each question was conveyed using ASL through an onscreen video, and the following English question text was shown on the questionnaire: (a) Good ASL grammar? (10=Perfect, 1=Bad); (b) Easy to understand? (10=Clear, 1=Confusing); (c) Natural? (10=Moves like person, 1=Like robot). These questions have been used in many prior experimental studies to evaluate animations of ASL, e.g. (Kacorri and Huenerfauth, 2015), and were shared with research community as a standard evaluation tool in (Huenerfauth and Kacorri, 2014). To calculate a single score for each animation, the scalar response scores for the three questions were averaged.

Figure 5 shows distributions of subjective scores as boxplots with a 1.5 interquartile range (IQR). For comparison, means are denoted with a star and their values are labeled above each boxplot. When comparing the subjective scores that participants assigned to the animations in Figure 5, we found a significant difference (Kruskal-Wallis test used since the data was not normally

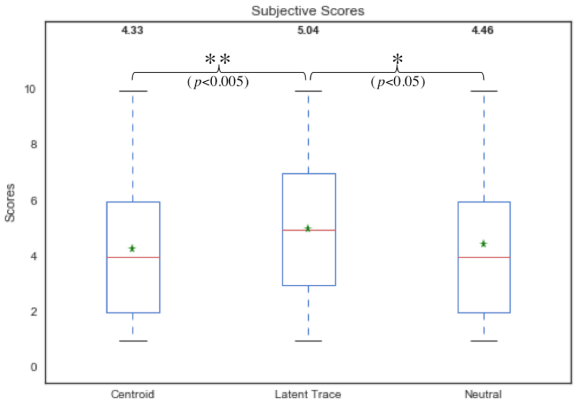


Figure 5: Subjective scores for centroid, latent trace, and neutral animations.

distributed) between the latent trace and centroid ($p < 0.005$) and between the latent trace and neutral ($p < 0.05$).

In summary, our CPM modeling approach for generating an animation out-performed an animation produced from an actual recording of a single human performance (the "centroid" approach). In prior methodological studies, we demonstrated that it is valid to use either videos of humans or animations (driven by a human performance) as the baseline for comparison in a study of ASL animation (Kacorri et al., 2013a). As suggested by Figure 4, the differences in face and head movements between the Centroid and CPM conditions were subtle, yet fluent ASL signers rated the CPM animations higher in this study.

4 Conclusion and Future Work

To facilitate the creation of ASL content that can easily be updated or maintained, we have investigated technologies for automating the synthesis of ASL animations from a sparse representation of the message. Specifically, this paper has focused on the synthesis of syntactic ASL facial expressions, which are essential to sentence meaning, using a data-driven methodology in which recordings of human ASL signers are used as a basis for generating face and head movements for animation. To avoid idiosyncratic aspects of a single performance, we have modeled a facial expression based on the underlying trace of the movement trained on multiple recordings of different sentences where this type of facial expression occurs. We obtain the latent trace with Continuous Profile Model (CPM), a probabilistic generative model that relies on Hidden Markov Models. We

assessed our modeling approach through comparison to an alternative centroid approach, where a single performance was selected as a representative. Through both a metric evaluation and an experimental user study, we found that the facial expressions driven by our CPM models produce high-quality facial expressions that are more similar to human performance of novel sentences.

While this work used the latent trace as the basis for animation, in future work, we also plan to explore methods for sampling from the model to produce variations in face and head movement. In addition, to aid CPM convergence to a good local optimum, in future work we will investigate dimensionality reduction approaches that are reversible such as Principal Component Analysis (Pearson, 1901) and other pre-processing approaches similar to (Listgarten, 2007), where the training data set is coarsely pre-aligned and pre-scaled based on the center of mass of the time series. In addition we plan to further investigate how to fine-tune some of the hyper parameters of the CPM such as spline scaling, single global scaling factor, convergence tolerance, and initialization of the latent trace with a centroid. In subsequent work, we would like to explore alternatives for enhancing CPMs by incorporating contextual features in the training data set such as timing of hand movements, and preceding, succeeding, and co-occurring facial expressions.

Acknowledgments

This material is based upon work supported by the National Science Foundation under award number 1065009 and 1506786. This material is also based upon work supported by the Science Fellowship and Dissertation Fellowship programs of The Graduate Center, CUNY. We are grateful for support and resources provided by Ali Raza Syed at The Graduate Center, CUNY, and by Carol Neidle at Boston University.

References

Charlotte Baker-Shenk. 1983. A microanalysis of the nonmanual components of questions in american sign language.

Sarah Ebling and John Glauert. 2013. Exploiting the full potential of jasigning to build an avatar signing train announcements. In *Proceedings of the Third International Symposium on Sign Language Translation and Avatar Technology (SLTAT), Chicago, USA, October*, volume 18, page 19.

Michael Filhol, Mohamed N Hadjadj, and Benoît Testu. 2013. A rule triggering system for automatic text-to-sign translation. *Universal Access in the Information Society*, pages 1–12.

Sylvie Gibet, Nicolas Courty, Kyle Duarte, and Thibaut Le Naour. 2011. The signcom system for data-driven animation of interactive virtual signers: Methodology and evaluation. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 1(1):6.

Alexis Heloir and Michael Kipp. 2009. Embr–a realtime animation engine for interactive embodied agents. In *Intelligent Virtual Agents*, pages 393–404. Springer.

Matt Huenerfauth and Hernisa Kacorri. 2014. Release of experimental stimuli and questions for evaluating facial expressions in animations of american sign language. In *Proceedings of the 6th Workshop on the Representation and Processing of Sign Languages: Beyond the Manual Channel, The 9th International Conference on Language Resources and Evaluation (LREC 2014), Reykjavik, Iceland*.

Matt Huenerfauth and Hernisa Kacorri. 2015a. Augmenting embr virtual human animation system with mpeg-4 controls for producing asl facial expressions. In *International Symposium on Sign Language Translation and Avatar Technology*, volume 3.

Matt Huenerfauth and Hernisa Kacorri. 2015b. Best practices for conducting evaluations of sign language animation. *Journal on Technology and Persons with Disabilities*, 3.

ISO/IEC. 1999. Information technology—Coding of audio-visual objects—Part 2: Visual. ISO 14496-2:1999, International Organization for Standardization, Geneva, Switzerland.

Hernisa Kacorri and Matt Huenerfauth. 2015. Evaluating a dynamic time warping based scoring algorithm for facial expressions in asl animations. In *6th Workshop on Speech and Language Processing for Assistive Technologies (SLPAT)*, page 29.

Hernisa Kacorri, Pengfei Lu, and Matt Huenerfauth. 2013a. Effect of displaying human videos during an evaluation study of american sign language animation. *ACM Transactions on Accessible Computing (TACCESS)*, 5(2):4.

Hernisa Kacorri, Pengfei Lu, and Matt Huenerfauth. 2013b. Evaluating facial expressions in american sign language animations for accessible online information. In *Universal Access in Human-Computer Interaction. Design Methods, Tools, and Interaction Techniques for eInclusion*, pages 510–519. Springer.

Hernisa Kacorri, Allen Harper, and Matt Huenerfauth. 2014. Measuring the perception of facial expressions in american sign language animations with eye tracking. In *Universal Access in Human-Computer Interaction. Design for All and Accessibility Practice*, pages 553–563. Springer.

Hernisa Kacorri, Matt Huenerfauth, Sarah Ebling, Kasmira Patel, and Mackenzie Willard. 2015. Demographic and experiential factors influencing acceptance of sign language animation by deaf users. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers & Accessibility*, pages 147–154. ACM.

Hernisa Kacorri, Ali Raza Syed, Matt Huenerfauth, and Carol Neidle. 2016. Centroid-based exemplar selection of asl non-manual expressions using multidimensional dynamic time warping and mpeg4 features. In *Proceedings of the 7th Workshop on the Representation and Processing of Sign Languages: Corpus Mining, The 10th International Conference on Language Resources and Evaluation (LREC 2016), Portoroz, Slovenia*. http://huenerfauth.ist.rit.edu/pubs/lrec2016.pdf.

Hernisa Kacorri. 2015. Tr-2015001: A survey and critique of facial expression synthesis in sign language animation. Technical report, The Graduate Center, CUNY. http://academicworks.cuny.edu/gc_cs_tr/403.

Jennifer Listgarten, Radford M Neal, Sam T Roweis, and Andrew Emili. 2004. Multiple alignment of continuous time series. In *Advances in neural information processing systems*, pages 817–824.

Jennifer Listgarten. 2007. *Analysis of sibling time series data: alignment and difference detection*. Ph.D. thesis, University of Toronto.

Carol Neidle, Jingjing Liu, Bo Liu, Xi Peng, Christian Vogler, and Dimitris Metaxas. 2014. Computer-based tracking, analysis, and visualization of linguistically significant nonmanual events in american sign language (asl). In *LREC Workshop on the Representation and Processing of Sign Languages: Beyond the Manual Channel*. Citeseer.

Karl Pearson. 1901. Principal components analysis. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 6(2):559.

Alan B Poritz. 1988. Hidden markov models: A guided tour. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 7–13. IEEE.

Christoph Schmidt, Oscar Koller, Hermann Ney, Thomas Hoyoux, and Justus Piater. 2013. Enhancing gloss-based corpora with facial features using active appearance models. In *International Symposium on Sign Language Translation and Avatar Technology*, volume 2.

Daniel Stein, Christoph Schmidt, and Hermann Ney. 2012. Analysis, preparation, and optimization of statistical sign language machine translation. *Machine Translation*, 26(4):325–357.

Carol Bloomquist Traxler. 2000. The stanford achievement test: National norming and performance standards for deaf and hard-of-hearing students. *Journal of deaf studies and deaf education*, 5(4):337–348.

Technologies Visage. 2016. Face tracking. https://visagetechnologies.com/products-and-services/visagesdk/facetrack. Accessed: 2016-03-10.

A Appendix: Supplemental Material

In Section 2.3, we made use of a freely available CPM implementation available from http://www.cs.toronto.edu/~jenn/CPM/ in MATLAB, Version 8.5.0.197613 (R2015a).

One parameter for regularizing the latent trace (Listgarten, 2007) is a smoothing parameter (λ), with values being dataset-dependent. To select a good λ, we experimented with held-out data and found that $\lambda = 4$ and $NumberOfIterations = 3$ resulted in a latent trace curve that captures the shape of the ASL features well. Other CPM parameters were:

- $USE\_SPLINE = 0$: if set to 1, uses spline scaling rather than HMM scale states
- $oneScaleOnly = 0$: no HMM scale states (only a single global scaling factor is applied to each time series.)
- $extraPercent(\varepsilon)$ = 0.05: slack on the length of the latent trace M, where $M = (2 + \varepsilon)N$.
- $learnStateTransitions = 0$: whether to learn the HMM state-transition probabilities
- $learnGlobalScaleFactor = 1$: learn single global scale factor for each time series

Section 3.1 described how the centroids were selected from among videos in the Boston University dataset (Neidle et al., 2014), and the gold standard videos were selected from among videos in a different dataset (Huenerfauth and Kacorri, 2014). Table 3 lists the code names of the selected videos, using the nomenclature of each dataset.

Subcategory	Centroid Codename	Gold-Standard Codename
YesNo_A	2011-12-01_0037-cam2-05	Y4
YesNo_B	2011-12-01_0037-cam2-09	Y3
WhQuestion_A	2011-12-01_0038-cam2-05	W1
WhQuestion_B	2011-12-01_0038-cam2-07	W2
Rhetorical_A	2011-12-01_0041-cam2-04	R3
Rhetorical_B	2011-12-01_0041-cam2-02	R9
Topic_A	2012-01-27_0050-cam2-05	T4
Topic_B	2012-01-27_0051-cam2-09	T3
Negative_A	2012-01-27_0051-cam2-03	N2
Negative_B	2012-01-27_0051-cam2-30	N5

Table 3: Codenames of videos selected as centoids and gold standards for comparison in section 3.1.

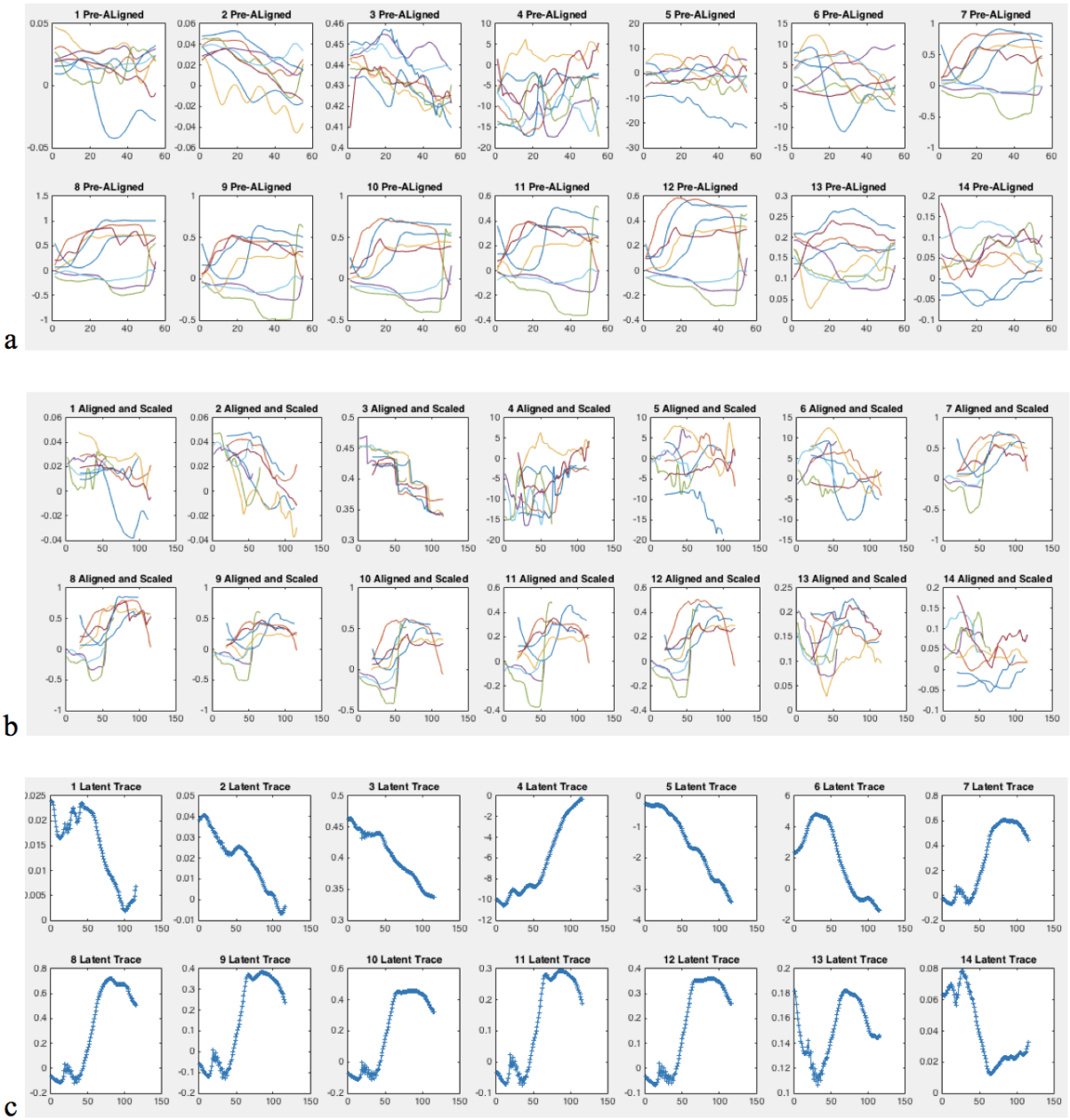


Figure 6: Example of CPM modeling for Rhetorical_B: (a) training examples before CPM (each plot shows one of the 14 face features over time, with 8 colored lines in each plot showing each of the 8 training examples), (b) after CPM time-alignment and rescaling, and (c) the final latent trace based upon all 8 examples.

Evaluating Sentiment Analysis in the Context of Securities Trading

Siavash Kazemian and **Shunan Zhao** and **Gerald Penn**
Department of Computer Science
University of Toronto
{kazemian,szhao,gpenn}@cs.toronto.edu

Abstract

There are numerous studies suggesting that published news stories have an important effect on the direction of the stock market, its volatility, the volume of trades, and the value of individual stocks mentioned in the news. There is even some published research suggesting that automated sentiment analysis of news documents, quarterly reports, blogs and/or twitter data can be productively used as part of a trading strategy. This paper presents just such a family of trading strategies, and then uses this application to re-examine some of the tacit assumptions behind how sentiment analyzers are generally evaluated, in spite of the contexts of their application. This discrepancy comes at a cost.

1 Introduction

The proliferation of opinion-rich text on the World Wide Web, which includes anything from product reviews to political blog posts, led to the growth of sentiment analysis as a research field more than a decade ago. The market need to quantify opinions expressed in social media and the blogosphere has provided a great opportunity for sentiment analysis technology to make an impact in many sectors, including the financial industry, in which interest in automatically detecting news sentiment in order to inform trading strategies extends back at least 10 years. In this case, sentiment takes on a slightly different meaning; positive sentiment is not the emotional and subjective use of laudatory language. Rather, a news article that contains positive sentiment is optimistic about the future financial prospects of a company.

Zhang and Skiena (2010) experimented with news sentiment to inform simple market neutral trading algorithms, and produced an impressive maximum yearly return of around 30% — even more when using sentiment from blogs and twitter data. They did so, however, without an appropriate baseline, making it very difficult to appreciate the significance of this number. Using a very standard, and in fact somewhat dated sentiment analyzer, we are regularly able to garner annualized returns over twice that percentage, and in a manner that highlights two of the better design decisions that Zhang and Skiena (2010) made, viz., (1) their decision to trade based upon numerical SVM scores rather than upon discrete positive or negative sentiment classes, and (2) their decision to go long (resp., short) in the n best- (worst-) ranking securities rather than to treat all positive (negative) securities equally.

On the other hand, we trade based upon the raw SVM score itself, rather than its relative rank within a basket of other securities as Zhang and Skiena (2010) did, and we experimentally tune a threshold for that score that determines whether to go long, neutral or short. We sampled our stocks for both training and evaluation in two runs, one without *survivor bias*, the tendency for long positions in stocks that are publicly traded as of the date of the experiment to pay better using historical trading data than long positions in random stocks sampled on the trading days themselves. Most of the evaluations of sentiment-based trading either unwittingly adopt this bias, or do not need to address it because their returns are computed over very brief historical periods. We also provide appropriate trading baselines as well as Sharpe ratios (Sharpe, 1966) to attempt to quantify the relative risk inherent to our experimental strategies. As tacitly assumed by most of the work on this subject, our trading strategy is not portfolio-limited, and our returns are calculated on a percentage basis with theoretical, commission-free trades.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2094–2103,
Berlin, Germany, August 7-12, 2016.

It is important to understand at the outset, however, that the purpose of this research was not to beat Zhang and Skiena's (2010) returns (although we have), nor merely to conduct the first properly controlled, sufficiently explicit, scientific test of the descriptive hypothesis that sentiment analysis is of benefit to securities trading (although, to our knowledge, we did). The main purpose of this study was in fact to reappraise the evaluation standards used by the sentiment analysis community. It is not at all uncommon within this community to evaluate a sentiment analyzer with a variety of classification accuracy or hypothesis testing scores such as F-measures, SARs, kappas or Krippendorf alphas derived from human-subject annotations — even when more extensional measures are available, such as actual market returns from historical data in the case of securities trading. With Hollywood films, another popular domain for automatic sentiment analysis, one might refer to box-office returns or the number of award nominations that a film receives rather than to its star-rankings on review websites where pile-on and confirmation biases are widely known to be rampant. Are the opinions of human judges, paid or unpaid, a sufficient proxy for the business cases that actually drive the demand for sentiment analyzers?

We regret to report that they do not seem to be. As a case study to demonstrate this point (Section 4.3), we exhibit one particular modification to our experimental financial sentiment analyzer that, when evaluated against an evaluation test set sampled from the same pool of human-subject annotations as the analyzer's training data, returns poorer performance, but when evaluated against actual market returns, yields better performance. This should worry any researcher who relies on classification accuracies, because the improvements that they report, whether due to better feature selection or different pattern recognition algorithms, may in fact not be improvements at all. Differences in the amount or degree of improvement might arguably be rescalable, but Section 4.3 shows that such intrinsic measures are not even accurate up to a determination of the delta's *sign*.

On the other hand, the results reported here should not be construed as an indictment of sentiment analysis as a technology or its potential application. In fact, one of our baselines alternatively attempts to train the same classifier directly on market returns, and the experimental approach handily beats that, too. It is important to train on human-annotated sentiments, but then it is equally important to tune, and eventually evaluate, on an empirically grounded task-specific measure, such as market returns. This paper thus presents, to our knowledge, the first real proof that sentiment is worth analyzing in this or any other domain.

A likely machine-learning explanation for this experimental result is that whenever two unbiased estimators are pitted against each other, they often result in an improved combined performance because each acts as a regularizer against the other. If true, this merely attests to the relative independence of task-based and human-annotated knowledge sources. A more HCI-oriented view, however, would argue that direct human-subject annotations are highly problematic unless the annotations have been elicited in manner that is *ecologically valid*. When human subjects are paid to annotate quarterly reports or business news, they are paid regardless of the quality of their annotations, the quality of their training, or even their degree of comprehension of what they are supposed to be doing. When human subjects post film reviews on web-sites, they are participating in a cultural activity in which the quality of the film under consideration is only one factor. These sources of annotation have not been properly controlled in previous experiments on sentiment analysis.

Regardless of the explanation, this is a lesson that applies to many more areas of NLP than just sentiment analysis, and to far more recent instances of sentiment analysis than the one that we based our experiments on here. Indeed, we chose sentiment analysis because this is an area that *can* set a higher standard; it has the right size for an NLP component to be embedded in real applications and to be evaluated properly. This is noteworthy because it is challenging to explain why recent publications in sentiment analysis research would so dramatically increase the value that they assign to sentence-level sentiment scoring algorithms based on syntactically compositional derivations of "good-for/ bad-for" annotation (Anand and Reschke, 2010; Deng et al., 2013), when statistical parsing itself has spent the last twenty-five years staggering through a linguistically induced delirium as it attempts to document any of its putative advances without recourse to clear empirical evidence that PTB-style syntactic derivations are a reliable approximation of seman-

tic content or structure.

We submit, in light of our experience with the present study, that the most crucial obstacle facing the state of the art in sentiment analysis is not a granularity problem, nor a pattern recognition problem, but an evaluation problem. Those evaluations must be task-specific to be reliable, and sentiment analysis, in spite of our careless use of the term in the NLP community, is not a task. Stock trading is a task — one of many in which a sentiment analyzer is a potentially useful component. This paper provides an example of how to test that utility.

2 Related Work in Financial Sentiment Analysis

Studies confirming the relationship between media and market performance date back to at least Niederhoffer (1971), who looked at NY Times headlines and determined that large market changes were more likely following world events than on random days. Conversely, Tetlock (2007) looked at media pessimism and concluded that high media pessimism predicts downward prices. Tetlock (2007) also developed a trading strategy, achieving modest annualized returns of 7.3%. Engle and Ng (1993) looked at the effects of news on volatility, showing that bad news introduces more volatility than good news. Chan (2003) claimed that prices are slow to reflect bad news and stocks with news exhibit momentum. Antweiler and Frank (2004) showed that there is a significant, but negative correlation between the number of messages on financial discussion boards about a stock and its returns, but that this trend is economically insignificant. Aside from Tetlock (2007), none of this work evaluated the effectiveness of an actual sentiment-based trading strategy.

There is, of course, a great deal of work on automated sentiment analysis itself; see Pang and Lee (2008) for a survey. More recent developments germane to our work include the use of information retrieval weighting schemes (Paltoglou and Thelwall, 2010), with which accuracies of up to 96.6% have models based upon Latent Dirichlet Allocation (LDA) (Lin and He, 2009).

There has also been some work that analyzes the sentiment of financial documents without actually using those results in trading strategies (Koppel and Shtrimberg, 2004; Ahmad et al., 2006; Fu et al., 2008; O'Hare et al., 2009; Devitt and Ahmad, 2007; Drury and Almeida, 2011). As to the relationship between sentiment and stock price, Das and Chen (2007) performed sentiment analysis on discussion board posts. Using this, they built a "sentiment index" that computed the time-varying sentiment of the 24 stocks in the Morgan Stanley High-Tech Index (MSH), and tracked how well their index followed the aggregate price of the MSH itself. Their sentiment analyzer was based upon a voting algorithm, although they also discussed a vector distance algorithm that performed better. Their baseline, the Rainbow algorithm, also came within 1 percentage point of their reported accuracy. This is one of the very few studies that has evaluated sentiment analysis itself (as opposed to a sentiment-based trading strategy) against market returns (versus gold-standard sentiment annotations). Das and Chen (2007) focused exclusively on discussion board messages and their evaluation was limited to the stocks on the MSH, whereas we focus on Reuters newswire and evaluate over a wide range of NYSE-listed stocks and market capitalization levels.

Butler and Keselj (2009) try to determine sentiment from corporate annual reports using both character n-gram profiles and readability scores. They also developed a sentiment-based trading strategy with high returns, but do not report how the strategy works or how they computed the returns, making the results difficult to compare to ours. Basing a trading strategy upon annual reports also calls into question the frequency with which the trading strategy could be exercised.

The work most similar to ours is Zhang and Skiena's (2010). They look at both financial blog posts and financial news, forming a market-neutral trading strategy whereby each day, companies are ranked by their reported sentiment. The strategy then goes long and short on equal numbers of positive- and negative-sentiment stocks, respectively. They conduct their trading evaluation over the period from 2005 to 2009, and report a yearly return of roughly 30% when using news data, and yearly returns of up to 80% when they use Twitter and blog data. Crucially, they trade based upon the ranked relative order of documents by sentiment rather than upon the documents' raw sentiment scores.

Zhang and Skiena (2010) compare their strategy to two baselines. The "Worst-sentiment" Strategy trades the opposite of their strategy: short

on positive-sentiment stocks and long on negative sentiment stocks. The "Random-selection" Strategy randomly picks stocks to go long and short on. As trading strategies, these baselines set a very low standard. Our evaluation uses standard trading benchmarks such as momentum trading and holding the S&P, as well as oracle trading strategies over the same holding periods.

3 Method and Materials

3.1 News Data

Our dataset combines two collections of *Reuters* news documents. The first was obtained for a roughly evenly weighted collection of 22 small-, mid- and large-cap companies, randomly sampled from the list of all companies traded on the NYSE as of 10^{th} March, 1997. The second was obtained for a collection of 20 companies randomly sampled from those companies that were publicly traded in March, 1997 and still listed on 10^{th} March, 2013. For both collections of companies, we collected every chronologically third Reuters news document about them from the period March, 1997 to March, 2013. The news articles prior to 10^{th} March, 2005 were used as training data, and the news articles on or after 10^{th} March, 2005 were reserved as testing data.[1] We split the dataset at a fixed date rather than randomly in order not to incorporate future news into the classifier through lexical choice.

In total, there were 1256 financial news documents. Each was labelled by two human annotators as being negative, positive, or neutral in sentiment. The annotators were instructed to gauge the author's belief about the company, rather than to make a personal assessment of the company's prospects. Only the 991 documents that were labelled twice as negative or positive were used for training and evaluation.

3.2 Sentiment Analysis Algorithm

For each selected document, we first filter out all punctuation characters and the most common 429 stop words. Because this is a document-level sentiment scoring task, not sentence-level, our sentiment analyzer is a support-vector machine with a linear kernel function implemented using SVM^{light} (Joachims, 1999), using all of its default parameters.[2] We have experimented with raw term frequencies, binary term-presence features, and term frequencies weighted by the BM25 scheme, which had the most resilience in the study of information-retrieval weighting schemes for sentiment analysis by Paltoglou and Thelwall (2010). We performed 10 fold cross-validation on the training data, constructing our folds so that each contains an approximately equal number of negative and positive examples. This ensures that we do not accidentally bias a fold.

Representation	Accuracy
bm25_freq	81.143%
term_presence	80.164%
bm25_freq_sw	79.827%
freq_with_sw	75.564%
freq	79.276%

Table 1: Average 10-fold cross validation accuracy of the sentiment classifier using different term-frequency weighting schemes. The same folds were used in all feature sets.

Pang et al. (2002) use word presence features with no stop list, instead excluding all words with frequencies of 3 or less. Pang et al. (2002) normalize their word presence feature vectors, rather than term weighting with an IR-based scheme like BM25, which also involves a normalization step. Pang et al. (2002) also use an SVM with a linear kernel on their features, but they train and compute sentiment values on film reviews rather than financial texts, and their human judges also classified the training films on a scale from 1 to 5, whereas ours used a scale that can be viewed as being from -1 to 1, with specific qualitative interpretations assigned to each number. Antweiler and Frank (2004) use SVMs with a polynomial kernel (of unstated degree) to train on word frequencies relative to a three-valued classification, but they only count frequencies for the 1000 words with the highest mutual information scores relative to the classification labels. Butler and Keselj (2009) also use an SVM trained upon a very different set of features, and with a polynomial kernel of degree

[1] An anonymous reviewer expressed concern about chronological bias in the training data relative to the test data because of this decision. While this may indeed influence our results, ecological validity requires us to situate all training data before some date, and all testing data after that date, because traders only have access to historical data before making a future trade.

[2] There has been one important piece of work (Tang et al., 2015) on neural computing architectures for document-level sentiment scoring (most neural computing architectures for sentiment scoring are sentence-level), but the performance of this type of architecture is not mature enough to replace SVMs just yet.

3.

As a sanity check, we measured our sentiment analyzer's accuracy on film reviews by training and evaluating on Pang and Lee's (2004) film review dataset, which contains 1000 positively and 1000 negatively labelled reviews. Pang and Lee conveniently labelled the folds that they used when they ran their experiments. Using these same folds, we obtain an average accuracy of 86.85%, which is comparable to Pang and Lee's 86.4% score for subjectivity extraction. The purpose of this comparison is simply to demonstrate that our implementation is a faithful rendering of Pang and Lee's (2004) algorithm.

Table 1 shows the performance of SVM with BM25 weighting on our Reuters evaluation set versus several baselines. All baselines are identical except for the term weighting schemes used, and whether stop words were removed. As can be observed, SVM-BM25 has the highest sentiment classification accuracy: 80.164% on average over the 10 folds. This compares favourably with previous reports of 70.3% average accuracy over 10 folds on financial news documents (Koppel and Shtrimberg, 2004). We will nevertheless adhere to normalized term presence for now, in order to stay close to Pang and Lee's (2004) implementation.

3.3 Trading Algorithm

Overall, our trading strategy is simple: go long when the classifier reports positive sentiment in a news article about a company, and short when the classifier reports negative sentiment.

We will embed the aforementioned sentiment analyzer into three different trading algorithms. In Section 4.1, we use the discrete polarity returned by the classifier to decide whether go long/abstain/short a stock. In Section 4.2.1 we instead use the distance of the current document from the classifier's decision boundary reported by the SVM. These distances do have meaningful interpretations apart from their internal use in assigning class labels. Platt (Platt, 1999) showed that they can be converted into posterior probabilities, for example, by fitting a sigmoid function onto them, but we will simply use the raw distances. In Section 4.2.2, we impose a safety zone onto the interpretation of these raw distance scores.

4 Experiments

In the experiments of this section, we will evaluate an entire trading strategy, which includes the sentiment analyzer and the particulars of the trading algorithm itself. The purpose of these experiments is to refine the trading strategy itself and so the sentiment analyzer will be held constant. In Section 4.3, we will hold the trading strategy constant, and instead vary the document representation features in the underlying sentiment analyzer.

In all three experiments, we compare the per-position returns of the following four standard strategies, where the number of days for which a position is held remains constant:

1. The momentum strategy computes the price of the stock h days ago, where h is the holding period. Then, it goes long for h days if the previous price is lower than the current price. It goes short otherwise.
2. The S&P strategy simply goes long on the *S&P 500* for the holding period. This strategy completely ignores the stock in question and the news about it.
3. The oracle S&P strategy computes the value of the *S&P 500* index h days into the future. If the future value is greater than the current day's value, then it goes long on the *S&P 500* index. Otherwise, it goes short.
4. The oracle strategy computes the value of the stock h days into the future. If the future value is greater than the current day's value, then it goes long on the stock. Otherwise, it goes short.

The oracle and oracle S&P strategies are included as toplines to determine how close the experimental strategies come to ones with perfect knowledge of the future. "Market-trained" is the same as "experimental" at test time, but trains the sentiment analyzer on the market return of the stock in question for h days following a training article's publication, rather than the article's annotation.

4.1 Experiment One: Utilizing Sentiment Labels

Given a news document for a publicly traded company, the trading agent first computes the sentiment class of the document. If the sentiment is positive, the agent goes long on the stock on the date the news is released; if negative, it goes short.

Strategy	Period	Return	S. Ratio
Experimental	30 days	-0.037%	-0.002
	5 days	0.763%	0.094
	3 days	0.742%	0.100
	1 day	0.716%	0.108
Momentum	30 days	1.176%	0.066
	5 days	0.366%	0.045
	3 days	0.713%	0.096
	1 day	0.017%	-0.002
S&P	30 days	0.318%	0.059
	5 days	-0.038%	-0.016
	3 days	-0.035%	-0.017
	1 day	0.046%	0.036
Oracle S&P	30 days	3.765%	0.959
	5 days	1.617%	0.974
	3 days	1.390%	0.949
	1 day	0.860%	0.909
Oracle	30 days	11.680%	0.874
	5 days	5.143%	0.809
	3 days	4.524%	0.761
	1 day	3.542%	0.630
Market-trained	30 days	0.286%	0.016
	5 days	0.447%	0.054
	3 days	0.358%	0.048
	1 day	0.533%	0.080

Table 2: Returns and Sharpe ratios for the Experimental, baseline and topline trading strategies over 30, 5, 3, and 1 day(s) holding periods.

All trades are made based on the adjusted closing price on this date. We evaluate the performance of this strategy using four different holding periods: 30, 5, 3, and 1 day(s).

The returns and Sharpe ratios are presented in Table 2 for the four different holding periods and the five different trading strategies. The Sharpe ratio is a return-to-risk ratio, with a high value indicating good return for relatively low risk. The Sharpe ratio is calculated as: $S = \frac{E[R_a - R_b]}{\sqrt{\text{var}(R_a - R_b)}}$, where R_a is the return of a single asset and R_b is the risk-free return of a 10-year U.S. Treasury note.

The returns from this experimental trading system are fairly low, although they do beat the baselines. A one-way ANOVA test among the experimental, momentum and S&P strategies using the percent returns from the individual trades yields p values of 0.06493, 0.08162, 0.1792, and 0.4164, respectively, thus failing to reject the null hypothesis that the returns are not significantly higher.[3]

[3]An anonymous reviewer observed that Tetlock (2007) showed a statistically significant improvement from the use of sentiment, apparently contradicting this result. Tetlock's (2007) sentiment-based trading strategy used a safety zone (see Section 4.2.2), and was never compared to a realistic baseline or control strategy. Instead, Tetlock's (2007) significance test was conducted to demonstrate that his returns (positive in 12 of 15 calendar years of historical market data) were unlikely to have been generated by chance from a normal distribution centred at zero.

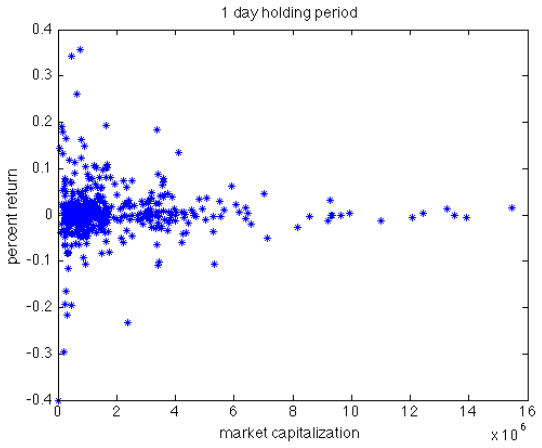


Figure 1: Percent returns for 1 day holding period versus market capitalization of the traded stocks.

Furthermore, the means and medians of all three trading strategies are approximately the same and centred around 0. The standard deviations of the experimental strategy and the momentum strategy are nearly identical, differing only in the thousandths digit. The standard deviations for the S&P strategy differ from the other two strategies due to the fact that the strategy buys and sells the entire S&P 500 index and not the individual stocks described in the news articles. There is, in fact, no convincing evidence that discrete sentiment class leads to an improved trading strategy from this or any other study with which we are familiar, based on their published details. One may note, however, that the returns from the experimental strategy have slightly higher Sharpe ratios than either of the baselines.

One may also note that using a sentiment analyzer mostly beats training directly on market data. This vindicates using sentiment annotation as an information source.

Figure 1 shows the market capitalizations of each individual trade's companies plotted against their percent return with a 1 day holding period. The correlation between the two variables is not significant. Returns for the other holding periods are similarly dispersed.

The importance of having good baselines is demonstrated by the fact that when we annualize our returns for the 3-day holding period, we get 70.086%. This number appears very high, but the annualized return from the momentum strategy is

70.066%[4], which is not significantly lower.

Figure 2 shows the percent change in share value plotted against the raw SVM score for the different holding periods. We can see a weak correlation between the two. For the 30 days, 5 days, 3 days, and 1 day holding periods, the correlations are 0.017, 0.16, 0.16, and 0.16, respectively. The line of best fit is shown. This prompts our next experiment.

4.2 Utilizing SVM scores

4.2.1 Experiment Two: Variable Single Threshold

Before, we labelled documents as positive (negative) when the score was above (below) 0, because 0 was the decision boundary. But 0 might not be the best threshold, θ, for high returns. To determine θ, we divided the evaluation dataset, i.e. the dataset with news articles dated on or after March 10, 2005, into two folds having an equal number of documents with positive and negative sentiment. We used the first fold to determine θ and traded using the data from the second fold and θ. For every news article, if the SVM score for that article is above (below) θ, then we go long (short) on the appropriate stock on the day the article was released. A separate theta was determined for each holding period. We varied θ from -1 to 1 in increments of 0.1.

Using this method, we were able to obtain significantly higher returns. In order of 30, 5, 3, and 1 day holding periods, the returns were 0.057%, 1.107%, 1.238%, and 0.745% ($p < 0.001$ in every case). This is a large improvement over the previous returns, as they are average per-position figures.[5]

4.2.2 Experiment Three: Safety Zones

For every news item classified, SVM outputs a score. For a binary SVM with a linear kernel function f, given some feature vector $\mathbf{x}$, $f(\mathbf{x})$ can be viewed as the signed distance of $\mathbf{x}$ from the decision boundary (Boser et al., 1992). It is then possibly justified to interpret raw SVM scores as degrees to which an article is positive or negative.

As in the previous section, we separate the evaluation set into the same two folds, only now we use two thresholds, $\theta \geq \zeta$. We will go long when the SVM score is above θ, abstain when the SVM score is between θ and ζ, and go short when the SVM score is below ζ. This is a strict generalization of the above experiment, in which $\zeta = \theta$.

For convenience, we will assume in this section that $\zeta = -\theta$, leaving us again with one parameter to estimate. We again vary θ from 0 to 1 in increments of 0.1. Figure 3 shows the returns as a function of θ for each holding period on the development dataset. If we increased the upper bound on θ to be greater than 1, then there would be too few trading examples (less than 10) to reliably calculate the Sharpe ratio. Using this method with $\theta = 1$, we were able to obtain even higher returns: 3.843%, 1.851%, 1.691, and 2.251% for the 30, 5, 3, and 1 day holding periods, versus 0.057%, 1.107%, 1.238%, and 0.745% in the second task-based experiment.

Representation	Accuracy	30 days	5 days	3 days	1 day
term_presence	80.164%	3.843%	1.851%	1.691%	2.251%
bm25_freq	81.143%	1.110%	1.770%	1.781%	0.814%
bm25_freq_dnc	62.094%	3.458%	2.834%	2.813%	2.586%
bm25_freq_sw	79.827%	0.390%	1.685%	1.581%	1.250%
freq	79.276%	1.596%	1.221%	1.344%	1.330%
freq_with_sw	75.564%	1.752%	0.638%	1.056%	2.205%

Table 3: Sentiment classification accuracy (average 10-fold cross-validation) and trade returns of different feature sets and term frequency weighting schemes in Exp. 3. The same folds were used for the different representations. The non-annualized returns are presented in columns 3-6.

4.3 Experiment Four: Feature Selection

In our final experiment, let us now hold the trading strategy fixed (at the third one, with safety zones) and turn to the underlying sentiment analyzer. With a good trading strategy in place, it is clearly possible to vary some aspect of the sentiment analyzer in order to determine its best setting in this context. We will measure both market return and classifier accuracy to determine whether they agree. Is the latter a suitable proxy for the former? Indeed, we may hope that classifier accuracy will be more portable to other possible tasks, but then it must at least correlate well with task-based performance.

In addition to evaluating those feature sets attempted in Section 3.2, we now hypothesize that the passive voice may be useful to emphasize in our representations, as the existential passive can be used to evade responsibility. So we add to the

[4]The momentum strategy has a different number of possible trades in any actual calendar year because it is a function of the holding period.

[5]Training directly on market data, by comparison, yields -0.258%, -0.282%, -0.036% and -0.388%, respectively.

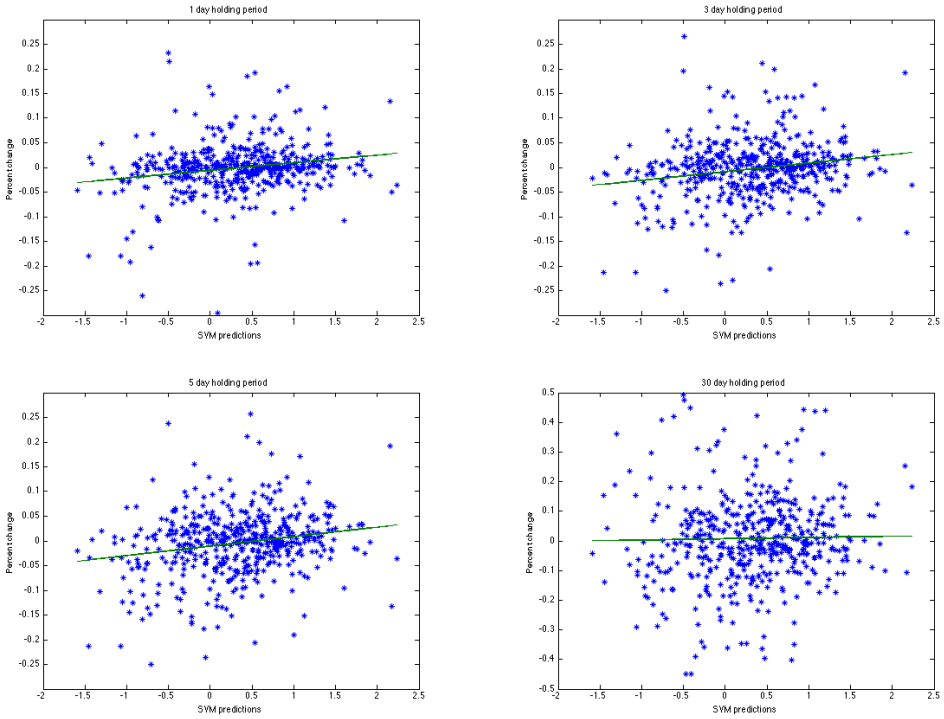


Figure 2: Percent change of trade returns plotted against SVM values for the 1, 3, 5, and 30 day holding periods in Exp. 1. Graphs are cropped to zoom in.

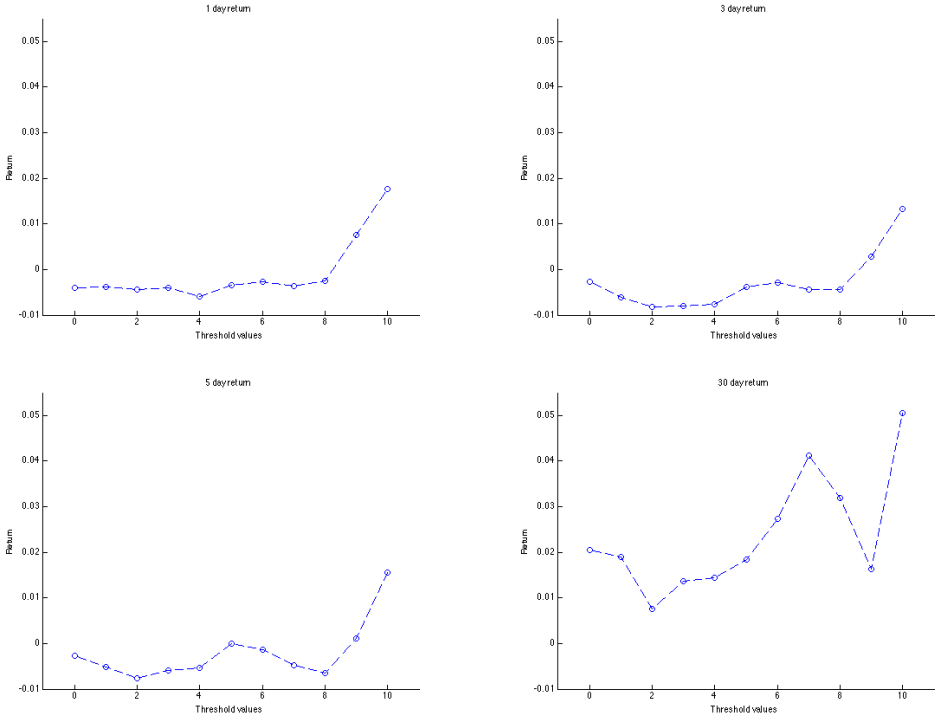


Figure 3: Returns for different thresholds on the development data for 30, 5, 3, and 1 day holding periods in Exp. 2 with safety zone.

BM25 weighted vector the counts of word tokens ending in "n" or "d" as well as the total count of every conjugated form of the copular verb: "be", "is", "am", "are", "were", "was", and "been". These three features are superficial indicators of the passive voice. Clearly, we could have used a part-of-speech tagger to detect the passive voice more reliably, but we are more interested here in how well our task-based evaluation will correspond to a more customary classifier-accuracy evaluation, rather than finding the world's best indicators of the passive voice.

Table 3 presents returns obtained from these 6 feature sets. The feature set with BM25-weighted term frequencies plus the number of copulars and tokens ending in "n", "d" (bm25_freq_dnc) yields higher returns than any other representation attempted on the 5, 3, and 1 day holding periods, and the second-highest on the 30 days holding period. But it has the worst classification accuracy by far: a full 18 percentage points below term presence. This is a very compelling illustration of how misleading an intrinsic evaluation can be.

5 Conclusion

In this paper, we examined sentiment analysis applied to stock trading strategies. We built a binary sentiment classifier that achieves high accuracy when tested on movie data and financial news data from *Reuters*. In four task-based experiments, we evaluated the usefulness of sentiment analysis to simple trading strategies. Although high annual returns are achieved simply by utilizing sentiment labels while trading, they can be increased by incorporating the output of the SVM's decision function. But classification accuracy alone is not an accurate predictor of task-based performance. This calls into question the suitability of intrinsic sentiment classification accuracy, particularly (as here) when the relative cost of a task-based evaluation may be comparably low. We have also determined that training on human-annotated sentiment does in fact perform better than training on market returns themselves. So sentiment analysis is an important component, but it must be tuned against task data.

Our price data only included adjusted opening and closing prices and most of our news data contain only the date of the article, with no specific time. This limits our ability to test much shorter-term trading strategies.

Deriving sentiment labels for supervised training is an important topic for future study, as is inferring the sentiment of published news from stock price fluctuations instead of the reverse. We should also study how "sentiment" is defined in the financial world. This study has used a rather general definition of news sentiment, and a more precise definition may improve trading performance.

Acknowledgments

This research was supported by the Canadian Network Centre of Excellence in Graphics, Animation and New Media (GRAND).

References

Khurshid Ahmad, David Cheng, and Yousif Almas. 2006. Multi-lingual sentiment analysis of financial news streams. In *Proceedings of the 1st International Conference on Grid in Finance*.

Pranav Anand and Kevin Reschke. 2010. Verb classes as evaluativity functor classes. In *Interdisciplinary Workshop on Verbs: The Identification and Representation of Verb Features (Verb 2010)*.

Werner Antweiler and Murray Z Frank. 2004. Is all that talk just noise? the information content of internet stock message boards. *The Journal of Finance*, 59(3):1259–1294.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

Matthew Butler and Vlado Keselj. 2009. Financial forecasting using character n-gram analysis and readability scores of annual reports. In *Proceedings of Canadian AI'2009*, Kelowna, BC, Canada, May.

Wesley S. Chan. 2003. Stock price reaction to news and no-news: Drift and reversal after headlines. *Journal of Financial Economics*, 70(2):223–260.

Sanjiv R. Das and Mike Y. Chen. 2007. Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science*, 53(9):1375–1388.

Lingjia Deng, Yoonjung Choi, and Janyce Wiebe. 2013. Benefactive/malefactive event and writer attitude annotation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 120–125. Association for Computational Linguistics.

Ann Devitt and Khurshid Ahmad. 2007. Sentiment polarity identification in financial news: A cohesion-based approach. In *Proceedings of the ACL*.

Brett Drury and J. J. Almeida. 2011. Identification of fine grained feature based event and sentiment phrases from business news stories. In *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, WIMS '11, pages 27:1–27:7, New York, NY, USA. ACM.

Robert F. Engle and Victor K. Ng. 1993. Measuring and testing the impact of news on volatility. *The Journal of Finance*, 48(5):1749–1778.

Tak-Chung Fu, Ka ki Lee, Donahue C. M. Sze, Fu-Lai Chung, Chak man Ng, and Chak man Ng. 2008. Discovering the correlation between stock time series and financial news. In *Web Intelligence*, pages 880–883.

Thorsten Joachims. 1999. Making large-scale svm learning practical. advances in kernel methods-support vector learning, b. schölkopf and c. burges and a. smola.

Moshe Koppel and Itai Shtrimberg. 2004. Good news or bad news? let the market decide. In *AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pages 86–88. Press.

Chenghua Lin and Yulan He. 2009. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 375–384, New York, NY, USA. ACM.

Victor Niederhoffer. 1971. The analysis of world events and stock prices. *Journal of Business*, pages 193–219.

Neil O'Hare, Michael Davy, Adam Bermingham, Paul Ferguson, Páraic Sheridan, Cathal Gurrin, and Alan F. Smeaton. 2009. Topic-dependent sentiment analysis of financial blogs. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion measurement*.

Georgios Paltoglou and Mike Thelwall. 2010. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the ACL*, pages 1386–1395. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the ACL*, pages 271–278.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA. Association for Computational Linguistics.

John C. Platt. 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press.

William F Sharpe. 1966. Mutual fund performance. *The Journal of business*, 39(1):119–138.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of EMNLP*, pages 1422–1432.

Paul C. Tetlock. 2007. Giving content to investor sentiment: The role of media in the stock market. *The Journal of Finance*, 62(3):1139–1168.

Wenbin Zhang and Steven Skiena. 2010. Trading strategies to exploit blog and news sentiment. In *The 4th International AAAI Conference on Weblogs and Social Media*.

Edge-Linear First-Order Dependency Parsing with Undirected Minimum Spanning Tree Inference

Effi Levi[1] **Roi Reichart**[2] **Ari Rappoport**[1]
[1]Institute of Computer Science, The Hebrew Univeristy
[2]Faculty of Industrial Engineering and Management, Technion, IIT
{efle|arir}@cs.huji.ac.il roiri@ie.technion.ac.il

Abstract

The run time complexity of state-of-the-art inference algorithms in graph-based dependency parsing is super-linear in the number of input words (n). Recently, pruning algorithms for these models have shown to cut a large portion of the graph edges, with minimal damage to the resulting parse trees. Solving the inference problem in run time complexity determined solely by the number of edges (m) is hence of obvious importance.

We propose such an inference algorithm for first-order models, which encodes the problem as a minimum spanning tree (MST) problem in an *undirected graph*. This allows us to utilize state-of-the-art undirected MST algorithms whose run time is $O(m)$ at expectation and with a very high probability. A directed parse tree is then inferred from the undirected MST and is subsequently improved with respect to the directed parsing model through local greedy updates, both steps running in $O(n)$ time. In experiments with 18 languages, a variant of the first-order MSTParser (McDonald et al., 2005b) that employs our algorithm performs very similarly to the original parser that runs an $O(n^2)$ directed MST inference.

1 Introduction

Dependency parsers are major components of a large number of NLP applications. As application models are applied to constantly growing amounts of data, efficiency becomes a major consideration.

In graph-based dependency parsing models (Eisner, 2000; McDonald et al., 2005a; McDonald et al., 2005b; Carreras, 2007; Koo and Collins, 2010b), given an n word sentence and a model order k, the run time of exact inference is $O(n^3)$ for $k = 1$ and $O(n^{k+1})$ for $k > 1$ in the projective case (Eisner, 1996; McDonald and Pereira, 2006). In the non-projective case it is $O(n^2)$ for $k = 1$ and NP-hard for $k \geq 2$ (McDonald and Satta, 2007). [1] Consequently, a number of approximated parsers have been introduced, utilizing a variety of techniques: the Eisner algorithm (McDonald and Pereira, 2006), belief propagation (Smith and Eisner, 2008), dual decomposition (Koo and Collins, 2010b; Martins et al., 2013) and multi-commodity flows (Martins et al., 2009; Martins et al., 2011). The run time of all these approximations is super-linear in n.

Recent pruning algorithms for graph-based dependency parsing (Rush and Petrov, 2012; Riedel et al., 2012; Zhang and McDonald, 2012) have shown to cut a very large portion of the graph edges, with minimal damage to the resulting parse trees. For example, Rush and Petrov (2012) demonstrated that a single $O(n)$ pass of vine-pruning (Eisner and Smith, 2005) can preserve $> 98\%$ of the correct edges, while ruling out $> 86\%$ of all possible edges. Such results give strong motivation to solving the inference problem in a run time complexity that is determined solely by the number of edges (m). [2]

[1]We refer to parsing approaches that produce only projective dependency trees as projective parsing and to approaches that produce all types of dependency trees as non-projective parsing.

[2]Some pruning algorithms require initial construction of the full graph, which requires exactly $n(n-1)$ edge weight computations. Utilizing other techniques, such as length-dictionary pruning, graph construction and pruning can be

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2104–2113,
Berlin, Germany, August 7-12, 2016.

In this paper we propose to formulate the inference problem in first-order (arc-factored) dependency parsing as a minimum spanning tree (MST) problem in *an undirected graph*. Our formulation allows us to employ state-of-the-art algorithms for the MST problem in undirected graphs, whose run time depends solely on the number of edges in the graph. Importantly, a parser that employs our undirected inference algorithm can generate all possible trees, projective and non-projective.

Particularly, the undirected MST problem (§ 2) has a randomized algorithm which is $O(m)$ at expectation and with a very high probability ((Karger et al., 1995)), as well as an $O(m \cdot \alpha(m,n))$ worst-case deterministic algorithm (Pettie and Ramachandran, 2002), where $\alpha(m,n)$ is a certain natural inverse of Ackermann's function (Hazewinkel, 2001). As the inverse of Ackermann's function grows extremely slowly [3] the deterministic algorithm is in practice linear in m (§ 3). In the rest of the paper we hence refer to the run time of these two algorithms as *practically linear* in the number of edges m.

Our algorithm has four steps (§ 4). First, it encodes the first-order dependency parsing inference problem as an undirected MST problem, in up to $O(m)$ time. Then, it computes the MST of the resulting undirected graph. Next, it infers a *unique* directed parse tree from the undirected MST. Finally, the resulting directed tree is greedily improved with respect to the directed parsing model. Importantly, the last two steps take $O(n)$ time, which makes the total run time of our algorithm $O(m)$ at expectation and with very high probability. [4]

We integrated our inference algorithm into the first-order parser of (McDonald et al., 2005b) and compared the resulting parser to the original parser which employs the Chu-Liu-Edmonds algorithm (CLE, (Chu and Liu, 1965; Edmonds, 1967)) for inference. CLE is the most efficient exact inference algorithm for graph-based first-order non-projective parsers, running at $O(n^2)$ time.[5]

We experimented (§ 5) with 17 languages from the CoNLL 2006 and 2007 shared tasks on multilingual dependency parsing (Buchholz and Marsi, 2006; Nilsson et al., 2007) and in three English setups. Our results reveal that the two algorithms perform very similarly. While the averaged unlabeled attachment accuracy score (UAS) of the original parser is 0.97% higher than ours, in 11 of 20 test setups the number of sentences that are better parsed by our parser is larger than the number of sentences that are better parsed by the original parser.

Importantly, in this work we present an edge-linear first-order dependency parser which achieves similar accuracy to the existing one, making it an excellent candidate to be used for efficient MST computation in k-best trees methods, or to be utilized as an inference/initialization subroutine as a part of more complex approximation frameworks such as belief propagation. In addition, our model produces a different solution compared to the existing one (see Table 2), paving the way for using methods such as dual decomposition to combine these two models into a superior one.

Undirected inference has been recently explored in the context of transition based parsing (Gómez-Rodríguez and Fernández-González, 2012; Gómez-Rodríguez et al., 2015), with the motivation of preventing the propagation of erroneous early edge directionality decisions to subsequent parsing decisions. Yet, to the best of our knowledge this is the first paper to address undirected inference for graph based dependency parsing. Our motivation and algorithmic challenges are substantially different from those of the earlier transition based work.

2 Undirected MST with the Boruvka Algorithm

In this section we define the MST problem in undirected graphs. We then discuss the Burovka algorithm (Boruvka, 1926; Nesetril et al., 2001) which forms the basis for the randomized algorithm of (Karger et al., 1995) we employ in this paper. In the next section we will describe the Karger et al. (1995) algorithm in more details.

Problem Definition. For a connected undirected graph $G(V,E)$, where V is the set of n vertices

jointly performed in $O(n)$ steps. We therefore do not include initial graph construction and pruning in our complexity computations.

[3] $\alpha(m,n)$ is less than 5 for any practical input sizes (m,n).

[4] The output dependency tree contains exactly $n-1$ edges, therefore $m \geq n-1$, which makes $O(m)+O(n)=O(m)$.

[5] CLE has faster implementations: $O(m+nlogn)$ (Gabow et al., 1986) as well as $O(mlogn)$ for sparse graphs (Tarjan, 1977), both are super-linear in n for connected graphs. We refer here to the classical implementation employed by modern parsers (e.g. (McDonald et al., 2005b; Martins et al., 2013)).

and E the set of m weighted edges, the MST problem is defined as finding the sub-graph of G which is the tree (a connected acyclic graph) with the lowest sum of edge weights. The opposite problem – finding the maximum spanning tree – can be solved by the same algorithms used for the minimum variant by simply negating the graph's edge weights.

Graph Contraction. In order to understand the Boruvka algorithm, let us first define the *Graph Contraction* operation. For a given undirected graph $G(V, E)$ and a subset $\tilde{E} \subseteq E$, this operation creates a new graph, $G_C(V_C, E_C)$. In this new graph, V_C consists of a vertex for each connected component in $\tilde{G}(V, \tilde{E})$ (these vertices are referred to as *super-vertices*). E_C, in turn, consists of one edge, $(\hat{u}, \hat{v})$, for each edge $(u, v) \in E \setminus \tilde{E}$, where $\hat{u}, \hat{v} \in V_C$ correspond to $\tilde{G}$'s connected components to which u and v respectively belong. Note that this definition may result in multiple edges between two vertices in V_C (denoted *repetitive edges*) as well as in edges from a vertex in V_C to itself (denoted *self edges*).

Algorithm 1 The basic step of the Boruvka algorithm for the undirected MST problem.

Contract_graph

Input: a graph $G(V, E)$, a subset $\tilde{E} \subseteq E$
$C \leftarrow$ connected components of $\tilde{G}(V, \tilde{E})$
return $G_C(C, E \setminus \tilde{E})$

Boruvka-step

Input: a graph $G(V, E)$
1: **for all** $(u, v) \in E$ **do**
2: **if** $w(u, v) < w(u.minEdge)$ **then**
3: $u.minEdge \leftarrow (u, v)$
4: **end if**
5: **if** $w(u, v) < w(v.minEdge)$ **then**
6: $v.minEdge \leftarrow (u, v)$
7: **end if**
8: **end for**
9: **for all** $v \in V$ **do**
10: $E_m \leftarrow E_m \cup \{v.minEdge\}$
11: **end for**
12: $G_B(V_B, E_B) \leftarrow$ Contract_graph$(G(V, E), E_m)$
13: Remove from E_B self edges and non-minimal repetitive edges
14: **return** $G_B(V_B, E_B), E_m$

The Boruvka-Step. Next, we define the basic step of the Borukva algorithm (see example in Figure 1 and pseudocode in Algorithm 1). In each such step, the algorithm creates a subset $E_m \subset E$ by selecting the minimally weighted edge for each vertex in the input graph $G(V, E)$ (Figure 1 (a,b) and Algorithm 1 (lines 1-11)). Then, it performs the contraction operation on the graph G and E_m to receive a new graph $G_B(V_B, E_B)$ (Figure 1 (c) and Algorithm 1 (12)). Finally, it removes from E_B all self-edges and repetitive edges that are not the minimal edges between the vertices V_B's which they connect (Figure 1 (d) and Algorithm 1 (13)). The set E_m created in each such step is guaranteed to consist only of edges that belong to G's MST and is therefore also returned by the Boruvka step.

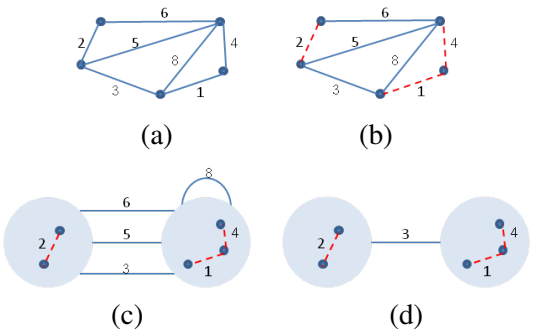


Figure 1: An illustration of a *Boruvka step*: (a) The original graph; (b) Choosing the minimal edge for each vertex (marked in red); (c) The contracted graph; (d) The contracted graph after removing one self edge and two non-minimal repetitive edges.

The Boruvka algorithm runs successive Boruvka-steps until it is left with a single super-vertex. The MST of the original graph G is given by the unification of the E_m sets returned in each step. The resulting computational complexity is $O(m \log n)$ (Nesetril et al., 2001). We now turn to describe how the undirected MST problem can be solved in a time practically linear in the number of graph edges.

3 Undirected MST in Edge Linear Time

There are two algorithms that solve the undirected MST problem in time practically linear in the number of edges in the input graph. These algorithms are based on substantially different approaches: one is deterministic and the other is randomized [6].

[6]Both these algorithms deal with a slightly more general case where the graph is not necessarily connected, in which case the *minimum spanning forest (MSF)* is computed. In our case, where the graph is connected, the MSF reduces to an MST.

The complexity of the first, deterministic, algorithm (Chazelle, 2000; Pettie and Ramachandran, 2002) is $O(m \cdot \alpha(m,n))$, where $\alpha(m,n)$ is a natural inverse of Ackermann's function, whose value for any practical values of n and m is lower than 5. As this algorithm employs very complex data-structures, we do not implement it in this paper.

The second, randomized, algorithm (Karger et al., 1995) has an expected run time of $O(m+n)$ (which for connected graphs is $O(m)$), and this run time is achieved with a high probability of $1 - exp(-\Omega(m))$. [7] In this paper we employ only this algorithm for first-order graph-based parsing inference, and hence describe it in details in this section.

Definitions and Properties. We first quote two properties of undirected graphs (Tarjan, 1983): (1) The **cycle property**: The heaviest edge in a cycle in a graph does not appear in the MSF; and (2) The **cut property**: For any proper nonempty subset V' of the graph vertices, the lightest edge with exactly one endpoint in V' is included in the MSF.

We continue with a number of definitions and observations. Given an undirected graph $G(V,E)$ with weighted edges, and a forest F in that graph, $F(u,v)$ is the path in that forest between u and v (if such a path exists), and $s_F(u,v)$ is the maximum weight of an edge in $F(u,v)$ (if the path does not exist then $s_F(u,v) = \infty$). An edge $(u,v) \in E$ is called *F-heavy* if $s(u,v) > s_F(u,v)$, otherwise it is called *F-light*. An alternative equivalent definition is that an edge is *F-heavy* if adding it to F creates a cycle in which it is the heaviest edge. An important observation (derived from the cycle property) is that for any forest F, no *F-heavy* edge can possibly be a part of an MSF for G. It has been shown that given a forest F, all the *F-heavy* edges in G can be found in $O(m)$ time (Dixon et al., 1992; King, 1995).

Algorithm. The randomized algorithm can be outlined as follows (see pseudocode in algorithm 2): first, two successive Boruvka-steps are applied to the graph (line 4, *Boruvka-step2* stands for two successive Boruvka-steps), reducing the number of vertices by (at least) a factor of 4 to receive a contracted graph G_C and an edge set E_m (§ 2). Then, a subgraph G_s is randomly constructed, such that each edge in G_C, along with the vertices which it connects, is included in G_s with probability $\frac{1}{2}$ (lines 5-10). Next, the algorithm is recursively applied to G_s to obtain its minimum spanning forest F (line 11). Then, all *F-heavy* edges are removed from G_C (line 12), and the algorithm is recursively applied to the resulting graph to obtain a spanning forest F_C (line 13). The union of that forest with the edges E_m forms the requested spanning forest (line 14).

Algorithm 2 Pseudocode for the Randomized MSF algorithm of(Karger et al., 1995).

Randomized_MSF

Input: a graph $G(V,E)$

1: **if** E is empty **then**
2: **return** $\emptyset$
3: **end if**
4: $G_C(V_C,E_C), E_m \leftarrow$ Boruvka-step2(G)
5: **for all** $(u,v) \in E_C$ **do**
6: **if** coin-flip == head **then**
7: $E_s \leftarrow E_s \cup \{(u,v)\}$
8: $V_s \leftarrow V_s \cup \{u,v\}$
9: **end if**
10: **end for**
11: $F \leftarrow$ Randomized_MSF($G_s(V_s,E_s)$)
12: remove all *F-heavy* edges from $G_C(V_C,E_C)$
13: $F_C \leftarrow$ Randomized_MSF($G_C(V_C,E_C)$)
14: **return** $F_C \cup E_m$

Correctness. The correctness of the algorithm is proved by induction. By the cut property, every edge returned by the Boruvka step (line 4), is part of the MSF. Therefore, the rest of the edges in the original graph's MSF form an MSF for the contracted graph. The removed *F-heavy* edges are, by the cycle property, not part of the MSF (line 12). By the induction assumption, the MSF of the remaining graph is then given by the second recursive call (line 13).

4 Undirected MST Inference for Dependency Parsing

There are several challenges in the construction of an *undirected MST parser*: an MST parser that employs an undirected MST algorithm for inference.[8] These challenges stem from the mismatch between the undirected nature of the inference algorithm and the directed nature of the resulting

[7] This complexity analysis is beyond the scope of this paper.

[8] Henceforth, we refer to an MST parser that employs a directed MST algorithm for inference as *directed MST parser*.

parse tree.

The first problem is that of *undirected encoding*. Unlike directed MST parsers that explicitly encode the directed nature of dependency parsing into a directed input graph to which an MST algorithm is applied (McDonald et al., 2005b), an undirected MST parser needs to encode directionality information into an undirected graph. In this section we consider two solutions to this problem.

The second problem is that of *scheme conversion*. The output of an undirected MST algorithm is an undirected tree while the dependency parsing problem requires finding a directed parse tree. In this section we show that for rooted undirected spanning trees there is only one way to define the edge directions under the constraint that the root vertex has no incoming edges and that each non-root vertex has exactly one incoming edge in the resulting directed spanning tree. As dependency parse trees obey the first constraint and the second constraint is a definitive property of directed trees, the output of an undirected MST parser can be transformed into a directed tree using a simple $O(n)$ time procedure.

Unfortunately, as we will see in § 5, even with our best undirected encoding method, an undirected MST parser does not produce directed trees of the same quality as its directed counterpart. At the last part of this section we therefore present a simple, $O(n)$ time, *local enhancement procedure*, that improves the score of the directed tree generated from the output of the undirected MST parser with respect to the edge scores of a standard directed MST parser. That is, our procedure improves the output of the undirected MST parser with respect to a directed model without having to compute the MST of the latter, which would take $O(n^2)$ time.

We conclude this section with a final remark stating that the output class of our inference algorithm is non-projective. That is, it can generate all possible parse trees, projective and non-projective.

Undirected Encoding Our challenge here is to design an encoding scheme that encodes directionality information into the graph of the undirected MST problem. One approach would be to compute directed edge weights according to a feature representation scheme for directed edges (e.g. one of the schemes employed by existing directed MST parsers) and then transform these directed weights into undirected ones.

Specifically, given two vertices u and v with directed edges (u, v) and (v, u), weighted with $s_d(u, v)$ and $s_d(v, u)$ respectively, the goal is to compute the weight $s_u(u\hat{,}v)$ of the undirected edge $(u\hat{,}v)$ connecting them in the undirected graph. We do this using a pre-determined function $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, such that $f(s_d(u, v), s_d(v, u)) = s_u(u\hat{,}v)$. f can take several forms including mean, product and so on. In our experiments the mean proved to be the best choice.

Training with the above approach is implemented as follows. w, the parameter vector of the parser, consists of the weights of *directed features*. At each training iteration, w is used for the computation of $s_d(u, v) = w \cdot \phi(u, v)$ and $s_d(v, u) = w \cdot \phi(v, u)$ (where $\phi(u, v)$ and $\phi(v, u)$ are the feature representations of these directed edges). Then, f is applied to compute the undirected edge score $s_u(u\hat{,}v)$. Next, the undirected MST algorithm is run on the resulting weighted undirected graph, and its output MST is transformed into a directed tree (see below). Finally, this directed tree is used for the update of w with respect to the gold standard (directed) tree.

At test time, the vector w which resulted from the training process is used for s_d computations. Undirected graph construction, undirected MST computation and the undirected to directed tree conversion process are conducted exactly as in training. [9]

Unfortunately, preliminary experiments in our development setup revealed that this approach yields parse trees of much lower quality compared to the trees generated by the directed MST parser that employed the original directed feature set. In § 5 we discuss these results in details.

An alternative approach is to employ an undirected feature set. To implement this approach, we employed the feature set of the MST parser ((McDonald et al., 2005a), Table 1) with one difference: some of the features are directional, distinguishing between the properties of the source (parent) and the target (child) vertices. We stripped those features from that information, which resulted in an undirected version of the feature set.

Under this feature representation, training with undirected inference is simple. w, the parameter vector of the parser, now consists of the weights

[9] In evaluation setup experiments we also considered a variant of this model where the training process utilized directed MST inference. As this variant performed poorly, we exclude it from our discussion in the rest of the paper.

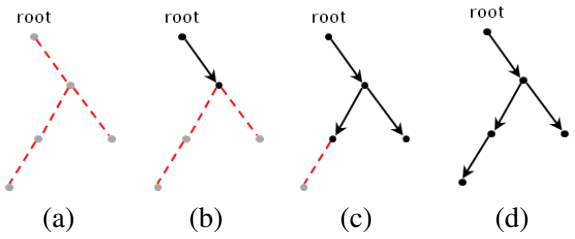


Figure 2: An illustration of directing an undirected tree, given a constrained root vertex: (a) The initial undirected tree; (b) Directing the root's outgoing edge; (c) Directing the root's child's outgoing edges; (d) Directing the last remaining edge, resulting in a directed tree.

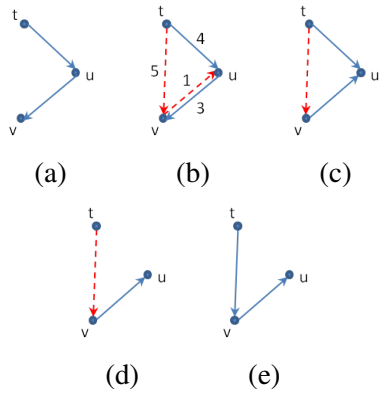


Figure 3: An illustration of the local enhancement procedure for an edge (u, v) in the du-tree. Solid lines indicate edges in the du-tree, while dashed lines indicate edges not in the du-tree. (a) Example subtree; (b) Evaluate $gain = s_d(t,u) + s_d(u,v) - (s_d(t,v) + s_d(v,u)) = 4+3-(5+1) = 1$; (c) In case a modification is made, first replace (u, v) with (v, u); and then (d) Remove the edge (t, u); and, finally, (e) Add the edge (t, v).

of *undirected features*. Once the undirected MST is computed by an undirected MST algorithm, w can be updated with respect to an undirected variant of the gold parse trees. At test time, the algorithm constructs an undirected graph using the vector w resulted from the training process. This graph's undirected MST is computed and then transformed into a directed tree.

Interestingly, although this approach does not explicitly encode edge directionality information into the undirected model, it performed very well in our experiments (§ 5), especially when combined with the local enhancement procedure described below.

Scheme Conversion Once the undirected MST is found, we need to direct its edges in order for the end result to be a directed dependency parse tree. Following a standard practice in graph-based dependency parsing (e.g. (McDonald et al., 2005b)), before inference is performed we add a dummy root vertex to the initial input graph with edges connecting it to all of the other vertices in the graph. Consequently, the final undirected tree will have a designated root vertex. In the resulting directed tree, this vertex is constrained to have only outgoing edges. As observed by Gómez-Rodríguez and Fernández-González (2012), this effectively forces the direction for the rest of the edges in the tree.

Given a root vertex that follows the above constraint, and together with the definitive property of directed trees stating that each non-root vertex in the graph has exactly one incoming edge, we can direct the edges of the undirected tree using a simple BFS-like algorithm (Figure 2). Starting with the root vertex, we mark its undirected edges as outgoing, mark the vertex itself as *done* and its descendants as *open*. We then recursively repeat the same procedure for each open vertex until there are no such vertices left in the tree, at which point we have a directed tree. Note that given the constraints on the root vertex, there is no other way to direct the undirected tree edges. This procedure runs in $O(n)$ time, as it requires a constant number of operations for each of the $n-1$ edges of the undirected spanning tree.

In the rest of the paper we refer to the *directed* tree generated by the undirected and directed MST parsers as *du-tree* and *dd-tree* respectively.

Local Enhancement Procedure As noted above, experiments in our development setup (§ 5) revealed that the directed parser performs somewhat better than the undirected one. This motivated us to develop a local enhancement procedure that improves the tree produced by the undirected model with respect to the directed model without compromising our $O(m)$ run time. Our enhancement procedure is motivated by development experiments, revealing the much smaller gap between the quality of the du-tree and dd-tree of the same sentence under undirected evaluation compared to directed evaluation (§ 5 demonstrates this for test results).

For a du-tree that contains the vertex u and the edges (t, u) and (u, v), we therefore consider the replacement of (u, v) with (v, u). Note that after this change our graph would no longer be a directed tree, since it would cause u to have two parents, v and t, and v to have no parent. This, however, can be rectified by replacing the edge (t, u) with the edge (t, v).

It is easy to infer whether this change results in a better (lower weight) spanning tree under the directed model by computing the equation: $gain = s_d(t,u) + s_d(u,v) - (s_d(t,v) + s_d(v,u))$, where $s_d(x,y)$ is the score of the edge (x,y) according to the directed model. This is illustrated in Figure 3.

Given the du-tree, we traverse its edges and compute the above gain for each. We then choose the edge with the maximal positive gain, as this forms the maximal possible decrease in the directed model score using modifications of the type we consider, and perform the corresponding modification. In our experiments we performed this procedure five times per inference problem.[10] This procedure performs a constant number of operations for each of the $n-1$ edges of the du-tree, resulting in $O(n)$ run time.

Output Class. Our undirected MST parser is non-projective. This stems from the fact that the undirected MST algorithms we discuss in § 3 do not enforce any structural constraint, and particularly the non-crossing constraint, on the resulting undirected MST. As the scheme conversion (edge directing) and the local enhancement procedures described in this section do not enforce any such constraint as well, the resulting tree can take any possible structure.

5 Experiments and Results

Experimental setup We evaluate four models: (a) The original directed parser (D-MST, (McDonald et al., 2005b)); (b) Our undirected MST parser with undirected features and with the local enhancement procedure (U-MST-uf-lep);[11] (c) Our undirected MST parser with undirected features but without the local enhancement procedure (U-MST-uf); and (d) Our undirected MST parser with directed features (U-MST-df). All models are implemented within the MSTParser code[12].

The MSTParser does not prune its input graphs. To demonstrate the value of undirected parsing for sparse input graphs, we implemented the length-dictionary pruning strategy which eliminates all edges longer than the maximum length observed for each *directed* head-modifier POS pair in the training data. An undirected edge $\hat{(u,v)}$ is pruned iff both directed edges (u,v) and (v,u) are to be pruned according to the pruning method. To estimate the accuracy/graph-size tradeoff provided by undirected parsing (models (b)-(d)), we apply the pruning strategy only to these models leaving the the D-MST model (model (a)) untouched. This way D-MST runs on a complete directed graph with n^2 edges.

Our models were developed in a monolingual setup: training on sections 2-21 of WSJ PTB (Marcus et al., 1993) and testing on section 22. The development phase was devoted to the various decisions detailed throughout this paper and to the tuning of the single hyperparameter: the number of times the local enhancement procedure is executed.

We tested the models in 3 English and 17 multilingual setups. The English setups are: (a) *PTB*: training on sections 2-21 of the WSJ PTB and testing on its section 23; (b) *GENIA*: training with a random sample of 90% of the 4661 GENIA corpus (Ohta et al., 2002) sentences and testing on the other 10%; and (c) *QBank*: a setup identical to (b) for the 3987 QuestionBank (Judge et al., 2006) sentences. Multilingual parsing was performed with the multilingual datasets of the CoNLL 2006 (Buchholz and Marsi, 2006) and 2007 (Nilsson et al., 2007) shared tasks on multilingual dependency parsing, following their standard train/test split. Following previous work, punctuation was excluded from the evaluation.

Length-dictionary pruning reduces the number of undirected edges by 27.02% on average across our 20 setups (std = 11.02%, median = 23.85%), leaving an average of 73.98% of the edges in the undirected graph. In 17 of 20 setups the reduction is above 20%. Note that the number of edges in a complete directed graph is twice the number in its undirected counterpart. Therefore, on average, the number of input edges in the pruned undirected models amounts to $\frac{73.98\%}{2} = 36.49\%$ of the number of edges in the complete directed graphs. In fact, every edge-related operation (such as feature extraction) in the undirected model is actually performed on half of the number of edges compared to the directed model, saving run-time not only in the MST-inference stage but in every stage involving these operations. In addition, some pruning methods, such as length-dictionary pruning (used

[10] This hyperparameter was estimated once on our English development setup, and used for all 20 multilingual test setups.

[11] The directed edge weights for the local enhancement procedure (s_d in § 4) were computed using the trained D-MST parser.

[12] http://www.seas.upenn.edu/~strctlrn/MSTParser/MSTParser.html

	Swedish	Danish	Bulgarian	Slovene	Chinese	Hungarian	Turkish	German	Czech	Dutch
D-MST	87.7/88.9	88.5/89.5	90.4/90.9	80.4/83.4	86.1/87.7	82.9/84.3	75.2/75.3	89.6/90.2	81.7/84.0	81.3/83.0
U-MST-uf-lep	86.9/88.4	87.7/88.9	89.7/90.6	79.4/82.8	84.8/86.7	81.8/83.3	74.9/75.3	88.7/89.5	79.6/82.5	78.7/80.7
U-MST-uf	84.3/87.8	85.1/89.0	87.0/90.2	76.1/82.4	81.1/86.4	79.9/82.9	73.1/75.0	86.9/89.0	76.1/81.9	73.4/80.5
U-MST-df	72.0/79.2	74.3/82.9	69.5/81.4	66.8/75.8	65.9/76.5	68.2/72.1	57.4/62.6	77.7/82.5	57.3/70.9	59.0/71.3
	Japanese	Spanish	Catalan	Greek	Basque	Portuguese	Italian	PTB	QBank	GENIA
D-MST	92.5/92.6	83.8/86.0	91.8/92.2	82.7/84.9	72.1/75.8	89.2/89.9	83.4/85.4	92.1/92.8	95.8/96.3	88.9/90.0
U-MST-uf-lep	92.1/92.2	83.5/85.9	91.3/91.9	81.8/84.4	71.6/75.8	88.3/89.3	82.4/84.7	90.6/91.7	95.6/96.2	87.2/88.9
U-MST-uf	91.4/92.4	80.4/85.4	89.7/91.7	78.7/84	68.8/75.4	85.8/89.3	79.4/84.4	88.5/91.8	94.8/96.0	85.0/89.0
U-MST-df	74.4/85.2	73.1/81.3	73.1/83.5	71.3/78.7	62.8/71.4	67.9/79.7	65.2/77.2	77.2/85.4	89.1/92.9	72.4/81.6

Table 1: Directed/undirected UAS for the various parsing models of this paper.

	Swedish	Danish	Bulgarian	Slovene	Chinese	Hungarian	Turkish	German	Czech	Dutch
D-MST	20.6	20.8	15.1	25.4	15.5	26.4	22.3	21.3	29.7	27.7
U-MST-uf-lep	18.0	24.5	22.1	29.6	16.7	27.2	19.3	17.9	26.2	24.4
Oracle	88.9 (+1.2)	89.7 (+1.4)	91.6 (+1.2)	81.9 (+1.5)	87.8 (+1.7)	83.9 (+1)	77.1 (+1.9)	90.6 (+1)	82.8 (+1.1)	82.8 (+1.5)
	Japanese	Spanish	Catalan	Greek	Basque	Portuguese	Italian	PTB	QBank	GENIA
D-MST	5.7	26.7	23.4	28.9	23.4	22.6	22.5	27.8	5.3	33.7
U-MST-uf-lep	4.0	30.1	26.3	30.5	30.8	21.9	24.9	20.9	6.0	23.8
Oracle	93.1 (+0.6)	84.8 (+1)	92.6 (+0.8)	83.9 (+1.2)	74.1 (+2)	89.9 (+0.7)	84.4 (+1)	92.8 (+0.7)	96.4 (+0.8)	89.7 (+0.8)

Table 2: Top two lines (per language): percentage of sentences for which each of the models performs better than the other according to the directed UAS. Bottom line (Oracle): Directed UAS of an oracle model that selects the parse tree of the best performing model for each sentence. Improvement over the directed UAS score of D-MST is given in parenthesis.

in this work) perform feature extraction only for existing (un-pruned) edges, meaning that any reduction in the number of edges also reduces feature extraction operations.

For each model we report the standard directed unlabeled attachment accuracy score (D-UAS). In addition, since this paper explores the value of undirected inference for a problem that is directed in nature, we also report the undirected unlabeled attachment accuracy score (U-UAS), hoping that these results will shed light on the differences between the trees generated by the different models.

Results Table 1 presents our main results. While the directed MST parser (D-MST) is the best performing model across almost all test sets and evaluation measures, it outperforms our best model, U-MST-uf-lep, by a very small margin.

Particularly, for D-UAS, D-MST outperforms U-MST-uf-lep by up to 1% in 14 out of 20 setups (in 6 setups the difference is up to 0.5%). In 5 other setups the difference between the models is between 1% and 2%, and only in one setup it is above 2% (2.6%). Similarly, for U-UAS, in 2 setups the models achieve the same performance, in 15 setups the difference is less than 1% and in the other setups the differences is 1.1% - 1.5%. The average differences are 0.97% and 0.67% for D-UAS and U-UAS respectively.

The table further demonstrates the value of the local enhancement procedure. Indeed, U-MST-uf-lep outperforms U-MST in all 20 setups in D-UAS evaluation and in 15 out of 20 setups in U-UAS evaluation (in one setup there is a tie). However, the improvement this procedure provides is much more noticeable for D-UAS, with an averaged improvement of 2.35% across setups, compared to an averaged U-UAS improvement of only 0.26% across setups. While half of the changes performed by the local enhancement procedure are in edge directions, its marginal U-UAS improvement indicates that almost all of its power comes from edge direction changes. This calls for an improved enhancement procedure.

Finally, moving to directed features (the U-MST-df model), both D-UAS and U-UAS substantially degrade, with more noticeable degradation in the former. We hypothesize that this stems from the idiosyncrasy between the directed parameter update and the undirected inference in this model.

Table 2 reveals the complementary nature of our U-MST-uf-lep model and the classical D-MST: each of the models outperforms the other on an average of 22.2% of the sentences across test setups. An oracle model that selects the parse tree of the best model for each sentence would improve D-UAS by an average of 1.2% over D-MST across the test setups.

The results demonstrate the power of first-order graph-based dependency parsing with undirected inference. Although using a substantially different inference algorithm, our U-MST-uf-lep model performs very similarly to the standard MST parser which employs directed MST inference.

6 Discussion

We present a first-order graph-based dependency parsing model which runs in edge linear time at expectation and with very high probability. In extensive multilingual experiments our model performs very similarly to a standard directed first-order parser. Moreover, our results demonstrate the complementary nature of the models, with our model outperforming its directed counterpart on an average of 22.2% of the test sentences.

Beyond its practical implications, our work provides a novel intellectual contribution in demonstrating the power of undirected graph based methods in solving an NLP problem that is directed in nature. We believe this contribution has the potential to affect future research on additional NLP problems.

The potential embodied in this work extends to a number of promising research directions:

- Our algorithm may be used for efficient MST computation in k-best trees methods which are instrumental in margin-based training algorithms. For example, McDonald et al. (2005b) observed that k calls to the CLU algorithm might prove to be too inefficient; our more efficient algorithm may provide the remedy.

- It may also be utilized as an inference/initialization subroutine as a part of more complex approximation frameworks such as belief propagation (e.g. Smith and Eisner (2008), Gormley et al. (2015)).

- Finally, the complementary nature of the directed and undirected parsers motivates the development of methods for their combination, such as dual decomposition (e.g. Rush et al. (2010), Koo et al. (2010a)). Particularly, we have shown that our undirected inference algorithm converges to a different solution than the standard directed solution while still maintaining high quality (Table 2). Such techniques can exploit this diversity to produce a higher quality unified solution.

We intend to investigate all of these directions in future work. In addition, we are currently exploring potential extensions of the techniques presented in this paper to higher order, projective and non-projective, dependency parsing.

Acknowledgments

The second author was partly supported by a GIF Young Scientists' Program grant No. I-2388-407.6/2015 - Syntactic Parsing in Context.

References

Otakar Boruvka. 1926. O Jistém Problému Minimálním (About a Certain Minimal Problem) (in Czech, German summary). *Práce Mor. Prírodoved. Spol. v Brne III*, 3.

Sabine Buchholz and Erwin Marsi. 2006. Conll-x shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of CoNLL*.

Bernard Chazelle. 2000. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14.

Brandon Dixon, Monika Rauch, Robert, and Robert E. Tarjan. 1992. Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM J. Comput*, 21:1184–1192.

J. Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B:233–240.

Jason Eisner and Noah Smith. 2005. Parsing with soft and hard constraints on dependency length. In *Proc. IWPT*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies*. Kluwer Academic Publishers.

Harold N Gabow, Zvi Galil, Thomas Spencer, and Robert E Tarjan. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. *Combinatorica*, 6(2):109–122.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2012. Dependency parsing with undirected graphs. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 66–76. Association for Computational Linguistics.

Carlos Gómez-Rodríguez, Daniel Fernández-González, and Víctor Manuel Darriba Bilbao. 2015. Undirected dependency parsing. *Computational Intelligence*, 31(2):348–384.

Matthew Gormley, Mark Dredze, and Jason Eisner. 2015. Approximation-aware dependency parsing by belief propagation. *Transactions of the Association for Computational Linguistics*, 3:489–501.

Michiel Hazewinkel. 2001. Ackermann function. In *Encyclopedia of Mathematics*. Springer.

John Judge, Aoife Cahill, and Josef Van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of ACL-COLING*, pages 497–504.

David Karger, Philip Klein, and Robert Tarjan. 1995. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM*, 42(2):321–328.

Valerie King. 1995. A simpler minimum spanning tree verification algorithm. *Algorithmica*, 18:263–270.

Terry Koo and Michael Collins. 2010b. Efficient third-order dependency parsers. In *Proc. of ACL*.

T. Koo, A. M. Rush, M. Collins, T. Jaakkola, and D. Sontag. 2010a. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

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

A.F.T. Martins, N.A. Smith, and E.P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL*.

A.F.T. Martins, N.A. Smith, P.M.Q. Aguiar, and M.A.T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

A.F.T. Martins, Miguel Almeida, and N.A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proc. of EACL*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

Ryan McDonald, Koby Crammer, and Giorgio Satta. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

Jaroslav Nesetril, Eva Milková, and Helena Nesetrilová. 2001. Otakar boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history. *Discrete Mathematics*, 233(1-3):3–36.

Jens Nilsson, Sebastian Riedel, and Deniz Yuret. 2007. The conll 2007 shared task on dependency parsing. In *Proceedings of the CoNLL shared task session of EMNLP-CoNLL*, pages 915–932. sn.

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

Seth Pettie and Vijaya Ramachandran. 2002. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34.

Sebastian Riedel, David Smith, and Andrew McCallum. 2012. Parse, price and cut – delayed column and row generation for graph based parsers. In *Proc. of EMNLP-CoNLL 2012*.

Alexander Rush and Slav Petrov. 2012. Vine pruning for efficient multi-pass dependency parsing. In *Proc. of NAACL*.

Alexander M. Rush, David Sontag, Michael Collins, and Tommi Jaakkola. 2010. On dual decomposition and linear programming relaxations for natural language processing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Robert Endre Tarjan. 1977. Finding optimum branchings. *Networks*, 7(1):25–35.

Robert Endre Tarjan. 1983. *Data Structures and Network Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

Hao Zhang and Ryan McDonald. 2012. Generalized higher-order dependency parsing with cube pruning,. In *Proc. of EMNLP-CoNLL 2012*.

Topic Extraction from Microblog Posts Using Conversation Structures

Jing Li[1,2]*, **Ming Liao**[1,2], **Wei Gao**[3], **Yulan He**[4] and **Kam-Fai Wong**[1,2]
[1]The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
[2]MoE Key Laboratory of High Confidence Software Technologies, China
[3]Qatar Computing Research Institute, Hamad Bin Khalifa University, Doha, Qatar
[4]School of Engineering and Applied Science, Aston University, UK
{lijing,mliao,kfwong}@se.cuhk.edu.hk[1,2]
wgao@qf.org.qa[3], y.he9@aston.ac.uk[4]

Abstract

Conventional topic models are ineffective for topic extraction from microblog messages since the lack of structure and context among the posts renders poor message-level word co-occurrence patterns. In this work, we organize microblog posts as conversation trees based on reposting and replying relations, which enrich context information to alleviate data sparseness. Our model generates words according to topic dependencies derived from the conversation structures. In specific, we differentiate messages as leader messages, which initiate key aspects of previously focused topics or shift the focus to different topics, and follower messages that do not introduce any new information but simply echo topics from the messages that they repost or reply. Our model captures the different extents that leader and follower messages may contain the key topical words, thus further enhances the quality of the induced topics. The results of thorough experiments demonstrate the effectiveness of our proposed model.

1 Introduction

The increasing popularity of microblog platforms results in a huge volume of user-generated short posts. Automatically modeling topics out of such massive microblog posts can uncover the hidden semantic structures of the underlying collection and can be useful to downstream applications such as microblog summarization (Harabagiu and Hickl, 2011), user profiling (Weng et al., 2010), event tracking (Lin et al., 2010) and so on.

Popular topic models, like Probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 1999) and Latent Dirichlet Allocation (LDA) (Blei et al., 2003b), model the semantic relationships between words based on their co-occurrences in documents. They have demonstrated their success in conventional documents such as news reports and scientific articles, but perform poorly when directly applied to short and colloquial microblog content due to severe sparsity in microblog messages (Wang and McCallum, 2006; Hong and Davison, 2010).

A common way to deal with short text sparsity is to aggregate short messages into long pseudo-documents. Most of the studies heuristically aggregate messages based on authorship (Zhao et al., 2011; Hong and Davison, 2010), shared words (Weng et al., 2010), or hashtags (Ramage et al., 2010; Mehrotra et al., 2013). Some works directly take into account the word relations to alleviate document-level word sparseness (Yan et al., 2013; Sridhar, 2015). More recently, a self-aggregation-based topic model called SATM (Quan et al., 2015) was proposed to aggregate texts jointly with topic inference.

However, we argue that the existing aggregation strategies are suboptimal for modeling topics in short texts. Microblogs allow users to share and comment on messages with friends through reposting or replying, similar to our everyday conversations. Intuitively, the conversation structures can not only enrich context, but also provide useful clues for identifying relevant topics. This is nonetheless ignored in previous approaches. Moreover, the occurrence of non-topic words such as emotional, sentimental, functional and even meaningless words are very common in microblog posts, which may distract the models from recognizing topic-related key words and thus fail to produce coherent and meaningful topics.

We propose a novel topic model by utilizing the structures of conversations in microblogs. We link microblog posts using reposting and replying rela-

* Part of this work was conducted when the first author was visiting Aston University.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2114–2123,
Berlin, Germany, August 7-12, 2016.

tions to build conversation trees. Particularly, the root of a conversation tree refers to the original post and its edges represent the reposting/replying relations.

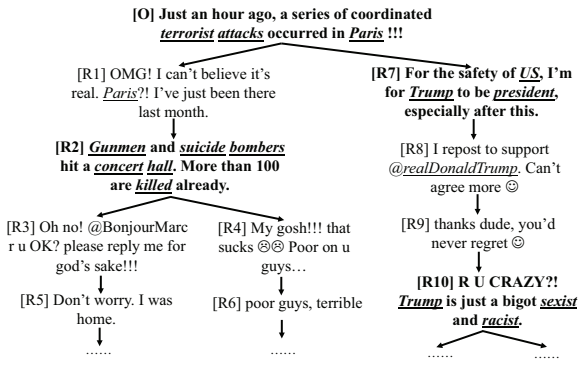


Figure 1: An example of conversation tree. [O]: the original post; [Ri]: the i-th repost/reply; Arrow lines: reposting/replying relations; Dark black posts: leaders to be detected; Underlined italic words: key words representing topics

Figure 1 illustrates an example of a conversation tree, in which messages can initiate a new topic such as [O] and [R7] or raise a new aspect (subtopic) of the previously discussed topics such as [R2] and [R10]. These messages are named as **leaders**, which contain salient content in topic description, e.g., the italic and underlined words in Figure 1. The remaining messages, named as **followers**, do not raise new issues but simply respond to their reposted or replied messages following what has been raised by the leaders and often contain non-topic words, e.g., *OMG*, *OK*, *agree*, etc.

Conversation tree structures from microblogs have been previously shown helpful to microblog summarization (Li et al., 2015), but have never been explored for topic modeling. We follows Li et al. (2015) to detect leaders and followers across paths of conversation trees using Conditional Random Fields (CRF) trained on annotated data. The detected leader/follower information is then incorporated as prior knowledge into our proposed topic model.

Our experimental results show that our model, which captures parent-child topic correlations in conversation trees and generates topics by considering messages being leaders or followers separately, is able to induce high-quality topics and outperforms a number of competitive baselines. In summary, our contributions are three-fold:

- We propose a novel topic model, which explicitly exploits the topic dependencies contained in conversation structures to enhance topic assignments.
- Our model differentiates the generative process of topical and non-topic words, according to the message where a word is drawn from being a leader or a follower. This helps the model distinguish the topic-specific information from background noise.
- Our model outperforms state-of-the-art topic models when evaluated on a large real-world microblog dataset containing over 60K conversation trees, which is publicly available[1].

2 Related Works

Topic models aim to discover the latent semantic information, i.e., topics, from texts and have been extensively studied. One of the most popular and well-known topic models is LDA (Blei et al., 2003b). It utilizes Dirichlet priors to generate document-topic and topic-word distributions, and has been shown effective in extracting topics from conventional documents.

Nevertheless, prior research has demonstrated that standard topic models, essentially focusing on document-level word co-occurrences, are not suitable for short and informal microblog messages due to severe data sparsity exhibited in short texts (Wang and McCallum, 2006; Hong and Davison, 2010). Therefore, how to enrich and exploit context information becomes a main concern. Weng et al. (2010), Hong et al. (2010) and Zhao et al. (2011) first heuristically aggregated messages posted by the same user or sharing the same words before applying classic topic models to extract topics. However, such a simple strategy poses some problems. For example, it is common that a user has various interests and posts messages covering a wide range of topics. Ramage et al. (2010) and Mehrotra et al. (2013) used hashtags as labels to train supervised topic models. But these models depend on large-scale hashtag-labeled data for model training, and their performance is inevitably compromised when facing unseen topics irrelevant to any hashtag in training data due to the rapid change and wide variety of topics in social media.

SATM (Quan et al., 2015) combined short texts aggregation and topic induction into a unified model. But in their work, no prior knowledge

[1] http://www1.se.cuhk.edu.hk/~lijing/data/microblog-topic-extraction-data.zip

was given to ensure the quality of text aggregation, which therefore can affect the performance of topic inference. In this work, we organize microblog messages as conversation trees based on reposting/reply relations, which is a more advantageous message aggregation strategy.

Another line of research tackled the word sparseness by modeling word relations instead of word occurrences in documents. For example, Gaussian Mixture Topic Model (GMTM) (Sridhar, 2015) utilized word embeddings to model the distributional similarities of words and then inferred clusters of words represented by word distributions using Gaussian Mixture Model (GMM) that capture the notion of latent topics. However, GMTM heavily relies on meaningful word embeddings that require a large volume of high-quality external resources for training.

Biterm Topic Model (BTM) (Yan et al., 2013) directly explores unordered word-pair co-occurrence patterns in each individual message. Our model learns topics from aggregated messages based on conversation trees, which naturally provide richer context since word co-occurrence patterns can be captured from multiple relevant messages.

3 LeadLDA Topic Model

In this section, we describe how to extract topics from a microblog collection utilizing conversation tree structures, where the trees are organized based on reposting and replying relations among the messages[2].

To identify key topic-related content from colloquial texts, we differentiate the messages as *leaders* and *followers*. Following Li et al. (2015), we extract all root-to-leaf paths on conversation trees and utilize the state-of-the-art sequence learning model CRF (Lafferty et al., 2001) to detect the leaders[3]. As a result, the posterior probability of each node being a leader or follower is obtained by averaging the different marginal probabilities of the same node over all the tree paths that contain the node. Then, the obtained probability distribution is considered as the observed prior variable input into our model.

[2]Reposting/replying relations are straightforward to obtain by using microblog APIs from Twitter and Sina Weibo.

[3]The CRF model for leader detection was trained on a public corpus with all the messages annotated on the tree paths. Details are described in Section 4.

3.1 Topics and Conversation Trees

Previous works (Zhao et al., 2011; Yan et al., 2013; Quan et al., 2015) have proven that assuming each short post contains a single topic is useful to alleviate the data sparsity problem. Thus, given a corpus of microblog posts organized as conversation trees and the estimated leader probabilities of tree nodes, we assume that each message only contains a single topic and a tree covers a mixture of multiple topics. Since leader messages subsume the content of their followers, the topic of a leader can be generated from the topic distribution of the entire tree. Consequently, the topic mixture of a conversation tree is determined by the topic assignments to the leader messages on it. The topics of followers, however, exhibit strong and explicit dependencies on the topics of their ancestors. So, their topics need to be generated in consideration of local constraints. Here, we mainly address how to model the topic dependencies of followers.

Enlighten by the general Structural Topic Model (strTM) (Wang et al., 2011), which incorporates document structures into topic model by explicitly modeling topic dependencies between adjacent sentences, we exploit the topical transitions between parents and children in the trees for guiding topic assignments.

Intuitively, the emergence of a leader results in potential topic shift. It tends to weaken the topic similarities between the emerging leaders and their predecessors. For example, [R7] in Figure 1 transfers the topic to a new focus, thus weakens the tie with its parent. We can simplify our case by assuming that followers are topically responsive just up to (hence not further than) their nearest ancestor leaders. Thus, we can dismantle each conversation tree into forest by removing the links between leaders and their parents hence producing a set of subgraphs like [R2]–[R6] and [R7]–[R9] in Figure 1. Then, we model the internal topic dependencies within each subgraph by inferring the parent-child topic transition probabilities satisfying the first-order Markov properties in a similar way as estimating the transition distribution of adjacent sentences in strTM (Wang et al., 2011). At topic assignment stage, the topic of a follower will be assigned by referring to its parent's topic and the transition distribution that captures topic similarities of followers to their parents (see Section 3.2).

In addition, every word in the corpus is either

a topical or non-topic (i.e., background) word, which highly depends on whether it occurs in a leader or a follower message.

Figure 2 illustrates the graphical model of our generative process, which is named as LeadLDA.

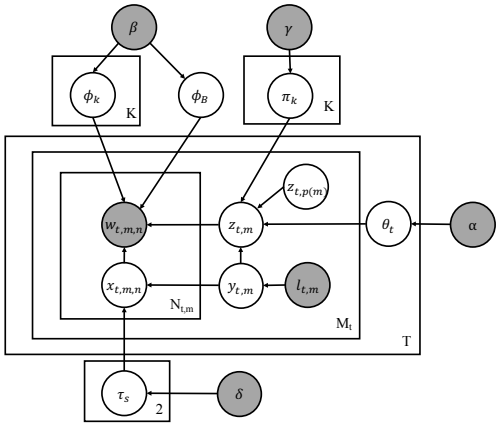


Figure 2: Graphical Model of LeadLDA

3.2 Topic Modeling

Formally, we assume that the microblog posts are organized as T conversation trees. Each tree t contains M_t message nodes and each message m contains $N_{t,m}$ words in the vocabulary. The vocabulary size is V and there are K topics embedded in the corpus represented by word distribution $\phi_k \sim Dir(\beta)$ $(k = 1, 2, ..., K)$. Also, a background word distribution $\phi_B \sim Dir(\beta)$ is included to capture the general information, which is not topic specific. ϕ_k and ϕ_B are multinomial distributions over the vocabulary. A tree t is modeled as a mixture of topics $\theta_t \sim Dir(\alpha)$ and any message m on t is assumed to contain a single topic $z_{t,m} \in \{1, 2, ..., K\}$.

(1) Topic assignments: The topic assignments of LeadLDA is inspired by Griffiths et al. (2004) that combines syntactic and semantic dependencies between words. LeadLDA integrates the outcomes of leader detection with a binomial switcher $y_{t,m} \in \{0, 1\}$ indicating whether m is a leader $(y_{t,m} = 1)$ or a follower $(y_{t,m} = 0)$, given each message m on the tree t. $y_{t,m}$ is parameterized by its leader probability $l_{t,m}$, which is the posterior probability output from the leader detection model and serves as an observed prior variable.

According to the notion of leaders, they initiate key aspects of previously discussed topics or signal a new topic shifting the focus of its descendant followers. So, the topics of leaders on tree t are directly sampled from the topic mixture θ_t.

To model the internal topic correlations within the subgraph of conversation tree consisting of a leader and all its followers, we capture parent-child topic transitions $\pi_k \sim Dir(\gamma)$, which is a distribution over K topics, and use $\pi_{k,j}$ to denote the probability of a follower assigned topic j when the topic of its parent is k. Specifically, if message m is sampled as a follower and the topic assignment to its parent message is $z_{t,p(m)}$, where $p(m)$ indexes the parent of m, then $z_{t,m}$ (i.e., the topic of m) is generated from topic transition distribution $\pi_{z_{t,p(m)}}$. In particular, since the root of a conversation tree has no parent and can only be a leader, we make the leader probability $l_{t,root} = 1$ to force its topic only to be generated from the topic distribution of tree t.

(2) Topical and non-topic words: We separately model the distributions of leader and follower messages emitting topical or non-topic words with τ_0 and τ_1, respectively, both of which are drawn from a symmetric Beta prior parametererized by δ. Specifically, for each word n in message m on tree t, we add a binomial background switcher $x_{t,m,n}$ controlled by whether m is a leader or a follower, i.e., $x_{t,m,n} \sim Bi(\tau_{y_{t,m}})$, which indicates n is a topical word if $x_{t,m,n} = 0$ or a background word if $x_{t,m,n} = 1$, and $x_{t,m,n}$ controls n to be generated from the topic-word distribution $\phi_{z_{t,m}}$, where $z_{t,m}$ is the topic of m, or from background word distribution ϕ_B modeling non-topic information.

(3) Generation process: To sum up, conditioned on the hyper-parameters $\Theta = (\alpha, \beta, \gamma, \delta)$, the generation process of a conversation tree t can be described as follows:

- Draw $\theta_t \sim Dir(\alpha)$
- For message $m = 1$ to M_t on tree t
  - Draw $y_{t,m} \sim Bi(l_{t,m})$
  - If $y_{t,m} == 1$
    - Draw $z_{t,m} \sim Mult(\theta_t)$
  - If $y_{t,m} == 0$
    - Draw $z_{t,m} \sim Mult(\pi_{z_{t,p(m)}})$
  - For word $n = 1$ to $N_{t,m}$ in m
    - Draw $x_{t,m,n} \sim Bi(\tau_{y_{t,m}})$
    - If $x_{t,m,n} == 0$
      - Draw $w_{t,m,n} \sim Mult(\phi_{z_{t,m}})$
    - If $x_{t,m,n} == 1$
      - Draw $w_{t,m,n} \sim Mult(\phi_B)$

$C^{LB}_{s,(r)}$	# of words with background switchers assigned as r and occurring in messages with leader switchers s.
$C^{LB}_{s,(\cdot)}$	# of words occurring in messages whose leader switchers are s, i.e., $\sum_{r\in\{0,1\}} C^{LB}_{s,(r)}$.
$N^{B}_{(r)}$	# of words occurring in message (t,m) and with background switchers assigned as r.
$N^{B}_{(\cdot)}$	# of words in message (t,m), i.e., $N^{B}_{(\cdot)} = \sum_{r\in\{0,1\}} N^{B}_{(r)}$.
$C^{TW}_{k,(v)}$	# of words indexing v in vocabulary, sampled as topic (non-background) words, and occurring in messages assigned topic k.
$C^{TW}_{k,(\cdot)}$	# of words assigned as topic (non-background) word and occurring in messages assigned topics k, i.e., $C^{TW}_{k,(\cdot)} = \sum_{v=1}^{V} C^{TW}_{k,(v)}$.
$N^{W}_{(v)}$	# of words indexing v in vocabulary that occur in message (t,m) and are assigned as topic (non-background) word.
$N^{W}_{(\cdot)}$	# of words assigned as topic (non-background) words and occurring in message (t,m), i.e., $N^{W}_{(\cdot)} = \sum_{v=1}^{V} N^{W}_{(v)}$.
$C^{TR}_{i,(j)}$	# of messages sampled as followers and assigned topic j, whose parents are assigned topic i.
$C^{TR}_{i,(\cdot)}$	# of messages sampled as followers whose parents are assigned topic i, i.e., $C^{TR}_{i,(\cdot)} = \sum_{j=1}^{K} C^{TR}_{i,(j)}$.
$I(\cdot)$	An indicator function, whose value is 1 when its argument inside () is true, and 0 otherwise.
$N^{CT}_{(j)}$	# of messages that are children of message (t,m), sampled as followers and assigned topic j.
$N^{CT}_{(\cdot)}$	# of message (t,m)'s children sampled as followers, i.e., $N^{CT}_{(\cdot)} = \sum_{j=1}^{K} N^{CT}_{(j)}$
$C^{TT}_{t,(k)}$	# of messages on conversation tree t sampled as leaders and assigned topic k.
$C^{TT}_{t,(\cdot)}$	# of messages on conversation tree t sampled as leaders, i.e., $C^{TT}_{t,(\cdot)} = \sum_{k=1}^{K} C^{TT}_{t,(k)}$
$C^{BW}_{(v)}$	# of words indexing v in vocabulary and assigned as background (non-topic) words
$C^{BW}_{(\cdot)}$	# of words assigned as background (non-topic) words, i.e., $C^{BW}_{(\cdot)} = \sum_{v=1}^{V} C^{BW}_{(v)}$

Table 1: The notations of symbols in the sampling formulas (1) and (2). (t,m): message m on conversation tree t.

3.3 Inference for Parameters

We use collapsed Gibbs Sampling (Griffiths, 2002) to carry out posterior inference for parameter learning. The hidden multinomial variables, i.e., message-level variables (y and z) and word-level variables (x) are sampled in turn, conditioned on a complete assignment of all other hidden variables. Due to the space limitation, we leave out the details of derivation but give the core formulas in the sampling steps.

We first define the notations of all variables needed by the formulation of Gibbs sampling, which are described in Table 1. In particular, the various C variables refer to counts excluding the message m on conversation tree t.

For each message m on a tree t, we sample the leader switcher $y_{t,m}$ and topic assignment $z_{t,m}$ according to the following conditional probability distribution:

$$\begin{aligned} &p(y_{t,m}=s, z_{t,m}=k|\mathbf{y}_{\neg(t,m)}, \mathbf{z}_{\neg(t,m)}, \mathbf{w}, \mathbf{x}, \mathbf{l}, \Theta) \\ \propto &\frac{\Gamma(C^{LB}_{s,(\cdot)}+2\delta)}{\Gamma(C^{LB}_{s,(\cdot)}+N^{B}_{(\cdot)}+2\delta)} \prod_{r\in\{0,1\}} \frac{\Gamma(C^{LB}_{s,(r)}+N^{B}_{(r)}+\delta)}{\Gamma(C^{LB}_{s,(r)}+\delta)} \\ \cdot &\frac{\Gamma(C^{TW}_{k,(\cdot)}+V\beta)}{\Gamma(C^{TW}_{k,(\cdot)}+N^{W}_{(\cdot)}+V\beta)} \prod_{v=1}^{V} \frac{\Gamma(C^{TW}_{k,(v)}+N^{W}_{(v)}+\beta)}{\Gamma(C^{TW}_{k,(v)}+\beta)} \\ \cdot &g(s,k,t,m) \end{aligned} \tag{1}$$

where $g(s,k,t,m)$ takes different forms depending on the value of s:

$$\begin{aligned} g(0,k,t,m) = &\frac{\Gamma(C^{TR}_{z_{t,p(m)},(\cdot)}+K\gamma)}{\Gamma(C^{TR}_{z_{t,p(m)},(\cdot)}+I(z_{t,p(m)}\neq k)+K\gamma)} \\ \cdot &\frac{\Gamma(C^{TR}_{k,(\cdot)}+K\gamma)}{\Gamma(C^{TR}_{k,(\cdot)}+I(z_{t,p(m)}=k)+N^{CT}_{(\cdot)}+K\gamma)} \\ \cdot &\prod_{j=1}^{K} \frac{\Gamma(C^{TR}_{k,(j)}+N^{CT}_{(j)}+I(z_{t,p(m)}=j=k)+\gamma)}{\Gamma(C^{TR}_{k,(j)}+\gamma)} \\ \cdot &\frac{\Gamma(C^{TR}_{z_{t,p(m)},(k)}+I(z_{t,p(m)}\neq k)+\gamma)}{\Gamma(C^{TR}_{z_{t,p(m)},(k)}+\gamma)} \cdot (1-l_{t,m}) \end{aligned}$$

and

$$g(1,k,t,m) = \frac{C^{TT}_{t,(k)}+\alpha}{C^{TT}_{t,(\cdot)}+K\alpha} \cdot l_{t,m}$$

For each word n in m on t, the sampling formula of its background switcher is given as the following:

$$\begin{aligned} &p(x_{t,m,n}=r|\mathbf{x}_{\neg(t,m,n)}, \mathbf{y}, \mathbf{z}, \mathbf{w}, \mathbf{l}, \Theta) \\ \propto &\frac{C^{LB}_{y_{t,m},(r)}+\delta}{C^{LB}_{y_{t,m},(\cdot)}+2\delta} \cdot h(r,t,m,n) \end{aligned} \tag{2}$$

where

$$h(r,t,m,n) = \begin{cases} \frac{C^{TW}_{z_{t,m},(w_{t,m,n})}+\beta}{C^{TW}_{z_{t,m},(\cdot)}+V\beta} & \text{if } r=0 \\ \frac{C^{BW}_{(w_{t,m,n})}+\beta}{C^{BW}_{(\cdot)}+V\beta} & \text{if } r=1 \end{cases}$$

4 Data Collection and Experiment Setup

To evaluate our LeadLDA model, we conducted experiments on real-world microblog dataset collected from Sina Weibo that has the same 140-character limitation and shares the similar market penetration as Twitter (Rapoza, 2011). For the hyper-parameters of LeadLDA, we fixed $\alpha = 50/K$, $\beta = 0.1$, following the common practice in previous works (Griffiths and Steyvers, 2004; Quan et al., 2015). Since there is no analogue of γ and δ in prior works, where γ controls topic dependencies of follower messages to their ancestors and δ controls the different tendencies of

Month	# of trees	# of messages	Vocab size
May	10,812	38,926	6,011
June	29,547	98,001	9,539
July	26,103	102,670	10,121

Table 2: Statistics of our three evaluation datasets

leaders and followers covering topical and non-topic words. We tuned γ and δ by grid search on a large development set containing around 120K posts and obtained $\gamma = 50/K$, $\delta = 0.5$.

Because the content of posts are often incomplete and informal, it is difficult to manually annotate topics in a large scale. Therefore, we follow Yan et al. (2013) to utilize hashtags led by '#', which are manual topic labels provided by users, as ground-truth categories of microblog messages. We collected the real-time trending hashtags on Sina Weibo and utilized the hashtag-search API[4] to crawl the posts matching the given hashtag queries. In the end, we built a corpus containing 596,318 posts during May 1 – July 31, 2014.

To examine the performance of models on various topic distributions, we split the corpus into 3 datasets, each containing messages of one month. Similar to Yan et al. (2013), for each dataset, we manually selected 50 frequent hashtags as topics, e.g. #mh17, #worldcup, etc. The experiments were conducted on the subsets of posts with the selected hashtags. Table 2 shows the statistics of the three subsets used in our experiments.

We preprocessed the datasets before topic extraction in the following steps: 1) Use FudanNLP toolkit (Qiu et al., 2013) for word segmentation, stop words removal and POS tagging for Chinese Weibo messages; 2) Generate a vocabulary for each dataset and remove words occurring less than 5 times; 3) Remove all hashtags in texts before input them to models, since the models are expected to extract topics without knowing the hashtags, which are ground-truth topics; 4) For LeadLDA, we use the CRF-based leader detection model (Li et al., 2015) to classify messages as leaders and followers. The leader detection model was implemented by using CRF++[5], which was trained on the public dataset composed of 1,300 conversation paths and achieved state-of-the-art 73.7% F1-score of classification accuracy (Li et al., 2015).

[4] http://open.weibo.com/wiki/2/search/topics

[5] https://taku910.github.io/crfpp/

5 Experimental Results

We evaluated topic models with two sets of K, i.e., the number of topics. One is $K = 50$, to match the count of hashtags following Yan et al. (2013), and the other is $K = 100$, much larger than the "real" number of topics. We compared LeadLDA with the following 5 state-of-the-art basedlines.

TreeLDA: Analogous to Zhao et al. (2011), where they aggregated messages posted by the same author, TreeLDA aggregates messages from one conversation tree as a pseudo-document. Additionally, it includes a background word distribution to capture non-topic words controlled by a general Beta prior without differentiating leaders and followers. TreeLDA can be considered as a degeneration of LeadLDA, where topics assigned to all messages are generated from the topic distributions of the conversation trees they are on.

StructLDA: It is another variant of LeadLDA, where topics assigned to all messages are generated based on topic transitions from their parents. The strTM (Wang et al., 2011) utilized a similar model to capture the topic dependencies of adjacent sentences in a document. Following strTM, we add a dummy topic T_{start} emitting no word to the "pseudo parents" of root messages. Also, we add the same background word distribution to capture non-topic words as TreeLDA does.

BTM: Biterm Topic Model (BTM)[6] (Yan et al., 2013) directly models topics of all word pairs (biterms) in each post, which outperformed LDA, Mixture of Unigrams model, and the model proposed by Zhao et al. (2011) that aggregated posts by authorship to enrich context.

SATM: A general unified model proposed by Quan et al. (2015) that aggregates documents and infers topics simultaneously. We implemented SATM and examined its effectiveness specifically on microblog data.

GMTM: To tackle word sparseness, Sridhar et al. (2015) utilized Gaussian Mixture Model (GMM) to cluster word embeddings generated by a log-linear word2vec model[7].

The hyper-parameters of BTM, SATM and GMTM were set according to the best hyper-parameters reported in their original papers. For TreeLDA and StructLDA, the parameter settings were kept the same as LeadLDA since they are its

[6] https://github.com/xiaohuiyan/BTM

[7] https://code.google.com/archive/p/word2vec/

variants. And the background switchers were parameterized by symmetric Beta prior on 0.5, following Chemudugunta et al. (2006). We ran Gibbs samplings (in LeadLDA, TreeLDA, StructLDA, BTM and SATM) and EM algorithm (in GMTM) with 1,000 iterations to ensure convergence.

Topic model evaluation is inherently difficult. In previous works, perplexity is a popular metric to evaluate the predictive abilities of topic models given held-out dataset with unseen words (Blei et al., 2003b). However, Chang et al. (2009) have demonstrated that models with high perplexity do not necessarily generate semantically coherent topics in human perception. Therefore, we conducted objective and subjective analysis on the coherence of produced topics.

5.1 Objective Analysis

The quality of topics is commonly measured by coherence scores (Mimno et al., 2011), assuming that words representing a coherent topic are likely to co-occur within the same document. However, due to the severe sparsity of short text posts, we modify the calculation of commonly-used topic coherence measure based on word co-occurrences in messages tagged with the same hashtag, named as hashtag-document, assuming that those messages discuss related topics[8].

Specifically, we calculate the coherence score of a topic given the top N words ranked by likelihood as below:

$$\mathcal{C} = \frac{1}{K} \cdot \sum_{k=1}^{K} \sum_{i=2}^{N} \sum_{j=1}^{i-1} log \frac{D(w_i^k, w_j^k) + 1}{D(w_j^k)}, \qquad (3)$$

where w_i^k represents the i-th word in topic k ranked by $p(w|k)$, $D(w_i^k, w_j^k)$ refers to the count of hashtag-documents where word w_i^k and w_j^k co-occur, and $D(w_i^k)$ denotes the number of hashtag-documents that contain word w_i^k.

Table 3 shows the absolute values of $\mathcal{C}$ scores for topics produced on three evaluation datasets (May, June and July), and the top 10, 15, 20 words of topics were selected for evaluation. Lower scores indicate better coherence in the induced topic.

We have the following observations:

• GMTM gave the worst coherence scores, which may be ascribed to its heavy reliance on relevant large-scale high-quality external data, without which the trained word embedding model failed to capture meaningful semantic features for words, and hence could not yield coherent topics.

• TreeLDA and StructLDA produced competitive results compared to the state-of-the-art baseline models, which indicates the effectiveness of using conversation structures to enrich context and thus generate topics of reasonably good quality.

• The coherence of topics generated by LeadLDA outperformed all the baselines on the three datasets, most of time by large margins and was only outperformed by BTM on the May dataset when $K = 50$ and $N = 10$. The generally higher performance of LeadLDA is due to three reasons: 1) It effectively identifies topics using the conversation tree structures, which provide richer context information; 2) It jointly models the topics of leaders and the topic dependencies of other messages on a tree. TreeLDA and StructLDA, each only considering one of the factors, performed worse than LeadLDA; 3) LeadLDA separately models the probabilities of leaders and followers containing topical or non-topic words while the baselines only model the general background information regardless of the different types of messages. This implies that leaders and followers do have different capacities in covering key topical words or background noise, which is useful to identify key words for topic representation.

N	Model	May		June		July	
		K50	K100	K50	K100	K50	K100
10	TREE	27.9	30.5	24.0	23.8	23.9	26.1
	STR	29.9	30.8	24.0	24.1	24.4	26.4
	BTM	**26.7**	28.9	27.8	25.5	25.4	25.2
	SATM	30.6	29.9	23.8	23.7	24.3	27.5
	GMTM	40.8	40.1	44.0	44.2	41.7	40.8
	LEAD	28.4	**26.9**	**19.8**	**23.4**	**22.6**	**25.1**
15	TREE	71.9	76.4	55.3	60.4	61.2	66.2
	STR	76.4	74.1	57.6	62.2	58.1	61.1
	BTM	69.6	71.4	58.5	60.3	59.1	63.0
	SATM	74.3	73.0	54.8	60.4	61.2	65.3
	GMTM	96.4	93.1	100.4	105.1	94.6	94.9
	LEAD	**67.4**	**65.2**	**52.8**	**57.7**	**55.3**	**57.8**
20	TREE	138.8	138.6	102.0	115.0	115.8	119.7
	STR	134.0	136.9	104.3	112.7	111.0	117.3
	BTM	125.2	131.1	109.4	115.7	115.3	120.2
	SATM	134.6	131.9	105.5	114.3	113.5	118.9
	GMTM	173.5	169.0	184.7	190.9	167.4	171.2
	LEAD	**120.9**	**127.2**	**101.6**	**106.0**	**97.2**	**104.9**

Table 3: Absolute values of coherence scores. Lower is better. K50: 50 topics; K100: 100 topics; N: # of top words ranked by topic-word probabilities; TREE: TreeLDA; STR: StructLDA; LEAD: LeadLDA.

[8]We sampled posts and their corresponding hashtags in our evaluation set and found only 1% mismatch.

TreeLDA	StructLDA	BTM	SATM	LeadLDA
香港 微博 马航 家属 证实 入境处 客机 消息 曹格 投给 二胎 选项 教父 滋养 飞机 外国 心情 坠毁 男子 同胞	乌克兰 航空 亲爱 国民 绕开 飞行 航班 领空 所有 避开 宣布 空域 东部 俄罗斯 终于 忘记 公司 绝望 看看 珍贵	香港 入境处 家属 证实 男子 护照 外国 消息 坠毁 马航 报道 联系 电台 客机 飞机 同胞 确认 事件 霍家 直接	马航 祈祷 安息 生命 逝者 世界 艾滋病 恐怖 广州 飞机 无辜 默哀 远离 事件 击落 公交车 中国人 国际 愿逝者 真的	乌克兰 马航 客机 击落 飞机 坠毁 导弹 俄罗斯 消息 乘客 中国 马来西亚 香港 遇难 事件 武装 航班 恐怖 目前 证实
Hong Kong, microblog, family, confirm, immigration, airliner, news, Grey Chow, vote, second baby, choice, god father, nourish, airplane, foreign, feeling, crash, man, fellowman	Ukraine, airline, dear, national, bypass, fly, flight, airspace, all, avoid, announce, airspace, eastern, Russia, finally, forget, company, disappointed, look, valuable	Hong Kong, immigration, family, confirm, man, passport, foreign, news, crash, Malaysia Airlines, report, contact, broadcast station, airliner, airplane, fellowman, confirm, event, Fok's family, directly	Malaysia Airlines, prey, rest in peace, life, dead, world, AIDS, terror, Guangzhou, airplane, innocent, silent tribute, keep away from, event, shoot down, bus, Chinese, international, wish the dead, really	Ukraine, Malaysia Airlines, airliner, shoot down, airplane, crash, missile, Russia, news, passenger, China, Malaysia, Hong Kong, killed, event, militant, flight, terror, current, confirm

Figure 3: The extracted topics describing MH17 crash. Each column represents the similar topic generated by the corresponding model with the top 20 words. The 2nd row: original Chinese words; The 3rd row: English translations.

5.2 Subjective Analysis

To evaluate the coherence of induced topics from human perspective, we invited two annotators to subjectively rate the quality of every topic (by displaying the top 20 words) generated by different models on a 1-5 Likert scale. A higher rating indicates better quality of topics. The Fless's Kappa of annotators' ratings measured for various models on different datasets given $K = 50$ and 100 range from 0.62 to 0.70, indicating substantial agreements (Landis and Koch, 1977).

Table 4 shows the overall subjective ratings. We noticed that humans preferred topics produced given $K = 100$ to $K = 50$, but coherence scores gave generally better grades to models for $K = 50$, which matched the number of topics in ground truth. This is because models more or less mixed more common words when K is larger. Coherence score calculation (Equation (3)) penalizes common words that occur in many documents, whereas humans could somehow "guess" the meaning of topics based on the rest of words thus gave relatively good ratings. Nevertheless, annotators gave remarkably higher ratings to LeadLDA than baselines on all datasets regardless of K being 50 or 100, which confirmed that LeadLDA effectively yielded good-quality topics.

For a detailed analysis, Figure 3 lists the top 20 words about "MH17 crash" induced by different models[9] when $K = 50$. We have the following observations:

Model	May		June		July	
	K50	K100	K50	K100	K50	K100
TREE	3.12	3.41	3.42	3.44	3.03	3.48
STR	3.05	3.45	3.38	3.48	3.08	3.53
BTM	3.04	3.26	3.40	3.37	3.15	3.57
SATM	3.08	3.43	3.30	3.55	3.09	3.54
GMTM	2.02	2.37	1.99	2.27	1.97	1.90
LEAD	**3.40**	**3.57**	**3.52**	**3.63**	**3.55**	**3.72**

Table 4: Subjective ratings of topics. The meanings of K50, K100, TREE, STR and LEAD are the same as in Table 3.

- BTM, based on word-pair co-occurrences, mistakenly grouped "Fok's family" (a tycoon family in Hong Kong), which co-occurred frequently with "Hong Kong" in other topics, into the topic of "MH17 crash". "Hong Kong" is relevant here as a Hong Kong passenger died in the MH17 crash.
- The topical words generated by SATM were mixed with words relevant to the bus explosion in Guangzhou, since it aggregated messages according to topic affinities based on the topics learned in the previous step. Thus the posts about bus explosion and MH17 crash, both pertaining to disasters, were aggregated together mistakenly, which generated spurious topic results.
- Both TreeLDA and StructLDA generated topics containing non-topic words like "microblog" and "dear". This means that without distinguishing leaders and followers, it is difficult to filter out non-topic words. The topic quality of StructLDA nevertheless seems better than

[9] As shown in Table 3 and 4, the topic coherence scores of GMTM were the worst. Hence, the topic generated by GMTM is not shown due to space limitation.

TreeLDA, which implies the usefulness of exploiting topic dependencies of posts in conversation structures.

• LeadLDA not only produced more semantically coherent words describing the topic, but also revealed some important details, e.g., MH17 was shot down by a missile.

6 Conclusion and Future Works

This paper has proposed a novel topic model by considering the conversation tree structures of microblog posts. By rigorously comparing our proposed model with a number of competitive baselines on real-world microblog datasets, we have demonstrated the effectiveness of using conversation structures to help model topics embedded in short and colloquial microblog messages.

This work has proven that detecting leaders and followers, which are coarse-grained discourse derived from conversation structures, is useful to model microblogging topics. In the next step, we plan to exploit fine-grained discourse structures, e.g., dialogue acts (Ritter et al., 2010), and propose a unified model that jointly inferring discourse roles and topics of posts in context of conversation tree structures. Another extension is to extract topic hierarchies by integrating the conversation structures into hierarchical topic models like HLDA (Blei et al., 2003a) to extract fine-grained topics from microblog posts.

Acknowledgment

This work is supported by General Research Fund of Hong Kong (417112), the Innovation and Technology Fund of Hong Kong SAR (ITP/004/16LP), Shenzhen Peacock Plan Research Grant (KQCX20140521144507925) and Innovate UK (101779). We would like to thank Shichao Dong for his efforts on data processing and anonymous reviewers for the useful comments.

References

David M. Blei, Thomas L. Griffiths, Michael I. Jordan, and Joshua B. Tenenbaum. 2003a. Hierarchical topic models and the nested chinese restaurant process. In *Proceedings of the 17th Annual Conference on Neural Information Processing Systems, NIPS*, pages 17–24.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003b. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Jonathan Chang, Jordan L. Boyd-Graber, Sean Gerrish, Chong Wang, and David M. Blei. 2009. Reading tea leaves: How humans interpret topic models. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems, NIPS*, pages 288–296.

Chaitanya Chemudugunta, Padhraic Smyth, and Mark Steyvers. 2006. Modeling general and specific aspects of documents with a probabilistic topic model. In *Proceedings of the 20th Annual Conference on Neural Information Processing Systems, NIPS*, pages 241–248.

Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235.

Thomas L. Griffiths, Mark Steyvers, David M. Blei, and Joshua B. Tenenbaum. 2004. Integrating topics and syntax. In *Proceedings of the 18th Annual Conference on Neural Information Processing Systems, NIPS*, pages 537–544.

Tom Griffiths. 2002. Gibbs sampling in the generative model of latent dirichlet allocation.

Sanda M. Harabagiu and Andrew Hickl. 2011. Relevance modeling for microblog summarization. In *Proceedings of the 5th International Conference on Web and Social Media, ICWSM*.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *In Proceedings of the 22nd Annual International, ACM SIGIR*, pages 50–57.

Liangjie Hong and Brian D Davison. 2010. Empirical study of topic modeling in twitter. In *Proceedings of the first workshop on social media analytics*, pages 80–88.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning, ICML*, pages 282–289.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*, pages 159–174.

Jing Li, Wei Gao, Zhongyu Wei, Baolin Peng, and Kam-Fai Wong. 2015. Using content-level structures for summarizing microblog repost trees. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 2168–2178.

Cindy Xide Lin, Bo Zhao, Qiaozhu Mei, and Jiawei Han. 2010. PET: a statistical model for popular

events tracking in social communities. In *Proceedings of the 16th International Conference on Knowledge Discovery and Data Mining, ACM SIGKDD*, pages 929–938.

Rishabh Mehrotra, Scott Sanner, Wray L. Buntine, and Lexing Xie. 2013. Improving LDA topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th International conference on research and development in Information Retrieval, ACM SIGIR*, pages 889–892.

David M. Mimno, Hanna M. Wallach, Edmund M. Talley, Miriam Leenders, and Andrew McCallum. 2011. Optimizing semantic coherence in topic models. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 262–272.

Xipeng Qiu, Qi Zhang, and Xuanjing Huang. 2013. Fudannlp: A toolkit for chinese natural language processing. In *51st Annual Meeting of the Association for Computational Linguistics, ACL*, pages 49–54.

Xiaojun Quan, Chunyu Kit, Yong Ge, and Sinno Jialin Pan. 2015. Short and sparse text topic modeling via self-aggregation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence, IJCAI*, pages 2270–2276.

Daniel Ramage, Susan T. Dumais, and Daniel J. Liebling. 2010. Characterizing microblogs with topic models. In *Proceedings of the 4th International Conference on Web and Social Media, ICWSM*.

Kenneth Rapoza. 2011. China's weibos vs us's twitter: And the winner is? *Forbes (May 17, 2011)*.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Proceedings of the 2010 Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*, pages 172–180.

Vivek Kumar Rangarajan Sridhar. 2015. Unsupervised entity linking with abstract meaning representation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT*, pages 1130–1139.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-markov continuous-time model of topical trends. In *Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining, ACM SIGKDD*, pages 424–433.

Hongning Wang, Duo Zhang, and ChengXiang Zhai. 2011. Structural topic model for latent topical structure analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 1526–1535.

Jianshu Weng, Ee-Peng Lim, Jing Jiang, and Qi He. 2010. Twitterrank: finding topic-sensitive influential twitterers. In *Proceedings of the 3rd International Conference on Web Search and Web Data Mining, WSDM*, pages 261–270.

Xiaohui Yan, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2013. A biterm topic model for short texts. In *Proceedings of the 22nd International World Wide Web Conference, WWW*, pages 1445–1456.

Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. 2011. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval - 33rd European Conference on IR Research, ECIR*, pages 338–349.

Neural Relation Extraction with Selective Attention over Instances

Yankai Lin[1], **Shiqi Shen**[1], **Zhiyuan Liu**[1,2]*, **Huanbo Luan**[1], **Maosong Sun**[1,2]
[1] Department of Computer Science and Technology,
State Key Lab on Intelligent Technology and Systems,
National Lab for Information Science and Technology, Tsinghua University, Beijing, China
[2] Jiangsu Collaborative Innovation Center for Language Competence, Jiangsu, China

Abstract

Distant supervised relation extraction has been widely used to find novel relational facts from text. However, distant supervision inevitably accompanies with the wrong labelling problem, and these noisy data will substantially hurt the performance of relation extraction. To alleviate this issue, we propose a sentence-level attention-based model for relation extraction. In this model, we employ convolutional neural networks to embed the semantics of sentences. Afterwards, we build sentence-level attention over multiple instances, which is expected to dynamically reduce the weights of those noisy instances. Experimental results on real-world datasets show that, our model can make full use of all informative sentences and effectively reduce the influence of wrong labelled instances. Our model achieves significant and consistent improvements on relation extraction as compared with baselines. The source code of this paper can be obtained from `https://github.com/thunlp/NRE`.

1 Introduction

In recent years, various large-scale knowledge bases (KBs) such as Freebase (Bollacker et al., 2008), DBpedia (Auer et al., 2007) and YAGO (Suchanek et al., 2007) have been built and widely used in many natural language processing (NLP) tasks, including web search and question answering. These KBs mostly compose of relational facts with triple format, e.g., (*Microsoft*, `founder`, *Bill Gates*). Although existing KBs contain a massive amount of facts, they are still far from complete compared to the infinite real-world facts. To enrich KBs, many efforts have been invested in automatically finding unknown relational facts. Therefore, relation extraction (RE), the process of generating relational data from plain text, is a crucial task in NLP.

Most existing supervised RE systems require a large amount of labelled relation-specific training data, which is very time consuming and labor intensive. (Mintz et al., 2009) proposes distant supervision to automatically generate training data via aligning KBs and texts. They assume that if two entities have a relation in KBs, then all sentences that contain these two entities will express this relation. For example, (*Microsoft*, `founder`, *Bill Gates*) is a relational fact in KB. Distant supervision will regard all sentences that contain these two entities as active instances for relation `founder`. Although distant supervision is an effective strategy to automatically label training data, it always suffers from wrong labelling problem. For example, the sentence "*Bill Gates 's turn to philanthropy was linked to the antitrust problems Microsoft had in the U.S. and the European union.*" does not express the relation `founder` but will still be regarded as an active instance. Hence, (Riedel et al., 2010; Hoffmann et al., 2011; Surdeanu et al., 2012) adopt multi-instance learning to alleviate the wrong labelling problem. The main weakness of these conventional methods is that most features are explicitly derived from NLP tools such as POS tagging and the errors generated by NLP tools will propagate in these methods.

Some recent works (Socher et al., 2012; Zeng et al., 2014; dos Santos et al., 2015) attempt to use deep neural networks in relation classification without handcrafted features. These methods build classifier based on sentence-level annotated data, which cannot be applied in large-scale

* Corresponding author: Zhiyuan Liu (liuzy@tsinghua.edu.cn).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2124–2133,
Berlin, Germany, August 7-12, 2016.

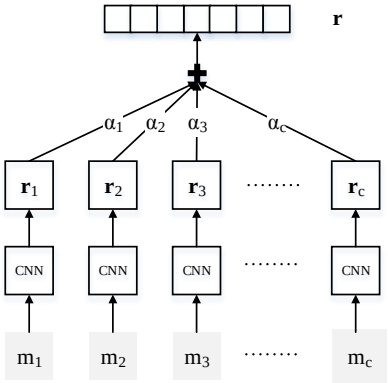


Figure 1: The architecture of sentence-level attention-based CNN, where m_i indicates the original sentence for an entity pair, α_i is the weight given by sentence-level attention.

KBs due to the lack of human-annotated training data. Therefore, (Zeng et al., 2015) incorporates multi-instance learning with neural network model, which can build relation extractor based on distant supervision data. Although the method achieves significant improvement in relation extraction, it is still far from satisfactory. The method assumes that at least one sentence that mentions these two entities will express their relation, and only selects the most likely sentence for each entity pair in training and prediction. It's apparent that the method will lose a large amount of rich information containing in neglected sentences.

In this paper, we propose a sentence-level attention-based convolutional neural network (CNN) for distant supervised relation extraction. As illustrated in Fig. 1, we employ a CNN to embed the semantics of sentences. Afterwards, to utilize all informative sentences, we represent the relation as semantic composition of sentence embeddings. To address the wrong labelling problem, we build sentence-level attention over multiple instances, which is expected to dynamically reduce the weights of those noisy instances. Finally, we extract relation with the relation vector weighted by sentence-level attention. We evaluate our model on a real-world dataset in the task of relation extraction. The experimental results show that our model achieves significant and consistent improvements in relation extraction as compared with the state-of-the-art methods.

The contributions of this paper can be summarized as follows:

- As compared to existing neural relation extraction model, our model can make full use of all informative sentences of each entity pair.
- To address the wrong labelling problem in distant supervision, we propose selective attention to de-emphasize those noisy instances.
- In the experiments, we show that selective attention is beneficial to two kinds of CNN models in the task of relation extraction.

2 Related Work

Relation extraction is one of the most important tasks in NLP. Many efforts have been invested in relation extraction, especially in supervised relation extraction. Most of these methods need a great deal of annotated data, which is time consuming and labor intensive. To address this issue, (Mintz et al., 2009) aligns plain text with Freebase by distant supervision. However, distant supervision inevitably accompanies with the wrong labelling problem. To alleviate the wrong labelling problem, (Riedel et al., 2010) models distant supervision for relation extraction as a multi-instance single-label problem, and (Hoffmann et al., 2011; Surdeanu et al., 2012) adopt multi-instance multi-label learning in relation extraction. Multi-instance learning was originally proposed to address the issue of ambiguously-labelled training data when predicting the activity of drugs (Dietterich et al., 1997). Multi-instance learning considers the reliability of the labels for each instance. (Bunescu and Mooney, 2007) connects weak supervision with multi-instance learning and extends it to relation extraction. But all the feature-based methods depend strongly on the quality of the features generated by NLP tools, which will suffer from error propagation problem.

Recently, deep learning (Bengio, 2009) has been widely used for various areas, including computer vision, speech recognition and so on. It has also been successfully applied to different NLP tasks such as part-of-speech tagging (Collobert et al., 2011), sentiment analysis (dos Santos and Gatti, 2014), parsing (Socher et al., 2013), and machine translation (Sutskever et al., 2014). Due to the recent success in deep learning, many researchers have investigated the possibility of using neural networks to automatically learn features

for relation extraction. (Socher et al., 2012) uses a recursive neural network in relation extraction. They parse the sentences first and then represent each node in the parsing tree as a vector. Moreover, (Zeng et al., 2014; dos Santos et al., 2015) adopt an end-to-end convolutional neural network for relation extraction. Besides, (Xie et al., 2016) attempts to incorporate the text information of entities for relation extraction.

Although these methods achieve great success, they still extract relations on sentence-level and suffer from a lack of sufficient training data. In addition, the multi-instance learning strategy of conventional methods cannot be easily applied in neural network models. Therefore, (Zeng et al., 2015) combines at-least-one multi-instance learning with neural network model to extract relations on distant supervision data. However, they assume that only one sentence is active for each entity pair. Hence, it will lose a large amount of rich information containing in those neglected sentences. Different from their methods, we propose sentence-level attention over multiple instances, which can utilize all informative sentences.

The attention-based models have attracted a lot of interests of researchers recently. The selectivity of attention-based models allows them to learn alignments between different modalities. It has been applied to various areas such as image classification (Mnih et al., 2014), speech recognition (Chorowski et al., 2014), image caption generation (Xu et al., 2015) and machine translation (Bahdanau et al., 2014). To the best of our knowledge, this is the first effort to adopt attention-based model in distant supervised relation extraction.

3 Methodology

Given a set of sentences $\{x_1, x_2, \cdots, x_n\}$ and two corresponding entities, our model measures the probability of each relation r. In this section, we will introduce our model in two main parts:

- **Sentence Encoder.** Given a sentence x and two target entities, a convolutional neutral network (CNN) is used to construct a distributed representation $\mathbf{x}$ of the sentence.
- **Selective Attention over Instances**. When the distributed vector representations of all sentences are learnt, we use sentence-level attention to select the sentences which really express the corresponding relation.

3.1 Sentence Encoder

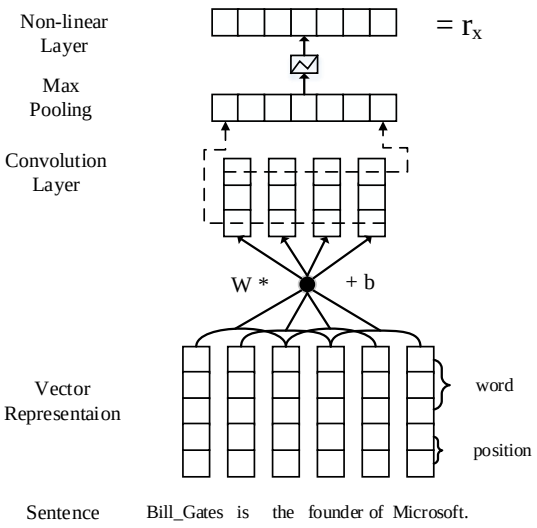


Figure 2: The architecture of CNN/PCNN used for sentence encoder.

As shown in Fig. 2, we transform the sentence x into its distributed representation $\mathbf{x}$ by a CNN. First, words in the sentence are transformed into dense real-valued feature vectors. Next, convolutional layer, max-pooling layer and non-linear transformation layer are used to construct a distributed representation of the sentence, i.e., $\mathbf{x}$.

3.1.1 Input Representation

The inputs of the CNN are raw words of the sentence x. We first transform words into low-dimensional vectors. Here, each input word is transformed into a vector via word embedding matrix. In addition, to specify the position of each entity pair, we also use position embeddings for all words in the sentence.

Word Embeddings. Word embeddings aim to transform words into distributed representations which capture syntactic and semantic meanings of the words. Given a sentence x consisting of m words $x = \{w_1, w_2, \cdots, w_m\}$, every word w_i is represented by a real-valued vector. Word representations are encoded by column vectors in an embedding matrix $\mathbf{V} \in \mathbb{R}^{d^a \times |V|}$where V is a fixed-sized vocabulary.

Position Embeddings. In the task of relation extraction, the words close to the target entities are usually informative to determine the relation between entities. Similar to (Zeng et al., 2014), we use position embeddings specified by entity pairs. It can help the CNN to keep track of how close

each word is to head or tail entities. It is defined as the combination of the relative distances from the current word to head or tail entities. For example, in the sentence "Bill_Gates is the founder of Microsoft.", the relative distance from the word "founder" to head entity *Bill_Gates* is 3 and tail entity *Microsoft* is 2.

In the example shown in Fig. 2, it is assumed that the dimension d^a of the word embedding is 3 and the dimension d^b of the position embedding is 1. Finally, we concatenate the word embeddings and position embeddings of all words and denote it as a vector sequence $\mathbf{w} = \{\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_m\}$, where $\mathbf{w}_i \in \mathbb{R}^d (d = d^a + d^b \times 2)$.

3.1.2 Convolution, Max-pooling and Non-linear Layers

In relation extraction, the main challenges are that the length of the sentences is variable and the important information can appear in any area of the sentences. Hence, we should utilize all local features and perform relation prediction globally. Here, we use a convolutional layer to merge all these features. The convolutional layer first extracts local features with a sliding window of length l over the sentence. In the example shown in Fig. 2, we assume that the length of the sliding window l is 3. Then, it combines all local features via a max-pooling operation to obtain a fixed-sized vector for the input sentence.

Here, convolution is defined as an operation between a vector sequence $\mathbf{w}$ and a convolution matrix $\mathbf{W} \in \mathbb{R}^{d^c \times (l \times d)}$, where d^c is the sentence embedding size. Let us define the vector $\mathbf{q}_i \in \mathbb{R}^{l \times d}$ as the concatenation of a sequence of w word embeddings within the i-th window:

$$\mathbf{q}_i = \mathbf{w}_{i-l+1:i} \quad (1 \leq i \leq m + l - 1). \tag{1}$$

Since the window may be outside of the sentence boundaries when it slides near the boundary, we set special padding tokens for the sentence. It means that we regard all out-of-range input vectors $\mathbf{w}_i (i < 1 \text{ or } i > m)$ as zero vector.

Hence, the i-th filter of convolutional layer is computed as:

$$\mathbf{p}_i = [\mathbf{W}\mathbf{q} + \mathbf{b}]_i \tag{2}$$

where $\mathbf{b}$ is bias vector. And the i-th element of the vector $\mathbf{x} \in \mathbb{R}^{d^c}$ as follows:

$$[\mathbf{x}]_i = \max(\mathbf{p}_i), \tag{3}$$

Further, PCNN (Zeng et al., 2015), which is a variation of CNN, adopts piecewise max pooling in relation extraction. Each convolutional filter $\mathbf{p}_i$ is divided into three segments $(\mathbf{p}_{i1}, \mathbf{p}_{i2}, \mathbf{p}_{i3})$ by head and tail entities. And the max pooling procedure is performed in three segments separately, which is defined as:

$$[\mathbf{x}]_{ij} = \max(\mathbf{p}_{ij}), \tag{4}$$

And $[\mathbf{x}]_i$ is set as the concatenation of $[\mathbf{x}]_{ij}$.

Finally, we apply a non-linear function at the output, such as the hyperbolic tangent.

3.2 Selective Attention over Instances

Suppose there is a set S contains n sentences for entity pair $(head, tail)$, i.e., $S = \{x_1, x_2, \cdots, x_n\}$.

To exploit the information of all sentences, our model represents the set S with a real-valued vector $\mathbf{s}$ when predicting relation r. It is straightforward that the representation of the set S depends on all sentences' representations $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$. Each sentence representation $\mathbf{x}_i$ contains information about whether entity pair $(head, tail)$ contains relation r for input sentence x_i.

The set vector $\mathbf{s}$ is, then, computed as a weighted sum of these sentence vector $\mathbf{x}_i$:

$$\mathbf{s} = \sum_i \alpha_i \mathbf{x}_i, \tag{5}$$

where α_i is the weight of each sentence vector $\mathbf{x}_i$. In this paper, we define α_i in two ways:

Average: We assume that all sentences in the set X have the same contribution to the representation of the set. It means the embedding of the set S is the average of all the sentence vectors:

$$\mathbf{s} = \sum_i \frac{1}{n} \mathbf{x}_i, \tag{6}$$

It's a naive baseline of our selective attention.

Selective Attention: However, the wrong labelling problem inevitably occurs. Thus, if we regard each sentence equally, the wrong labelling sentences will bring in massive of noise during training and testing. Hence, we use a selective attention to de-emphasize the noisy sentence. Hence, α_i is further defined as:

$$\alpha_i = \frac{\exp(e_i)}{\sum_k \exp(e_k)}, \tag{7}$$

where e_i is referred as a query-based function which scores how well the input sentence x_i and the predict relation r matches. We select the bilinear form which achieves best performance in different alternatives:

$$e_i = \mathbf{x}_i \mathbf{A} \mathbf{r}, \tag{8}$$

where $\mathbf{A}$ is a weighted diagonal matrix, and $\mathbf{r}$ is the query vector associated with relation r which indicates the representation of relation r.

Finally, we define the conditional probability $p(r|S, \theta)$ through a softmax layer as follows:

$$p(r|S, \theta) = \frac{\exp(o_r)}{\sum_{k=1}^{n_r} \exp(o_k)}, \tag{9}$$

where n_r is the total number of relations and $\mathbf{o}$ is the final output of the neural network which corresponds to the scores associated to all relation types, which is defined as follows:

$$\mathbf{o} = \mathbf{M}\mathbf{s} + \mathbf{d}, \tag{10}$$

where $\mathbf{d} \in \mathbb{R}^{n_r}$ is a bias vector and $\mathbf{M}$ is the representation matrix of relations.

(Zeng et al., 2015) follows the assumption that at least one mention of the entity pair will reflect their relation, and only uses the sentence with the highest probability in each set for training. Hence, the method which they adopted for multi-instance learning can be regarded as a special case as our selective attention when the weight of the sentence with the highest probability is set to 1 and others to 0.

3.3 Optimization and Implementation Details

Here we introduce the learning and optimization details of our model. We define the objective function using cross-entropy at the set level as follows:

$$J(\theta) = \sum_{i=1}^{s} \log p(r_i|S_i, \theta), \tag{11}$$

where s indicates the number of sentence sets and θ indicates all parameters of our model. To solve the optimization problem, we adopt stochastic gradient descent (SGD) to minimize the objective function. For learning, we iterate by randomly selecting a mini-batch from the training set until converge.

In the implementation, we employ dropout (Srivastava et al., 2014) on the output layer to prevent overfitting. The dropout layer is defined as an element-wise multiplication with a a vector $\mathbf{h}$ of Bernoulli random variables with probability p. Then equation (10) is rewritten as:

$$\mathbf{o} = \mathbf{M}(\mathbf{s} \circ \mathbf{h}) + \mathbf{d}. \tag{12}$$

In the test phase, the learnt set representations are scaled by p, i.e., $\hat{\mathbf{s}}_i = p\mathbf{s}_i$. And the scaled set vector $\hat{\mathbf{r}}_i$ is finally used to predict relations.

4 Experiments

Our experiments are intended to demonstrate that our neural models with sentence-level selective attention can alleviate the wrong labelling problem and take full advantage of informative sentences for distant supervised relation extraction. To this end, we first introduce the dataset and evaluation metrics used in the experiments. Next, we use cross-validation to determine the parameters of our model. And then we evaluate the effects of our selective attention and show its performance on the data with different set size. Finally, we compare the performance of our method to several state-of-the-art feature-based methods.

4.1 Dataset and Evaluation Metrics

We evaluate our model on a widely used dataset[1] which is developed by (Riedel et al., 2010) and has also been used by (Hoffmann et al., 2011; Surdeanu et al., 2012). This dataset was generated by aligning Freebase relations with the New York Times corpus (NYT). Entity mentions are found using the Stanford named entity tagger (Finkel et al., 2005), and are further matched to the names of Freebase entities. The Freebase relations are divided into two parts, one for training and one for testing. It aligns the the sentences from the corpus of the years 2005-2006 and regards them as training instances. And the testing instances are the aligned sentences from 2007. There are 53 possible relationships including a special relation NA which indicates there is no relation between head and tail entities. The training data contains 522,611 sentences, 281,270 entity pairs and 18,252 relational facts. The testing set contains 172,448 sentences, 96,678 entity pairs and 1,950 relational facts.

Similar to previous work (Mintz et al., 2009), we evaluate our model in the held-out evaluation. It evaluates our model by comparing the relation

[1] http://iesl.cs.umass.edu/riedel/ecml/

facts discovered from the test articles with those in Freebase. It assumes that the testing systems have similar performances in relation facts inside and outside Freebase. Hence, the held-out evaluation provides an approximate measure of precision without time consumed human evaluation. We report both the aggregate curves precision/recall curves and Precision@N (P@N) in our experiments.

4.2 Experimental Settings

4.2.1 Word Embeddings

In this paper, we use the word2vec tool [2] to train the word embeddings on NYT corpus. We keep the words which appear more than 100 times in the corpus as vocabulary. Besides, we concatenate the words of an entity when it has multiple words.

4.2.2 Parameter Settings

Following previous work, we tune our models using three-fold validation on the training set. We use a grid search to determine the optimal parameters and select learning rate λ for SGD among $\{0.1, 0.01, 0.001, 0.0001\}$, the sliding window size $l \in \{1, 2, 3, \cdots, 8\}$, the sentence embedding size $n \in \{50, 60, \cdots, 300\}$, and the batch size B among $\{40, 160, 640, 1280\}$. For other parameters, since they have little effect on the results, we follow the settings used in (Zeng et al., 2014). For training, we set the iteration number over all the training data as 25. In Table 1 we show all parameters used in the experiments.

Table 1: Parameter settings

Window size l	3
Sentence embedding size d^c	230
Word dimension d^a	50
Position dimension d^b	5
Batch size B	160
Learning rate λ	0.01
Dropout probability p	0.5

4.3 Effect of Sentence-level Selective Attention

To demonstrate the effects of the sentence-level selective attention, we empirically compare different methods through held-out evaluation. We select the CNN model proposed in (Zeng et al., 2014) and the PCNN model proposed in (Zeng et al., 2015) as our sentence encoders and implement them by ourselves which achieve comparable results as the authors reported. And we compare the performance of the two different kinds of CNN with sentence-level attention (ATT) , its naive version (AVE) which represents each sentence set as the average vector of sentences inside the set and the at-least-one multi-instance learning (ONE) used in (Zeng et al., 2015).

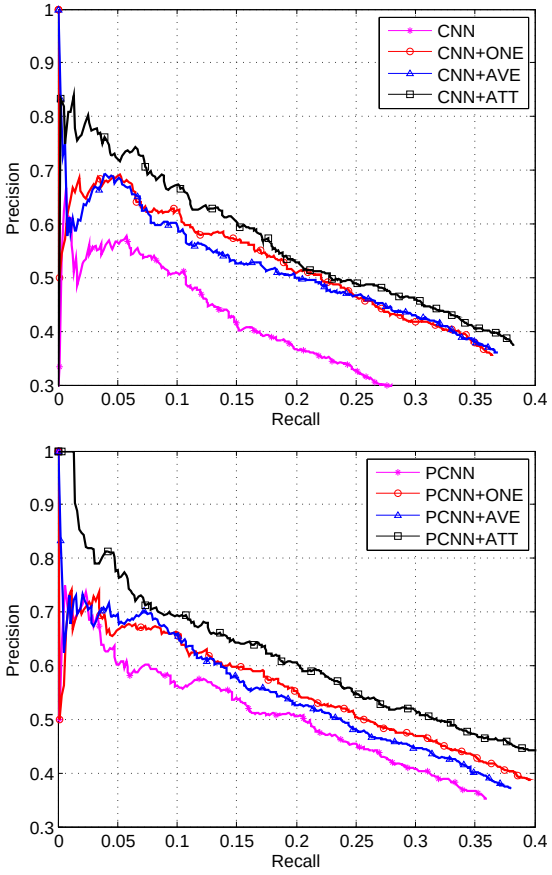


Figure 3: Top: Aggregate precion/recall curves of CNN, CNN+ONE, CNN+AVE, CNN+ATT. Bottom: Aggregate precion/recall curves of PCNN, PCNN+ONE, PCNN+AVE, PCNN+ATT

From Fig. 3, we have the following observation: (1) For both CNN and PCNN, the ONE method brings better performance as compared to CNN/PCNN. The reason is that the original distant supervision training data contains a lot of noise and the noisy data will damage the performance of relation extraction. (2) For both CNN and PCNN, the AVE method is useful for relation extraction as compared to CNN/PCNN. It indicates that considering more sentences is beneficial to relation extraction since the noise can be reduced by mutual complementation of information. (3) For both

[2] https://code.google.com/p/word2vec/

CNN and PCNN, the AVE method has a similar performance compared to the ONE method. It indicates that, although the AVE method brings in information of more sentences, since it regards each sentence equally, it also brings in the noise from the wrong labelling sentences which may hurt the performance of relation extraction. (4) For both CNN and PCNN, the ATT method achieves the highest precision over the entire range of recall compared to other methods including the AVE method. It indicates that the proposed selective attention is beneficial. It can effectively filter out meaningless sentences and alleviate the wrong labelling problem in distant supervised relation extraction.

4.4 Effect of Sentence Number

In the original testing data set, there are 74,857 entity pairs that correspond to only one sentence, nearly 3/4 over all entity pairs. Since the superiority of our selective attention lies in the entity pairs containing multiple sentences, we compare the performance of CNN/PCNN+ONE, CNN/PCNN+AVE and CNN/PCNN+ATT on the entity pairs which have more than one sentence. And then we examine these three methods in three test settings:

- **One**: For each testing entity pair, we randomly select one sentence and use this sentence to predict relation.
- **Two**: For each testing entity pair, we randomly select two sentences and proceed relation extraction.
- **All**: We use all sentences of each entity pair for relation extraction.

Note that, we use all the sentences in training. We will report the P@100, P@200, P@300 and the mean of them for each model in held-out evaluation.

Table 2 shows the P@N for compared models in three test settings. From the table, we can see that: (1) For both CNN and PCNN, the ATT method achieves the best performance in all test settings. It demonstrates the effectiveness of sentence-level selective attention for multi-instance learning. (2) For both CNN and PCNN, the AVE method is comparable to the ATT method in the One test setting. However, when the number of testing sentences per entity pair grows, the performance of the AVE methods has almost no improvement. It even drops gradually in P@100, P@200 as the sentence number increases. The reason is that, since we regard each sentence equally, the noise contained in the sentences that do not express any relation will have negative influence in the performance of relation extraction. (3) CNN+AVE and CNN+ATT have 5% to 8% improvements compared to CNN+ONE in the ONE test setting. Since each entity pair has only one sentence in this test setting, the only difference of these methods is from training. Hence, it shows that utilizing all sentences will bring in more information although it may also bring in some extra noises. (4) For both CNN and PCNN, the ATT method outperforms other two baselines over 5% and 9% in the Two and All test settings. It indicates that by taking more useful information into account, the relational facts which CNN+ATT ranks higher are more reliable and beneficial to relation extraction.

4.5 Comparison with Feature-based Approaches

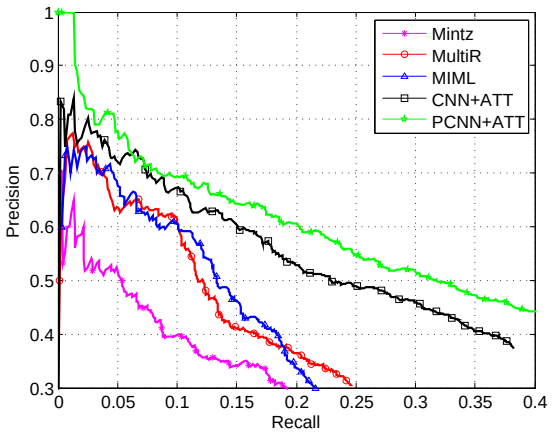


Figure 4: Performance comparison of proposed model and traditional methods

To evaluate the proposed method, we select the following three feature-based methods for comparison through held-out evaluation:

Mintz (Mintz et al., 2009) is a traditional distant supervised model.

MultiR (Hoffmann et al., 2011) proposes a probabilistic, graphical model of multi-instance learning which handles overlapping relations.

MIML (Surdeanu et al., 2012) jointly models both multiple instances and multiple relations.

We implement them with the source codes released by the authors.

Table 2: P@N for relation extraction in the entity pairs with different number of sentences

Test Settings	One				Two				All			
P@N(%)	100	200	300	Mean	100	200	300	Mean	100	200	300	Mean
CNN+ONE	68.3	60.7	53.8	60.9	70.3	62.7	55.8	62.9	67.3	64.7	58.1	63.4
+AVE	75.2	67.2	58.8	67.1	68.3	63.2	60.5	64.0	64.4	60.2	60.1	60.4
+ATT	**76.2**	65.2	**60.8**	67.4	76.2	65.7	62.1	68.0	**76.2**	68.6	59.8	68.2
PCNN+ONE	73.3	64.8	56.8	65.0	70.3	67.2	63.1	66.9	72.3	69.7	64.1	68.7
+AVE	71.3	63.7	57.8	64.3	73.3	65.2	62.1	66.9	73.3	66.7	62.8	67.6
+ATT	73.3	**69.2**	**60.8**	**67.8**	**77.2**	**71.6**	**66.1**	**71.6**	**76.2**	**73.1**	**67.4**	**72.2**

Fig. 4 shows the precision/recall curves for each method. We can observe that: (1) CNN/PCNN+ATT significantly outperforms all feature-based methods over the entire range of recall. When the recall is greater than 0.1, the performance of feature-based method drop out quickly. In contrast, our model has a reasonable precision until the recall approximately reaches 0.3. It demonstrates that the human-designed feature cannot concisely express the semantic meaning of the sentences, and the inevitable error brought by NLP tools will hurt the performance of relation extraction. In contrast, CNN/PCNN+ATT which learns the representation of each sentences automatically can express each sentence well. (2) PCNN+ATT performs much better as compared to CNN+ATT over the entire range of recall. It means that the selective attention considers the global information of all sentences except the information inside each sentence. Hence, the performance of our model can be further improved if we have a better sentence encoder.

4.6 Case Study

Table 3 shows two examples of selective attention from the testing data. For each relation, we show the corresponding sentences with highest and lowest attention weight respectively. And we highlight the entity pairs with bold formatting.

From the table we find that: The former example is related to the relation `employer_of`. The sentence with low attention weight does not express the relation between two entities, while the high one shows that *Mel Karmazin* is the chief executive of *Sirius Satellite Radio*. The later example is related to the relation `place_of_birth`. The sentence with low attention weight expresses where *Ernst Haefliger* is died in, while the high one expresses where he is born in.

Table 3: Some examples of selective attention in NYT corpus

Relation	employer_of
Low	When Howard Stern was preparing to take his talk show to **Sirius Satellite Radio**, following his former boss, **Mel Karmazin**, Mr. Hollander argued that ...
High	**Mel Karmazin**, the chief executive of **Sirius Satellite Radio**, made a lot of phone calls ...
Relation	place_of_birth
Low	**Ernst Haefliger**, a Swiss tenor who ... roles , died on Saturday in **Davos**, Switzerland, where he maintained a second home.
High	**Ernst Haefliger** was born in **Davos** on July 6, 1919, and studied at the Wettinger Seminary ...

5 Conclusion and Future Works

In this paper, we develop CNN with sentence-level selective attention. Our model can make full use of all informative sentences and alleviate the wrong labelling problem for distant supervised relation extraction. In experiments, we evaluate our model on relation extraction task. The experimental results show that our model significantly and consistently outperforms state-of-the-art feature-based methods and neural network methods.

In the future, we will explore the following directions:

- Our model incorporates multi-instance learning with neural network via instance-level selective attention. It can be used in not only distant supervised relation extraction but also other multi-instance learning tasks. We will explore our model in other area such as text

categorization.

- CNN is one of the effective neural networks for neural relation extraction. Researchers also propose many other neural network models for relation extraction. In the future, we will incorporate our instance-level selective attention technique with those models for relation extraction.

Acknowledgments

This work is supported by the 973 Program (No. 2014CB340501), the National Natural Science Foundation of China (NSFC No. 61572273, 61303075) and the Tsinghua University Initiative Scientific Research Program (20151080406).

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of KDD*, pages 1247–1250.

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of ACL*, volume 45, page 576.

Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. End-to-end continuous speech recognition using attention-based recurrent nn: first results. *arXiv preprint arXiv:1412.1602*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 89(1):31–71.

Cıcero Nogueira dos Santos and Maıra Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING*.

Cıcero Nogueira dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of ACL*, volume 1, pages 626–634.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*, pages 363–370. Association for Computational Linguistics.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of ACL-HLT*, pages 541–550.

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

Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Proceedings of NIPS*, pages 2204–2212.

Sebastian Riedel, Limin Yao, and Andrew McCallum. 2010. Modeling relations and their mentions without labeled text. In *Proceedings of ECML-PKDD*, pages 148–163.

Richard Socher, Brody Huval, Christopher D Manning, and Andrew Y Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP-CoNLL*, pages 1201–1211.

Richard Socher, John Bauer, Christopher D Manning, and Andrew Y Ng. 2013. Parsing with compositional vector grammars. In *Proceedings of ACL*. Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of WWW*, pages 697–706. ACM.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *Proceedings of EMNLP*, pages 455–465.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, pages 3104–3112.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. *Proceedings of ICML*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *Proceedings of COLING*, pages 2335–2344.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *Proceedings of EMNLP*.

Leveraging FrameNet to Improve Automatic Event Detection

Shulin Liu, Yubo Chen, Shizhu He, Kang Liu and Jun Zhao
National Laboratory of Pattern Recognition
Institute of Automation, Chinese Academy of Sciences, Beijing, 100190, China
{shulin.liu, yubo.chen, shizhu.he, kliu, jzhao}@nlpr.ia.ac.cn

Abstract

Frames defined in FrameNet (FN) share highly similar structures with events in ACE event extraction program. An event in ACE is composed of an event trigger and a set of arguments. Analogously, a frame in FN is composed of a lexical unit and a set of frame elements, which play similar roles as triggers and arguments of ACE events respectively. Besides having similar structures, many frames in FN actually express certain types of events. The above observations motivate us to explore whether there exists a good mapping from frames to event-types and if it is possible to improve event detection by using FN. In this paper, we propose a global inference approach to detect events in FN. Further, based on the detected results, we analyze possible mappings from frames to event-types. Finally, we improve the performance of event detection and achieve a new state-of-the-art result by using the events automatically detected from FN.

1 Introduction

In the ACE (Automatic Context Extraction) event extraction program, an event is represented as a structure consisting of an event trigger and a set of arguments. This paper tackles with the event detection (ED) task, which is a crucial component in the overall task of event extraction. The goal of ED is to identify event triggers and their corresponding event types from the given documents.

FrameNet (FN) (Baker et al., 1998; Fillmore et al., 2003) is a linguistic resource storing considerable information about lexical and predicate-argument semantics. In FN, a frame is defined as a composition of a Lexical Unit (LU) and a set of Frame Elements (FE). Most frames contain a set of exemplars with annotated LUs and FEs (see Figure 2 and Section 2.2 for details).

From the above definitions of events and frames, it is not hard to find that the frames defined in FN share highly similar structures as the events defined in ACE. Firstly, the LU of a Frame plays a similar role as the trigger of an event. ACE defines the trigger of an event as the word or phrase which most clearly expresses an event occurrence. For example, the following sentence "*He **died** in the hospital.*" expresses a `Die` event, whose trigger is the word *died*. Analogously, the LU of a frame is also the word or phrase which is capable of indicating the occurrence of the expressed semantic frame. For example, the sentence "*Aeroplanes **bombed** London.*" expresses an `Attack`[1] frame, whose LU is the word *bombed*. Secondly, the FEs of a frame also play similar roles as arguments of an event. Both of them indicate the participants involved in the corresponding frame or event. For example, in the first sentence, *He* and *hospital* are the arguments, and in the second sentence, *Aeroplanes* and *London* are the FEs.

Besides having similar structure as events, many frames in FN actually express certain types of events defined in ACE. Table 1 shows some examples of frames which also express events.

Frame	Event	Sample in FN
Attack	Attack	*Aeroplanes **bombed** London.*
Invading	Attack	*Hitler **invaded** Austria .*
Fining	Fine	*The court **fined** her $40.*
Execution	Execute	*He was **executed** yesterday.*

Table 1: Examples of frames expressing events.

The aforementioned observations motivate us to

[1]The notation of frames distinguishes from that of events by the italic decoration.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2134–2143,
Berlin, Germany, August 7-12, 2016.

explore: (1) whether there exists a good mapping from frames to event-types, and (2) whether it is possible to improve ED by using FN.

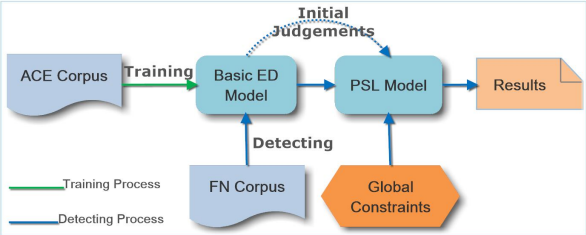


Figure 1: Our framework for detecting events in FN (including training and detecting processes).

For the first issue, we investigate whether a frame could be mapped to an event-type based on events expressed by exemplars annotated for that frame. Therefore the key is to detect events from the given exemplar sentences in FN. To achieve this goal, we propose a global inference approach (see figure 1). We firstly learn a basic ED model based on the ACE labeled corpus and employ it to yield initial judgements for each sentence in FN. Then, we apply a set of soft constraints for global inference based on the following hypotheses: 1). Sentences belonging to the same LU tend to express events of the same type; 2). Sentences belonging to related frames tend to express events of the same type; 3). Sentences belonging to the same frame tend to express events of the same type. All of the above constraints and initial judgments are formalized as first-order logic formulas and modeled by Probabilistic Soft Logic (PSL) (Kimmig et al., 2012; Bach et al., 2013). Finally, we obtain the final results via PSL-based global inference. We conduct both manual and automatic evaluations for the detected results.

For the second issue, ED generally suffers from data sparseness due to lack of labeled samples. Some types, such as `Nominate` and `Extradite`, contain even less than 10 labeled samples. Apparently, from such a small scale of training data is difficult to yield a satisfying performance. We notice that ACE corpus only contains about 6,000 labeled instances, while FN contains more than 150,000 exemplars. Thus, a straightforward solution to alleviate the data sparseness problem is to expand the ACE training data by using events detected from FN. The experimental results show that events from FN significantly improve the performance of the event detection task.

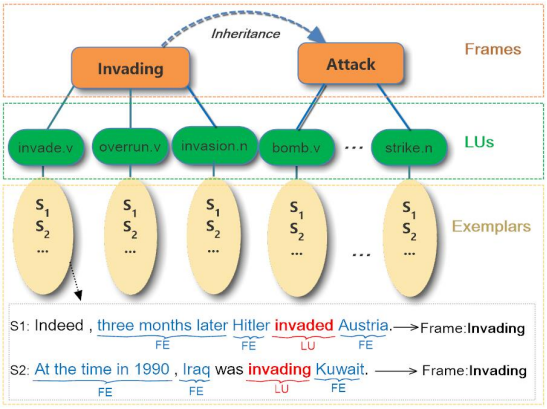


Figure 2: The hierarchy of FN corpus, where each S_k under a LU is a exemplar annotated for that LU. *Inheritance* is a semantic relation between the frames `Invading` and `Attack`.

To sum up, our main contributions are: (1) To our knowledge, this is the first work performing event detection over ACE and FN to explore the relationships between frames and events. (2) We propose a global inference approach to detect events in FN, which is demonstrated very effective by our experiments. Moreover, based on the detected results, we analyze possible mappings from frames to event-types (all the detecting and mapping results are released for further use by the NLP community[2]). (3) We improve the performance of event detection significantly and achieve a new state-of-the-art result by using events automatically detected from FN as extra training data.

2 Background

2.1 ACE Event Extraction

In ACE evaluations, an event is defined as a specific occurrence involving several participants. ACE event evaluation includes 8 types of events, with 33 subtypes. Following previous work, we treat them simply as 33 separate event types and ignore the hierarchical structure among them. In this paper, we use the ACE 2005 corpus[3] in our experiments. It contains 599 documents, which include about 6,000 labeled events.

2.2 FrameNet

The FrameNet is a taxonomy of manually identified semantic frames for English[4]. Figure 2 shows

[2]Available at https://github.com/subacl/acl16

[3]https://catalog.ldc.upenn.edu/LDC2006T06

[4]We use the latest released version, FrameNet 1.5 in this work (http://framenet.icsi.berkeley.edu).

the hierarchy of FN corpus. Listed in the FN with each frame are a set of lemmas with part of speech (i.e "invade.v") that can evoke the frame, which are called lexical units (LUs). Accompanying most LUs in the FN is a set of exemplars annotated for them. Moreover, there are a set of labeled relations between frames, such as *Inheritance*.

FN contains more than 1,000 various frames and 10,000 LUs with 150,000 annotated exemplars. Eight relations are defined between frames in FN, but in this paper we only use the following three of them because the others do not satisfy our hypotheses (see section 4.2):

Inheritance: A inherited from *B* indicates that *A* must correspond to an equally or more specific fact about *B*. It is a directional relation.

See_also: A and *B* connected by this relation indicates that they are similar frames.

Perspective_on: A and *B* connected by this relation means that they are different points-of-view about the same fact (i.e. `Receiving` vs. `Transfer`).

2.3 Related Work

Event extraction is an increasingly hot and challenging research topic in NLP. Many approaches have been proposed for this task. Nearly all the existing methods on ACE event task use supervised paradigm. We further divide them into feature-based methods and representation-based methods.

In feature-based methods, a diverse set of strategies has been exploited to convert classification clues into feature vectors. Ahn (2006) uses the lexical features(e.g., full word), syntactic features (e.g., dependency features) and external-knowledge features(WordNet (Miller, 1995)) to extract the event. Inspired by the hypothesis of One Sense Per Discourse (Yarowsky, 1995), Ji and Grishman (2008) combined global evidence from related documents with local decisions for the event extraction. To capture more clues from the texts, Gupta and Ji (2009), Liao and Grishman (2010) and Hong et al. (2011) proposed the cross-event and cross-entity inference for the ACE event task. Li et al. (2013) proposed a joint model to capture the combinational features of triggers and arguments. Liu et al. (2016) proposed a global inference approach to employ both latent local and global information for event detection.

In representation-based methods, candidate event mentions are represented by embedding, which typically are fed into neural networks. Two similarly related work has been proposed on event detection (Chen et al., 2015; Nguyen and Grishman, 2015). Nguyen and Grishman (2015) employed Convolutional Neural Networks (CNNs) to automatically extract sentence-level features for event detection. Chen et al. (2015) proposed dynamic multi-pooling operation on CNNs to capture better sentence-level features.

FrameNet is a typical resource for frame-semantic parsing, which consists of the resolution of predicate sense into a frame, and the analysis of the frame's participants (Thompson et al., 2003; Giuglea and Moschitti, 2006; Hermann et al., 2014; Das et al., 2014). Other tasks which have been studied based on FN include question answering (Narayanan and Harabagiu, 2004; Shen and Lapata, 2007), textual entailment (Burchardt et al., 2009) and paraphrase recognition (Padó and Lapata, 2005). This is the first work to explore the application of FN to event detection.

3 Basic Event Detection Model

Alike to existing work, we model event detection (ED) as a word classification task. In the ED task, each word in the given sentence is treated as a candidate trigger and the goal is to classify each of these candidates into one of 34 classes (33 event types plus a NA class). However, in this work, as we assumed that the LU of a frame is analogical to the trigger of an event, we only treat the LU annotated in the given sentence as a trigger candidate. Each sentence in FN only contains one candidate trigger, thus "the candidate" denotes both the candidate trigger of a sentence and the sentence itself for FN in the remainder of this paper. Another notable difference is that we train the detection model on one corpus (ACE) but apply it on another (FN). That means our task is also a **cross-domain** problem. To tackle with it, our basic ED approach follows representation-based paradigm, which has been demonstrated effective in the cross-domain situation (Nguyen and Grishman, 2015).

3.1 Model

We employ a simple three-layer (a input layer, a hidden layer and a soft-max output layer) Artificial Neural Networks (ANNs) (Hagan et al., 1996) to model the ED task. In our model, adjacent layers are fully connected.

Word embeddings learned from large amount of unlabeled data have been shown to be able to capture the meaningful semantic regularities of words

(Bengio et al., 2003; Erhan et al., 2010). This paper uses unsupervised learned word embeddings as the source of base features. We use the Skip-gram model (Mikolov et al., 2013) to learn word embeddings on the NYT corpus[5].

Given a sentence, we concatenate the embedding vector of the candidate trigger and the average embedding vector of the words in the sentence as the input to our model. We train the model using a simple optimization technique called stochastic gradient descent (SGD) over shuffled mini-batches with the Adadelta update rule (Zeiler, 2012). Regularization is implemented by a dropout (Kim, 2014; Hinton et al., 2012). The experiments show that this simple model is surprisingly effective for event detection.

4 Event Detection in FrameNet

To detect events in FN, we first learned the basic ED model based on ACE labeled corpus and then employ it to generate initial judgements (possible event types with confidence values) for each sentence in FN. Then, we apply a set of constraints for global inference based on the PSL model.

4.1 Probabilistic Soft Logic

PSL is a framework for collective, probabilistic reasoning in relational domains (Kimmig et al., 2012; Bach et al., 2013). Similar to Markov Logic Networks (MLNs) (Richardson and Domingos, 2006), it uses weighted first-order logic formulas to compactly encode complex undirected probabilistic graphical models. However, PSL brings two remarkable advantages compared with MLNs. First, PSL relaxes the boolean truth values of MLNs to continuous, soft truth values. This allows for easy integration of continuous values, such as similarity scores. Second, PSL restricts the syntax of first order formulas to that of rules with conjunctive bodies. Together with the soft truth values constraint, the inference in PSL is a convex optimization problem in continuous space and thus can be solved using efficient inference approaches. For further details, see the references (Kimmig et al., 2012; Bach et al., 2013).

4.2 Global Constraints

Our global inference approach is based on the following three hypotheses.

[5]https://catalog.ldc.upenn.edu/LDC2008T19

H1: Same Frame Same Event
This hypothesis indicates that sentences under the same frame tend to express events of the same type. For example, all exemplars annotated for the frame `Rape` express events of type `Attack`, and all sentences under the frame `Clothing` express NA (none) events. With this hypothesis, sentences annotated for the same frame help each other to infer their event types during global inference.

H2: Related Frame Same Event
This hypothesis is an extension of *H1*, which relaxes "the same frame" constraint to "related frames". In this paper, frames are considered to be related if and only if they are connected by one of the following three relations: *Inheritance*, *See_also* and *Perspective_on* (see section 2.2). For example, the frame `Invading` is inherited from `Attack`, and they actually express the same type of event, `Attack`. With this hypothesis, sentences under related frames help each other to infer their event types during global inference.

The previous two hypotheses are basically true for most frames but not perfect. For example, for the frame `Dead_or_alive`, only a few of the sentences under it express `Die` events while the remainder do not. To amend the this flaw, we introduce the third hypothesis.

H3: Same LU Same Event
This hypothesis indicates that sentences under the same LU tend to express events of the same type (as a remind, LUs are under frames). It is looser than the previous two hypotheses thus holds true in more situations. For example, *H3* holds true for the frame `Dead_or_alive` which violates *H1* and *H2*. In FN, LUs annotated for that frame are *alive.a, dead.a, deceased.a, lifeless.a, living.n, undead.a* and *undead.n*. All exemplars under *dead.a, deceased.a* and *lifeless.a* express `Die` events. Therefore, this hypothesis amends the flaws of the former two hypotheses.

On the other hand, the first two hypotheses also help *H3* in some cases. For example, most of the sentences belonging to the LU *suit.n* under the frame `Clothing` are misidentified as `Sue` events due to the ambiguity of the word "suit". However, in this situation, *H1* can help to rectify it because the majority of LUs under `Clothing` are not ambiguous words. Thus, under the first hypothesis, the misidentified results are expected to be corrected by the the results of other exemplars belonging to `Clothing`.

4.3 Inference

To model the above hypotheses as logic formulas in PSL, we introduce a set of predicates (see Table 2), which are grouped into two categories: observed predicates and target predicates. Observed predicates are used to encode evidences, which are always assumed to be known during the inference, while target predicates are unknown and thus need to be predicted.

$CandEvt(c,t)$ is introduced to represent $conf(c,t)$, which is the confidence value generated by the basic ED model for classifying the candidate c as an event of the type t. $SameFr(c_1,c_2)$ indicates whether the candidates c_1 and c_2 belong to the same frame. It is initialized by the indicator function $I_{sf}(c_1,c_2)$, which is defined as follows:

$$I_{sf}(c_1,c_2) = \begin{cases} 1 & c_1, c_2 \text{ from the same frame} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$SameLU(c_1,c_2)$ is similar, but applies for candidates under the same LU. The last three observed predicates in Table 2 are used to encode the aforementioned semantic relations between frames. For example, $Inherit(c_1,c_2)$ indicates whether the frame of c_1 is inherited from that of c_2, and it is initialized by the indicator function $I_{ih}(c_1,c_2)$, which is set to 1 if and only if the frame of c_1 is inherited from that of c_2, otherwise 0. $Evt(c,t)$ is the only target predicate, which indicates that the candidate c triggers an event of type t.

Type	Predicate	Assignment
Observed	$CandEvt(c,t)$	$conf(c,t)$
	$SameFr(c_1,c_2)$	$I_{sf}(c_1,c_2)$
	$SameLU(c_1,c_2)$	$I_{sl}(c_1,c_2)$
	$Inherit(c_1,c_2)$	$I_{ih}(c_1,c_2)$
	$SeeAlso(c_1,c_2)$	$I_{sa}(c_1,c_2)$
	$Perspect(c_1,c_2)$	$I_{pe}(c_1,c_2)$
Target	$Evt(c,t)$	—

Table 2: Predicates and their initial assignments.

Putting all the predicates together, we design a set of formulas to apply the aforementioned hypotheses in PSL (see Table 3). Formula f_1 connects the target predicate with the initial judgements from the basic ED model. Formulas f_2 and f_3 respectively encode *H1* and *H3*. Finally, the remaining formulas are designed for various relations between frames in *H2*. We tune the formulas's weights via grid search (see Section 5.4). The inference results provide us with the most likely

	Formulas
f_1	$CandEvt(c,t) \rightarrow Evt(c,t)$
f_2	$SameFr(c_1,c_2) \wedge Evt(c_1,t) \rightarrow Evt(c_2,t)$
f_3	$SameLU(c_1,c_2) \wedge Evt(c_1,t) \rightarrow Evt(c_2,t)$
f_4	$Inherit(c_1,c_2) \wedge Evt(c_1,t) \rightarrow Evt(c_2,t)$
f_5	$SeeAlso(c_1,c_2) \wedge Evt(c_1,t) \rightarrow Evt(c_2,t)$
f_6	$Perspect(c_1,c_2) \wedge Evt(c_1,t) \rightarrow Evt(c_2,t)$

Table 3: Formulas in the PSL model

interpretation, that is, the soft-truth values of the predicate Evt. The final detected event type t of candidate c is decided by the the equation:

$$t = \underset{t'}{\operatorname{argmax}}\, Evt(c,t') \quad (2)$$

5 Evaluations

In this section, we present the experiments and the results achieved. We first manually evaluate our novel PSL-based ED model on the FN corpus. Then, we also conduct automatic evaluations for the events detected from FN based on ACE corpus. Finally, we analyze possible mappings from frames/LUs to event types.

5.1 Data

We learned the basic ED model on ACE2005 dataset. In order to evaluate the learned model, we followed the evaluation of (Li et al., 2013): randomly selected 30 articles from different genres as the development set, and we subsequently conducted a test on a separate set of 40 ACE 2005 newswire documents. We used the remaining 529 articles as the training data set.

We apply our proposed PSL-based approach to detect events in FrameNet. Via collecting all exemplars annotated in FN, we totally obtain 154,484 sentences for detection.

5.2 Setup and Performance of Basic Model

We have presented the basic ED model in Section 3. Hyperparameters were tuned by grid search on the development data set. In our experiments, we set the size of the hidden layer to 300, the size of word embedding to 200, the batch size to 100 and the dropout rate to 0.5.

Table 4 shows the experimental results, from which we can see that the three-layer *ANN* model is surprisingly effective for event detection, which even yields competitive results compared with *Nguyen's CNN* and *Chen's DMCNN*. We believe the reason is that, compared with *CNN* and *DMCNN*,

Methods	Pre	Rec	F1
Nguyen's CNN (2015)	71.8	**66.4**	69.0
Chen's DMCNN (2015)	75.6	63.6	69.1
Liu's Approach (2016)	75.3	64.4	**69.4**
ANN (ours)	**79.5**	60.7	68.8
ANN-Random (ours)	**81.0**	49.5	61.5

Table 4: Performance of the basic ED model. *ANN* uses pre-trained word embeddings while *ANN-Random* uses randomly initialized embeddings.

ANN focuses on capturing lexical features which have been proved much more important than sentence features for the ED task by (Chen et al., 2015). Moreover, our basic model achieves much higher precision than state-of-the-art approaches (79.5% vs. 75.6%).

We also investigate the performance of the basic ED model without pre-trained word embeddings[6] (denoted by *ANN-Random*). The result shows that randomly initialized word embeddings decrease the F_1 score by 7.3 (61.5 vs. 68.8). The main reasons are: 1). ACE corpus only contains 599 articles, which are far insufficient to train good embeddings. 2). Words only existing in the test dataset always retain random embeddings.

5.3 Baselines

For comparison, we designed four baseline systems that utilize different hypotheses to detect events in FN.

(1) *ANN* is the first baseline, which directly uses a basic ED model learned on ACE training corpus to detect events in FN. This system does not apply any hypotheses between frames and events.

(2) *SameFrame (SF)* is the second baseline system, which applies *H1* over the results from *ANN*. For each frame, we introduce a score function $\phi(f,t)$ to estimate the probability that the frame f could be mapped to the event type t as follows:

$$\phi(f,t) = \frac{1}{||S_f||} \sum_{c \in S_f} I(c,t) \quad (3)$$

where S_f is the set of sentences under the frame f; $I(c,t)$ is an indicator function which is true if and only if *ANN* predicts the candidate c as an event of type t. Then for each frame f satisfying $\phi(f,t) > \alpha$, we mapped it to event type t, where α is a hyperparameter. Finally, all sentences under mapped frames are labeled as events. Note that, unlike the PSL-based approach which applies constraints as soft rules, this system utilizes *H1* as a hard constraint.

(3) *RelatedFrame (RF)* is the third baseline system, which applies *H2* over the results from *ANN*. For each frame f, we merge it and its related frames into a super frame, $f^{'}$. Similar with *SF*, a score function $\zeta(f^{'},t)$, which shares the same expression to equation 3, is introduced. For the merged frame satisfying $\zeta(f^{'},t) > \beta$, we mapped it to the event type t. Finally, all sentences under $f^{'}$ are labeled as events.

(4) *SameLU (SL)* is the last baseline, which applies the hypothesis *H3* over the results from *ANN*. Also, a score function $\psi(l,t)$ is introduced:

$$\psi(l,t) = \frac{1}{||S_l||} \sum_{c \in S_l} I(c,t) \quad (4)$$

where S_l is the set of sentences under the LU l. For each LU satisfying $\psi(l,t) > \gamma$, we mapped it to the event type t. Finally, all sentences under l are labeled as events.

5.4 Manual Evaluations

In this section, we manually evaluate the precision of the baseline systems and our proposed PSL-based approach. For fair comparison, we set α, β and γ to 0.32, 0.29 and 0.42 respectively to ensure they yield approximately the same amount of events as the first baseline system *ANN*. We tune the weights of formulas in PSL via grid search by using ACE development dataset. In details, we firstly detect events in FN under different configurations of formulas' weights and add them to ACE training dataset, respectively. Consequently, we obtain several different expanded training datasets. Then, we separately train a set of basic ED models based on each of these training datasets and evaluate them over the development corpus. Finally, the best weights are selected according to their performances on the development dataset. The weights of $f_1 \sim f_5$ used in this work are 100, 10, 100, 5, 5 and 1, respectively.

Manual Annotations

Firstly, we randomly select 200 samples from the results of each system. Each selected sample is a sentence with a highlighted trigger and a predicted event type. Figure 3 illustrates three samples. The first line of each sample is a sentence labeled with the trigger. The next line is the predicted event

[6]We thank the anonymous reviewer for this suggestion.

type of that sentence. Annotators are asked to assign one of two labels to each sample (annotating in the third line):

Y: the word highlighted in the given sentence indeed triggers an event of the predicted type.

N: the word highlighted in the given sentence does not trigger any event of the predicted type.

We can see that, it is very easy to annotate a sample for annotators, thus the annotated results are expected to be of high quality.

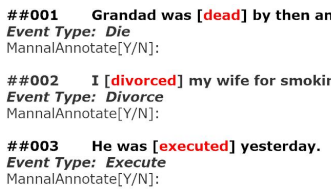


Figure 3: Examples of manual annotations.

To make the annotation more credible, each sample is independently annotated by three annotators[7] (including one of the authors and two of our colleagues who are familiar with ACE event task) and the final decision is made by voting.

Results

Table 5 shows the results of manual evaluations. Through the comparison of *ANN* and *SF*, we can see that the application of *H1* caused a loss of 5.5 point. It happens mainly because the performance of *SF* is very sensitive to the wrongly mapped frames. That is, if a frame is mismapped, then all sentences under it would be mislabeled as events. Thus, even a single mismapped frame could significantly hurt the performance. This result also proves that *H1* is inappropriate to be used as a hard constraint. As *H2* is only an extension of *H1*, *RF* performs similarly with *SF*. Moreover, *SL* obtains a gain of 2.0% improvement compared with *ANN*, which demonstrates that the "same LU" hypothesis is very useful. Finally, with all the hypotheses, the *PSL-based* approach achieves the best performance, which demonstrates that our hypotheses are useful and it is an effective way to jointly utilize them as soft constraints through PSL for event detection in FN.

5.5 Automatic Evaluations

To prepare for automatic evaluations, we respectively add the events detected from FN by each of the aforementioned five systems to ACE training corpus. Consequently, we obtain five expanded training datasets: *ACE-ANN-FN, ACE-SF-FN, ACE-RF-FN, ACE-SL-FN* and *ACE-PSL-FN*. Then, we separately train five basic ED models on each of these corpus and evaluate them on the ACE testing data set. This experiment is an indirect evaluation of the events detected from FN, which is based on the intuition that events with higher accuracy are expected to bring more improvements to the basic model.

[7]The inter-agreement rate is 86.1%

Methods		Precision (%)
Baselines	ANN	77.5
	SF	72.0
	RF	71.0
	SL	79.5
PSL-based Approach		**81.0**

Table 5: Results of manual evaluations.

Training Corpus	Pre	Rec	F_1
ACE-ANN-FN	77.2	63.5	69.7
ACE-SF-FN	73.2	64.1	68.4
ACE-RF-FN	72.6	63.9	68.0
ACE-SL-FN	77.5	64.3	70.3
ACE-PSL-FN	**77.6**	**65.2**	**70.7**

Table 6: Automatic evaluations of events from FN.

Table 6 presents the results where we measure precision, recall and F_1. Compared with *ACE-ANN-FN*, events from *SF* and *RF* hurt the performance. As analyzed in previous section, *SF* and *RF* yield quite a few false events, which dramatically hurt the accuracy. Moreover, *ACE-SL-FN* obtains a score of 70.3% in F_1 measure, which outperforms *ACE-ANN-FN*. This result illustrates the effectiveness of our "same LU" hypothesis. Finally and most importantly, consistent with the results of manual evaluations, *ACE-PSL-FN* performs the best, which further proves the effectiveness of our proposed approach for event detection in FN.

5.6 Improving Event Detection Using FN

Event detection generally suffers from data sparseness due to lack of labeled samples. In this section, we investigate the effects of alleviating the aforementioned problem by using the events detected from FN as extra training data. Our investigation is conducted by the comparison of two basic ED models, *ANN* and *ANN-FN*: the former is trained on ACE training corpus and the latter is trained on the new training corpus *ACE-PSL-FN* (introduced in the previous section), which contains 3,816 extra events detected from FN.

Methods	Pre	Rec	F_1
Nguyen's CNN(2015)	71.8	**66.4**	69.0
Chen's DMCNN(2015)	75.6	63.6	69.1
Liu's Approach(2016)	75.3	64.4	69.4
ANN (Ours)	**79.5**	60.7	68.8
ANN-FN (Ours)	77.6	65.2	**70.7**

Table 7: Effects of expanding training data using events automatically detected from FN.

Table 7 presents the experimental results. Compared with *ANN*, *ANN-FN* achieves a significant improvement of 1.9% in F_1 measure. It happens mainly because that the high accurate extra training data makes the model obtain a higher recall (from 60.7% to 65.2%) with less decrease of precision (from 79.5% to 77.6%). The result demonstrates the effectiveness of alleviating the data sparseness problem of ED by using events detected from FN. Moreover, compared with state-of-the-art methods, *ANN-FN* outperforms all of them with remarkable improvements (more than 1.3%).

5.7 Analysis of Frame-Event Mapping

In this section, we illustrate the details of mappings from frames to event types. The mapping pairs are obtained by computing the function ϕ (see Section 5.3) for each (frame, event-type) pair (f, t) based on the events detected by the PSL-based approach. Table 8 presents the top 10 mappings. We manually evaluate their quality by investigating: (1) whether the definition of each frame is compatible with its mapped event type; (2) whether exemplars annotated for each frame actually express events of its mapped event type.

For the first issue, we manually compare the definitions of each mapped pair. Except `Relational_nat_features`[8], definitions of all the mapped pairs are compatible. For the second issue, we randomly sample 20 exemplars (if possible) from each frame and manually annotate them. Except the above frame and `Invading`, exemplars of the remaining frames all express the right events. The only exemplar of `Invading` failing to express its mapped event is as follows: "*The **invasion** of China by western culture has had a number of far-reaching effects on Confucianism.*" ACE requires an `Attack` event to be a physical act, while the invasion of culture is unphysical. Thus, the above sentence does not express an `Attack` event. To sum up, the quality of our mappings is good, which demonstrates that the hypothesis *H1* is basically true.

[8]The full name is Relational_natural_relations in FN.

Frame	Event	N_e/$\|\|S_f\|\|$	ϕ
Hit_target	Attack	2/2	1.0
Relational nat_features	Meet	1/1	1.0
Invading	Attack	120/121	0.99
Fining	Fine	26/27	0.96
Being_born	Be-Born	32/36	0.88
Rape	Attack	104/125	0.83
Sentencing	Sentence	57/70	0.81
Attack	Attack	99/129	0.77
Quitting	End-Position	102/137	0.74
Notification of_charges	Charge-Indict	73/103	0.71

Table 8: Top 10 mappings from frames to event types. N_e is the number of exemplars detected as events; $||S_f||$ and ϕ hold the same meanings as mentioned in Section 5.3.

5.8 Analysis of LU-Event Mapping

This section illustrates the details of mappings from LUs to event types. The mapping pairs are obtained by computing the function ψ (see Section 5.3). Table 9 presents the top 10 mappings. In FN, each LU belongs to a frame. In table 9, we omit the frame of each LU because of space limitation[9].

LU	Event	N_e/$\|\|S_l\|\|$	ψ
gunfight.n	Attack	14/14	1.0
injure.v	Injure	14/14	1.0
divorce.n	Divorce	11/11	1.0
decapitation.n	Die	5/5	1.0
trial.n	Trial-Hearing	25/25	1.0
assault.v	Attack	21/21	1.0
fight.v	Attack	12/12	1.0
arrest.n	Arrest-Jail	38/38	1.0
divorce.v	Divorce	35/35	1.0
shoot.v	Attack	2/2	1.0

Table 9: Top 10 mappings from LUs to event types. N_e is the number of exemplars detected as events; $||S_l||$ and ψ hold the same meanings as mentioned in Section 5.3.

To investigate the mapping quality, we manu-

[9]Their frames separately are Hostile_encounter, Cause_harm, Forming_relationships, Killing, Trial, Attack, Quarreling, Arrest, Forming_relationships and Hit_target.

ally annotate the exemplars under these LUs. The result shows that all exemplars are rightly mapped. These mappings are quite good. We believe the reason is that an LU is hardly ambiguous due to its high specificity, which is not only specified by a lemma but also by a frame and a part of speech tag. Table 9 only presents the top 10 mappings. In fact, we obtain 54 mappings in total with $\psi = 1.0$. We released all the detected events and mapping results for further use by the NLP community.

6 Conclusions and Future Work

Motivated by the high similarity between frames and events, we conduct this work to study their relations. The key of this research is to detect events in FN. To solve this problem, we proposed a PSL-based global inference approach based on three hypotheses between frames and events. For evaluation, we first conduct manual evaluations on events detected from FN. The results reveal that our hypotheses are very useful and it is an effective way to jointly utilize them as soft rules through PSL. In addition, we also perform automatic evaluations. The results further demonstrate the effectiveness of our proposed approach for detecting events in FN. Furthermore, based on the detected results, we analyze the mappings from frames/LUs to event types. Finally, we alleviate the data sparseness problem of ED by using events detected from FN as extra training data. Consequently, we obtain a remarkable improvement and achieve a new state-of-the-art result for the ED task.

Event detection is only a component of the overall task of event extraction, which also includes event role detection. In the future, we will extend this work to the complete event extraction task. Furthermore, event schemas in ACE are quite coarse. For example, all kinds of violent acts, such as street fights and wars, are treated as a single event type `Attack`. We plan to refine the event schemas by the finer-grained frames defined in FN (i.e. `Attack` may be divided into *`Terrorism`*, *`Invading`*, etc.).

Acknowledgements

This work was supported by the Natural Science Foundation of China (No. 61533018), the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61272332). And this work was also supported by Google through focused research awards program.

References

David Ahn. 2006. The stages of event extraction. In *Proceedings of the Workshop on Annotating and Reasoning about Time and Events*, pages 1–8.

Stephen Bach, Bert Huang, Ben London, and Lise Getoor. 2013. Hinge-loss markov random fields: convex inference for structured prediction. In *Proceedings of 29th Annual Meeting of the Association for Uncertainty in Artificial Inteligence*, pages 1–10.

Collin F Baker, Charles J Fillmore, and John B Lowe. 1998. The berkeley framenet project. In *Proceedings of 17th Annual Meeting of the Association for Computational Linguistics*, pages 86–90.

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

Aljoscha Burchardt, Marco Pennacchiotti, Stefan Thater, and Manfred Pinkal. 2009. Assessing the impact of frame semantics on textual entailment. *Natural Language Engineering*, 15(04):527–550.

Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. 2015. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics*, pages 167–176.

Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. 2014. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660.

Charles J Fillmore, Christopher R Johnson, and Miriam RL Petruck. 2003. Background to framenet. *International Journal of Lexicography*, 16(3):235–250.

Ana-Maria Giuglea and Alessandro Moschitti. 2006. Shallow semantic parsing based on framenet, verbnet and propbank. *European Conference on Artificial Intelligence*, 141:563–567.

Prashant Gupta and Heng Ji. 2009. Predicting unknown time arguments based on cross-event propagation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 369–372.

Martin T Hagan, Howard B Demuth, Mark H Beale, et al. 1996. *Neural network design*. Pws Pub. Boston.

Karl Moritz Hermann, Dipanjan Das, Jason Weston, and Kuzman Ganchev. 2014. Semantic frame identification with distributed word representations. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1448–1458.

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

Yu Hong, Jianfeng Zhang, Bin Ma, Jianmin Yao, Guodong Zhou, and Qiaoming Zhu. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of 49th Annual Meeting of the Association for Computational Linguistics*, pages 1127–1136.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics*, pages 254–262.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751.

Angelika Kimmig, Stephen Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2012. A short introduction to probabilistic soft logic. In *Proceedings of the NIPS Workshop on Probabilistic Programming: Foundations and Applications*, pages 1–4.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of 51st Annual Meeting of the Association for Computational Linguistics*, pages 73–82.

Shasha Liao and Ralph Grishman. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of 48th Annual Meeting of the Association for Computational Linguistics*, pages 789–797.

Shulin Liu, Kang Liu, Shizhu He, and Jun Zhao. 2016. A probabilistic soft logic based approach to exploiting latent and global information in event classification. In *Proceedings of the thirtieth AAAI Conference on Artificail Intelligence*.

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

George A. Miller. 1995. Wordnet: a lexical database for english. *Communications of the Acm*, 38(11):39–41.

Srini Narayanan and Sanda Harabagiu. 2004. Question answering based on semantic structures. *International Conference on Computational Linguistics*.

Thien Huu Nguyen and Ralph Grishman. 2015. Event detection and domain adaptation with convolutional neural networks. In *Proceedings of 53rd Annual Meeting of the Association for Computational Linguistics*, pages 365–371.

Sebastian Padó and Mirella Lapata. 2005. Cross-linguistic projection of role-semantic information. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 859–866. Association for Computational Linguistics.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, pages 107–136.

Dan Shen and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 12–21.

Cynthia A Thompson, Roger Levy, and Christopher D Manning. 2003. A generative model for semantic role labeling. *European Conference on Machine Learning*, pages 397–408.

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of 14th Annual Meeting of the Association for Computational Linguistics*, pages 189–196.

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Learning To Use Formulas To Solve Simple Arithmetic Problems

Arindam Mitra
Arizona State University
amitra7@asu.edu

Chitta Baral
Arizona State University
chitta@asu.edu

Abstract

Solving simple arithmetic word problems is one of the challenges in Natural Language Understanding. This paper presents a novel method to learn to use formulas to solve simple arithmetic word problems. Our system, analyzes each of the sentences to identify the variables and their attributes; and automatically maps this information into a higher level representation. It then uses that representation to recognize the presence of a formula along with its associated variables. An equation is then generated from the formal description of the formula. In the training phase, it learns to score the <formula, variables> pair from the systematically generated higher level representation. It is able to solve 86.07% of the problems in a corpus of standard primary school test questions and beats the state-of-the-art by a margin of 8.07%.

1 Introduction

Developing algorithms to solve math word problems (Table 1) has been an interest of NLP researchers for a long time (Feigenbaum and Feldman, 1963). It is an interesting topic of study from the point of view of natural language understanding and reasoning for several reasons. First, it incorporates rigorous standards of accurate comprehension. Second, we know of a good representation to solve the word problems, namely algebraic equations. Finally, the evaluation is straightforward and the problems can be collected easily.

In the recent years several challenges have been proposed for natural language understanding. This includes the Winograd Schema challenge for commonsense reasoning (Levesque, 2011), Story Comprehension Challenge (Richardson et al., 2013), Facebook bAbl task (Weston et al., 2015), Semantic Textual Similarity (Agirre et al., 2012) and Textual Entailment (Bowman et al., 2015; Dagan et al., 2010). The study of word math problems is also an important problem as quantitative reasoning is inextricably related to human life. Clark & Etzioni (Clark, 2015; Clark and Etzioni, 2016) discuss various properties of math word (and science) problems emphasizing elementary school science and math tests as a driver for AI.

Researchers at Allen AI Institute have published two standard datasets as part of the Project Euclid[1] for future endeavors in this regard. One of them contains simple addition-subtraction arithmetic problems (Hosseini et al., 2014) and the other contains general arithmetic problems (Koncel-Kedziorski et al., 2015). In this research, we focus on the former one, namely the *AddSub* dataset.

Dan grew 42 turnips and 38 cantelopes . Jessica grew 47 turnips . How many turnips did they grow in total ?	
Formula	**Associated variables**
part-whole	*whole:* x, *parts:* $\{42, 47\}$
Equation	$x = 42 + 47$

Table 1: Solving a word problem using *part-whole*

Broadly speaking, common to the existing approaches (Kushman et al., 2014; Hosseini et al., 2014; Zhou et al., 2015; Shi et al., 2015; Roy and Roth, 2015) is the task of grounding, that takes as input a word problem in the natural language and represents it in a formal language, such as, a system of equations, expression trees or states (Hosseini et al., 2014), from which the answer can be easily computed. In this work, we divide this task of grounding into two parts as follows:

[1] http://allenai.org/euclid.html

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2144–2153,
Berlin, Germany, August 7-12, 2016.

In the first step, the system learns to connect the assertions in a word problem to abstract mathematical concepts or *formulas*. In the second step, it maps that formula into an algebraic equation. Examples of such formulas in the arithmetic domain includes *part whole* which says, 'the whole is equal to the sum of its parts', or the *Unitary Method* that is used to solve problems like '*A man walks seven miles in two hours. What is his average speed?*'.

Consider the problem in Table 1. If the system can determine it is a 'part whole' problem where the unknown quantity X plays the role of *whole* and its *parts* are 42 and 47, it can easily express the relation as $X = 42 + 47$. The translation of a formula to an equation requires only the knowledge of the formula and can be formally encoded. Thus, we are interested in the question, 'how can an agent learn to apply the formulas for the word problems?' Solving a word problem in general, requires several such applications in series or parallel, generating multiple equations. However, in this research, we restrict the problems to be of a single equation which requires only one application.

Our system currently considers three mathematical concepts: 1) the concept of *part whole*, 2) the concept of *change* and 3) the concept of *comparison*. These concepts are sufficient to solve the arithmetic word problems in *AddSub*. Table 2 illustrates each of these three concepts with examples. The *part whole* problems deal with the part whole relationships and ask for either the part or the whole. The *change* problems make use of the relationship between the new value of a quantity and its original value after the occurrence of a series of increase or decrease. The question then asks for either the initial value of the quantity or the final value of the quantity or the change. In case of *comparison problems*, the equation can be visualized as a comparison between two quantities and the question typically looks for either the larger quantity or the smaller quantity or the difference. While the equations are simple, the problems describe a wide variety of scenarios and the system needs to make sense of multiple sentences without a priori restrictions on the syntax or the vocabulary to solve the problem.

Training has been done in a supervised fashion. For each example problem, we specify the formula that should be applied to generate the appropriate equation and the relevant variables. The system then learns to apply the formulas for new problems. It achieves an accuracy of 86.07% on the *AddSub* corpus containing 395 word arithmetic problems with a margin of 8.07% with the current state-of-the-art (Roy and Roth, 2015).

Our contributions are three-fold: (a) We model the application of a formula and present a novel method to learn to apply a formula; (b) We annotate the publicly available *AddSub* corpus with the

Change
RESULT UNKNOWN
Mary had 18 baseball cards , and 8 were torn . Fred gave Mary 26 new baseball cards . Mary bought 40 baseball cards . How many baseball cards does Mary have now ?
CHANGE UNKNOWN
There were 28 bales of hay in the barn . Tim stacked bales in the barn today . There are now 54 bales of hay in the barn . How many bales did he store in the barn ?
START UNKNOWN
Sam 's dog had puppies and 8 had spots . He gave 2 to his friends . He now has 6 puppies . How many puppies did he have to start with?
Part Whole
TOTAL SET UNKNOWN
Tom went to 4 hockey games this year , but missed 7 . He went to 9 games last year . How many hockey games did Tom go to in all ?
PART UNKNOWN
Sara 's high school played 12 basketball games this year . The team won most of their games . They were defeated during 4 games . How many games did they win ?
Comparision
DIFFERENCE UNKNOWN
Last year , egg producers in Douglas County produced 1416 eggs . This year , those same farms produced 4636 eggs . How many more eggs did the farms produce this year ?
LARGE QUANTITY UNKNOWN
Bill has 9 marbles. Jim has 7 more marbles than Bill. How many marbles does Jim have?
SMALL QUANTITY UNKNOWN
Bill has 9 marbles. He has 7 more marbles than Jim. How many marbles does Jim have?

Table 2: Examples of Add-Sub Word Problems

correct formula and its associated variables; and (c) We make the code publicly available. [2]

The rest of the paper is organized as follows. In section 2, we formally define the problem and describe our learning algorithm. In section 3, we define our feature function. In section 4, we discuss related works. Section 5 provides a detailed description of the experimental evaluation. Finally, we conclude the paper in section 6.

2 Problem Formulation

A single equation word arithmetic problem P is a sequence of k words $\langle w_1, ..., w_k \rangle$ and contains a set of *variables* $\mathbb{V}_P = \{v_0, v_1, ..., v_{n-1}, x\}$ where $v_0, v_1, ..., v_{n-1}$ are *numbers* in P and x is the *unknown* whose value is the answer we seek (Koncel-Kedziorski et al., 2015). Let $\mathcal{P}_{addsub}$ be the set of all such problems, where each problem $P \in \mathcal{P}_{addsub}$ can be solved by a evaluating a valid mathematical equation E formed by combining the elements of $\mathbb{V}_P$ and the binary operators from $\mathcal{O} = \{+, -\}$.

We assume that each target equation E of $P \in \mathcal{P}_{addsub}$ is generated by applying one of the possible mathematical formulas from $\mathcal{C} = \{C_{partwhole}, C_{change}, C_{comparision}\}$. Let $\mathcal{P}^1_{addsub} \subseteq \mathcal{P}_{addsub}$ be the set of all problems where the target equation E can be generated by a single application of one of the possible formulas from $\mathcal{C}$. The goal is then to find the correct application of a formula for the problem $P \in \mathcal{P}^1_{addsub}$.

2.1 Modelling Formulas And their Applications

We model each formula as a template that has predefined slots and can be mapped to an equation when the slots are filled with variables. Application of a formula $C \in \mathcal{C}$ to the problem P, is then defined as the instantiation of the template by a subset of $\mathbb{V}_P$ that contains the *unknown*.

Part Whole The concept of *part whole* has two slots, one for the *whole* that accepts a single variable and the other for its *parts* that accepts a set of variables of size at least two. If the value of the whole is w and the value of the parts are $p_1, p_2, ..., p_m$, then that application is mapped to the equation, $w = p_1 + p_2 + ... + p_m$, denoting that *whole is equal to the sum of its parts*.

[2]The code and data is publicly available at *https://github.com/ari9dam/MathStudent*.

Change The *change* concept has four slots, namely *start, end, gains, losses* which respectively denote the original value of a variable, the final value of that variable, and the set of increments and decrements that happen to the original value of the variable. The *start* slot can be empty; in that case it is assumed to be 0. For example, consider the problem, '*Joan found 70 seashells on the beach . she gave Sam some of her seashells. She has 27 seashell . How many seashells did she give to Sam?*'. In this case, our assumption is that before finding the 70 seashells *Joan* had an empty hand. Given an instantiation of *change* concept the equation is generated as follows:

$$val_{start} + \sum_{g \in gains} val_g = \sum_{l \in losses} val_l + val_{end}$$

Comparision The *comparision* concept has three slots namely the *large* quantity, the *small* quantity and their *difference*. An instantiation of the *comparision* concept is mapped to the following equation: $large = small + difference$.

2.2 The Space of Possible Applications

Consider the problem in Table 1. Even though the correct application is an instance of *part whole* formula with *whole* = x and the *parts* being $\{42, 47\}$, there are many other possible applications, such as, *partWhole(whole=47, parts=x,42)*, *change(start=47, losses={x}, gains={}, end = 42)*, *comparison(large=47, small=x, difference=42)*. Note that, *comparison(large=47, small=38, difference=42)* is not a valid application since none of the associated variables is an *unknown*. Let A_P be the set of all possible applications to the problem P. The following *lemma* characterizes the size of A_P as a function of the number of variables in P.

Lemma 2.2.1. *Let $P \in \mathcal{P}^1_{addsub}$ be an arithmetic word problem with n variables ($|\mathbb{V}_P| = n$), then the following are true:*

1. *The number of possible applications of part whole formula to the problem P, $N_{partwhole}$ is $(n+1)2^{n-2} + 1$.*
2. *The number of possible applications of change formula to the problem P, N_{change} is $3^{n-3}(2n^2 + 6n + 1) - 2n + 1$.*
3. *The number of possible applications of comparison formula to the problem P, $N_{comparison}$ is $3(n-1)(n-2)$.*

4. *The number of all possible applications to the problem P is $N_{partwhole} + N_{change} + N_{comparison}$.*

Proof of lemma 2.2.1 is provided in the Appendix. The total number of applications for problems having 3, 6, 7, 8 number of variables are 47, 3,105, 11,755, 43,699 respectively. Addition-Subtraction arithmetic problems hardly contain more than 6 variables. So, the number of possible applications is not intractable in practice.

The total number of applications increases rapidly mainly due to the *change* concept. Since, the template involves two sets, there is a 3^{n-3} factor present in the formula of N_{change}. However, any application of *change* concept with *gains* and *losses* slots containing a collection of variables can be broken down into multiple instances of *change* concept where the *gains* and *losses* slots accepts only a single variable by introducing more intermediate *unknown* variables. Since, for any formula that does not have a slot that accepts a set, the number of applications is polynomial in the number of variables, there is a possibility to reduce the application space. We plan to explore this possibility in our future work. For the *part whole* concept, even though there is a exponential term involved, it is practically tractable (for $n = 10$, $N_{partwhole} = 2,817$). In practice, we believe that there will hardly be any *part whole* application involving more than 10 variables. For formulas that are used for other categories of word math problems (algebraic or arithmetic), such as the *unitary method*, formulas for *ratio*, *percentage*, *time-distance* and *rate of interest*, none of them have any slot that accepts sets of variables. Thus, further increase in the space of possible applications will be polynomial.

2.3 Probabilistic Model

For each problem P there are different possible applications $y \in A_P$, however not all of them are meaningful. To capture the semantics of the word problem to discriminate between competing applications we use the log-linear model, which has a feature function ϕ and parameter vector $\theta \in \mathbb{R}^d$. The feature function $\phi : H \rightarrow \mathbb{R}^d$ takes as input a problem P and a possible application y and maps it to a d-dimensional real vector (*feature vector*) that aims to capture the important information required to discriminate between competing applications. Here, the set H is defined as $\{(P, y) : P \in \mathcal{P}^1_{addsub} \wedge y \in A_P\}$, to accommodate the dependency of the possible applications on the problem instance. Given the definition of the feature function ϕ and the parameter vector θ, the probability of an application y given a problem P is defined as,

$$p(y|P;\theta) = \frac{e^{\theta.\phi(P,y)}}{\sum_{y' \in A_P} e^{\theta.\phi(P,y')}}$$

Here, . denotes dot product. Section 3 defines the feature function. Assuming that the parameter θ is known, the function f that computes the correct application is defined as,

$$f(P) = \arg\max_{y \in A_P} p(y|P;\theta)$$

2.4 Parameter Estimation

To learn the function f, we need to estimate the parameter vector θ. For that, we assume access to n training examples, $\{P_i, y_i^* : i = 1 \ldots n\}$, each containing a word problem P_i and the correct application y_i^* for the problem P_i. We estimate θ by minimizing the negative of the conditional log-likelihood of the data:

$$\begin{aligned} O(\theta) &= -\sum_{i=1}^{n} log\ p(y_i^*|P_i;\theta) \\ &= -\sum_{i=1}^{n} [\theta.\phi(P_i, y_i^*) - log \sum_{y \in A_{P_i}} e^{\theta.\phi(P_i,y)}] \end{aligned}$$

We use stochastic gradient descent to optimize the parameters. The gradient of the objective function is given by:

$$\begin{aligned} \frac{\nabla O}{\nabla \theta} = -\sum_{i=1}^{n} [\phi(P_i, y_i^*) - \\ \sum_{y \in A_{P_i}} p(y|P_i;\theta) \times \phi(P_i, y)] \end{aligned} \quad (1)$$

Note that, even though the space of possible applications vary with the problem P_i, the gradient for the example containing the problem P_i can be easily computed.

3 Feature Function ϕ

A formula captures the relationship between variables in a compact way which is sufficient to generate an appropriate equation. In a word problem, those relations are hidden in the assertions

of the story. The goal of the feature function is thus to gather enough information from the story so that underlying mathematical relation between the variables can be discovered. The feature function thus needs to be aware of the mathematical relations so that it knows what information it needs to find. It should also be "familiar" with the word problem language so that it can extract the information from the text. In this research, the feature function has access to machine readable dictionaries such as WordNet (Miller, 1995), ConceptNet (Liu and Singh, 2004) which captures inter word relationships such as hypernymy, synonymy, antonymy etc, and syntactic and dependency parsers that help to extract the subject, verb, object, preposition and temporal information from the sentences in the text. Given these resources, the feature function first computes a list of attributes for each variable. Then, for each application y it uses that information, to compute if some aspects of the expected relationship described in y is satisfied by the variables in y.

Let the first b dimensions of the feature vector contain part whole related features, the next c dimensions are for change related features and the remaining d features are for comparison concept. Then the feature vector for a problem P and an application of a formula y is computed in the following way:

Data: A word problem P, an application y
Result: d-dimensional feature vector, fv
Initialize $fv := 0$
if *y is instance of part whole* **then**
  compute $fv[1 : b]$
end
if *y is instance of change* **then**
  compute $fv[b + 1 : b + c]$
end
if *y is instance of comparision* **then**
  compute $fv[b + c + 1 : b + c + d]$
end

Algorithm 1: Skeleton of the feature function ϕ

The rest of the section is organized as follows. We first describe the attributes of the variables that are computed from the text. Then, we define a list of boolean variables which computes semantic relations between the attributes of each pair of variables. Finally, we present the complete definition of the feature function using the description of the attributes and the boolean variables.

3.1 Attributes of Variables

For each occurrence of a number in the text a variable is created with the attribute *value* referring to that numeric value. An unknown variable is created corresponding to the question. A special attribute *type* denotes the kind of object the variable refers to. Table 3 shows several examples of the *type* attribute. It plays an important role in identifying irrelevant numbers while answering the question.

Text	Type
John had **70** seashells	seashells
70 seashells and **8** were broken	seashells
61 male and 78 female salmon	male, salmon
35 pears and 27 apples	pear

Table 3: Example of *type* for highlighted variables.

The other attributes of a variable captures its linguistic context to surrogate the meaning of the variable. This includes the *verb* attribute i.e. the verb attached to the variable, and attributes corresponding to Stanford dependency relations (De Marneffe and Manning, 2008), such as *nsubj*, *tmod*, *prep in*, that spans from either the words in associated *verb* or words in the *type*. These attributes were computed using Stanford Core NLP (Manning et al., 2014). For the sentence, "John found 70 seashells on the beach." the attributes of the variable are the following: { **value** : {70}, **verb :** {*found*} , **nsubj :** {*John*}, **prep on :** {*beach* }}.

3.2 Cross Attribute Relations

Once the variables are created and their attributes are extracted, our system computes a set of boolean variables, each denoting whether the attribute a_1 of the variable v_1 has the same value as the attribute a_2 of the variable v_2. The value of each attribute is a set of words, consequently set equality is used to calculate attribute equality. Two words are considered equal if their lemma matches.

Four more boolean variables are computed for each pair of variables based on the attribute *type* and they are defined as follows:

subType: Variable v_1 is a $subType$ of variable v_2 if $v_2.type \subset v_1.type$ or their type consists of a single word and there exists the *IsA* relation between them in ConceptNet (Speer and Havasi, 2013; Liu and Singh, 2004).

disjointType is true if $v_1.type \bigcap v_2.type = \phi$

intersectingType is true if v_1 is neither a $subType$ of v_2 nor is $disjointType$ nor $equal$.

We further compute some more variables by utilizing several relations that exist between words:

antonym: For every pair of variables v_1 and v_2, we compute an antonym variable that is true if there exists a pair of word in $(v_1.verb \bigcup v1.adj) \times (v_2.verb \bigcup v2.adj)$ that are antonym to each other in WordNet irrespective of their part of speech tag.

relatedVerbs: The verbs of two variables are related if there exists a *RelatedTo* relations in ConceptNet between them.

subjConsume: The *nsubj* of v_1 consumes the *nsubj* of v_2 if the formers refers to a group and the latter is a part of that group. For example, in the problem, '*Joan grew 29 carrots and 14 watermelons . Jessica grew 11 carrots . How many carrots did they grow in all ?*', the *nsubj* of the *unknown* variable consumes others. This is computed using Stanford co-reference resolution. For the situation where there is a variable with *nsubj* as 'they' and it does not refer to any entity, the *subjConsume* variable is assumed to be implicitly true for any variable having a *nsubj* of type *person*.

3.3 Features: Part Whole

The *part whole* features look for some combinations of the boolean variables and the presence of some cue words (e.g. 'all') in the attribute list. These features capture the underlying reasonings that can affect the decision of applying a *part whole* concept. We describe the conditions which when satisfied activate the features. If active, the value of a feature is the number of variables associated with the application y and 0 otherwise. This is also true for *change* and *comparision* features also. *Part whole* features are computed only when the y is an instance of the formula *part whole*. The same applies for *change* and *comparision* features.

Generic Word Cue This feature is activated if $y.whole$ has a word in its attributes that belongs to the "total words set" containing the followings words "all", "total", "overall", "altogether", "together" and "combine"; and none of the variables in *parts* are marked with these words.

ISA Type Cue is active if all the part variables are *subType* of the whole.

Type-Verb Cue is active if the type and verb attributes of v_{whole} matches that of all the variables in the part slot of y.

Type-Individual Group Cue is active if the variable v_{whole} *subjConsume* each part variable v_p in y and their type matches.

Type-Verb-Tmod Cue is active if the variable in the slot *whole* is the *unknown* and for each part variable v_p their *verb*, *type* and *tmod* (time modifier of the verb) attributes match.

Type-SubType-Verb Cue is active if the variable in the slot *whole* is either the *unknown* or marked with a word in "total words set" and for all parts v_p, their verb matches and one of the type or *subType* boolean variable is true.

Type-SubType-Related Verb Cue is similar to *Type-SubType-Verb Cue* however relaxes the *verb* match conditions to *related verb* match. This is helpful in problems like '*Mary went to the mall. She spent $ 13.04 on a shirt and $ 12.27 on a jacket . She went to 2 shops . In total , how much money did Mary spend on clothing ? '*.

Type-Loose Verb Cue ConceptNet does not contain all relations between verbs. For example, according to ConceptNet 'buy' and 'spend' are related however there is no relation in ConceptNet between 'purchase' and 'spend'. To handle these situations, we use this feature which is similar to the previous one. The difference is that it assumes that the verbs of part-whole variable pairs are related if all verbs associated with the parts are same, even though there is no relation in ConceptNet.

Type-Verb-Prep Cue is active if type and verb matches. The whole does not have a "preposition" but parts have and they are different.

Other Cues There are also features that add *nsubj* match criteria to the above ones. The *prior* feature for *part whole* is that the whole if not unknown, is smaller than the sum of the parts. There is one more feature that is active if the two part variables are antonym to each other; one of type or subType should be true.

3.4 Features: Change

The *change* features are computed from a set of 10 simple indicator variables, which are computed in the following way:

Start Cue is active if the verb associated with the variable in *start* slot has one of the following possessive verbs : {'call for', 'be', 'contain', 'remain', 'want', 'has', 'have', 'hold', ...}; the type and *nsubj* of start variable match with the end variable and the tense of the end does not precede the start. The list of 'possessive verbs' is automatically constructed by adding all the verbs associated with the *start* and the *end* slot variables in annotated corpus.

Start Explicit Cue is active if one of following words, "started with", "initially", "begining", "originally" appear in the context of the start variable and the type of start and end variables match.

Start prior is active if the verb associated with the variable in *start* slot is a member of the set 'possessive verbs' and the variable appears in first sentence.

Start Default Cue is active if the start variable has a "possessive verb" with past tense.

End Cue is active if the verb associated with the variable in slot end has a possessive verb with the tense of the verb not preceding the tense of the start, in case the start is not missing. The type and nsubj should match with either the start or the gains in case the start is missing.

End Prior is true if v_{end} has a possessive verb and an unknown quantity and at least one of v_{end} or v_{start} does not have a *nsubj* attribute.

Gain Cue is active if for all variables in the *gains* slot, the type matches with either v_{end} or v_{start} and one of the following is true: 1) the nsubj of the variable matches with v_{end} or v_{start} and the verb implies gain (such as 'find') and 2) the nsubj of the variable does not match with v_{end} or v_{start} and the verb implies losing (e.g. spend). The set of gain and loss verbs are collected from the annotated corpus by following the above procedure.

Gain Prior is true if the problem contains only three variables, with $v_{start} < v_{end}$ and the only variable in the *gain* slot, associated with non-possessive verb is the unknown.

Loss Cue & Loss prior are designed in a fashion similar to the Gain cue and Gain Prior.

Let us say bad_{gains} denotes that none of the gain prior or gain cue is active even though the gain slot is not empty. bad_{losses} is defined similarly and let $bad = bad_{gains} \vee bad_{losses}$. Then the change features are computed from these boolean indicators using logical operators *and, or, not*. Table4 shows some of the change features.

$!bad \wedge gain_{cue} \wedge start_{default} \wedge end_{cue}$
$!bad \wedge !gain_{cue} \wedge loss_{cue} \wedge start_{default} \wedge end_{cue}$
$!bad \wedge (gain_{cue} \vee loss_{cue}) \wedge start_{cue} \wedge !start_{default} \wedge end_{cue}$
$!bad \wedge (gain_{cue} \vee loss_{cue}) \wedge start_{explicit} \wedge !start_{default} \wedge end_{cue}$
$!bad \wedge (gain_{cue} \vee loss_{cue}) \wedge start_{prior} \wedge (end_{cue} \| \| end_{prior})$
$!bad \wedge (gain_{cue} \vee loss_{cue}) \wedge (start_{prior} \vee start_{cue}) \wedge !start_{default} \wedge end_{prior}$

Table 4: Activation criteria of some *change* related features.

3.5 Features: Comparison

The features for the "compare" concept are relatively straight forward.

Difference Unknown Que If the application y states that the unknown quantity is the difference between the larger and smaller quantity, it is natural to see if the variable in the *difference* slot is marked with a comparative adjective or comparative adverb. The prior is that the value of the larger quantity must be bigger than the small one. Another two features add the type and subject matching criteria along with the previous ones.

Large & Small Unknown Que These features can be active only when the variable in the *large* or *small* slot is unknown. To detect if the referent is bigger or smaller, it is important to know the meaning of the comparative words such as 'less' and 'longer'. Since, the corpus contains only *33* comparison problems we collect these comparative words from web which are then divided into two categories. With these categories, the features are designed in a fashion similar to *change* features that looks for type, subject matches.

3.6 Handling Arbitrary Number of Variables

This approach can handle arbitrary number of variables. To see that consider the problem, 'Sally found 9 seashells , Tom found 7 seashells , and Jessica found 5 seashells on the beach . How many seashells did they find together ?'. Let us say that feature vector contains only the '*Type-Individual Group Cue*' feature and the weight

of that feature is 1. Consider the two following applications: y_1 = *partWhole(x,{9,7})* and y_2 = *partWhole(x,{9,7, 5})*. For both y_1 and y_2 the '*Type-Individual Group Cue*' feature is active since the subject of the unknown x refers to a group that contains the subject of all part variables in y_1 and y_2 and their types match. However, as mentioned in section 3.3, when active, the value of a feature is the number of variables associated with the application. Thus $\frac{p(y_2;P,\theta)}{p(y_1;P,\theta)} = \frac{e^4}{e^3} = e$. Thus, y_2 is more probable than y_1.

4 Related Works

Researchers in early years have studied math word problems in a constrained domain by either limiting the input sentences to a fixed set of patterns (Bobrow, 1964b; Bobrow, 1964a; Hinsley et al., 1977) or by directly operating on a propositional representation instead of a natural language text (Kintsch and Greeno, 1985; Fletcher, 1985). Mukherjee and Garain (2008) survey these works.

Among the recent algorithms, the most general ones are the work in (Kushman et al., 2014; Zhou et al., 2015) . Both algorithms try to map a word math problem to a 'system template' that contains a set of 'equation templates' such as $ax + by = c$. These 'system templates' are collected from the training data. They implicitly assume that these templates will reoccur in the new examples which is a major drawback of these algorithms. Also, Koncel-Kedziorski et al. (2015) show that the work of Kushman et al. (2014) heavily relies on the overlap between train and test data and when this overlap is reduced the system performs poorly.

Work of (Koncel-Kedziorski et al., 2015; Roy and Roth, 2015) on the other hand try to map the math word problem to an expression tree. Even though, these algorithms can handle all the four arithmetic operators they cannot solve problems that require more than one equation. Moreover, experiments show that our system is much more robust to diversity in the problem types between training and test data for the problems it handles.

The system ARIS in (Hosseini et al., 2014) solves the addition-subtraction problems by categorizing the verbs into seven categories such as 'positive transfer', 'loss' etc. It represents the information in a problem as a state and then updates the state according to the category of a verb as the story progresses. Both ARIS and our system share the property that they give some explanation behind the equation they create. However, the verb categorization approach of ARIS can only solve a subset of addition-subtraction problems (see error analysis in (Hosseini et al., 2014)); whereas the usage of formulas to model the word problem world, gives our system the ability to accommodate other math word problems as well.

5 Experimental Evaluation

5.1 Dataset

The *AddSub* dataset consist of a total of 395 addition-subtraction arithmetic problems for third, fourth, and fifth graders. The dataset is divided into three diverse set MA1, MA2, IXL containing 134, 140 and 121 problems respectively. As mentioned in (Hosseini et al., 2014), the problems in MA2 have more irrelevant information compared to the other two datasets, and IXL includes more information gaps.

5.2 Result

Hosseini et al. (2014) evaluate their system using 3-fold cross validation. We follow that same procedure. Table 5 shows the accuracy of our system on each dataset (when trained on the other two datasets). Table 6 shows the distribution of the *part whole, change, comparison* problems and the accuracy on recognizing the correct formula.

	MA1	IXL	MA2	Avg
ARIS	83.6	75.0	74.4	77.7
KAZB	89.6	51.1	51.2	64.0
ALGES	-	-	-	77.0
Roy & Roth	-	-	-	78.0
Majority	45.5	71.4	23.7	48.9
Our System	96.27	82.14	79.33	86.07

Table 5: Comparision with ARIS, KAZB (Kushman et al., 2014), ALGES (Koncel-Kedziorski et al., 2015) and the state of the art Roy & Roth on the accuracy of solving arithmetic problems.

As we can see in Table 6 only IXL contains problems of type 'comparison'. So, to study the accuracy in detecting the *compare* formula we uniformly distribute the 33 examples over the 3 datasets. Doing that results in only two errors in the recognition of a *compare* formula and also increases the overall accuracy of solving arithmetic problems to 90.38%.

5.3 Error Analysis

An equation that can be generated from a *change* or *comparision* formula can also be generated by a *part whole* formula. Four such errors happened for the *change* problems and out of the 33 *compare* problems, 18 were solved by *part whole*. Also, there are 3 problems that require two applications. One example of such problem is, "*There are 48 erasers in the drawer and 30 erasers on the desk. Alyssa placed 39 erasers and 45 rulers on the desk. How many erasers are now there in total ?*". To solve this we need to first combine the two numbers 48 and 30 to find the total number of erasers she initially had. This requires the knowledge of 'part-whole'. Now, that sum of 48 and 30, 39 and x can be connected together using the 'change' formula. With respect to 'solving' arithmetic problems, we find the following categories as the major source of errors:

Problem Representation: Solving problems in this category requires involved representation. Consider the problem, '*Sally paid $* 12.32 *total for peaches , after a '*3 *dollar' coupon , and $* 11.54 *for cherries . In total , how much money did Sally spend?*'. Since the associated verb for the variable 3 dollar is 'pay', our system incorrectly thinks that *Sally* did spend it.

Information Gap: Often, information that is critical to solve a problem is not present in the text. E.g. *Last year , 90171 people were born in a country , and 16320 people immigrated to it . How many new people began living in the country last year ?*. To correctly solve this problem, it is important to know that both the event 'born' and 'immigration' imply the 'began living' event, however that information is missing in the text. Another example is the problem, "*Keith spent $6.51 on a rabbit toy , $5.79 on pet food , and a cage cost him $12.51 . He found a dollar bill on the ground. What was the total cost of Keith 's purchases?* ". It is important to know here that if a cage cost Keith $12.51 then Keith has spent $12.51 for cage.

Modals: Consider the question '*Jason went to 11 football games this month . He went to 17 games last month , and plans to go to 16 games next month . How many games will he attend in all?*' To solve this question one needs to understand the meanings of the verb "plan" and "will". If we replace "will" in the question by "did" the answer will be different. Currently our algorithm cannot solve this problem and we need to either use a better representation or a more powerful learning algorithm to be able to answer correctly. Another interesting example of this kind is the following: "For his car , Mike spent $118.54 on speakers and $106.33 on new tires . Mike wanted 3 CD 's for $4.58 but decided not to . In total , how much did Mike spend on car parts?"

Type		MA1	IXL	MA2
part whole	Total	59	89	51
	correct	59	81	40
change	Total	74	18	68
	correct	70	15	56
compare	Total	0	33	0
	correct	0	0	0

Table 6: Accuracy on recognizing the correct application. None of the MA1 and MA2 dataset contains "compare" problems so the cross validation accuracy on "IXL" for "compare" problems is 0.

Incomplete IsA Knowledge: For the problem "Tom bought a skateboard for $ 9.46 , and spent $ 9.56 on marbles . Tom also spent $ 14.50 on shorts . In total , how much did Tom spend on toys ? ", it is important to know that 'skateboard' and 'marbles' are toys but 'shorts' are not. However, such knowledge is not always present in ConceptNet which results in error.

Parser Issue: Error in dependency parsing is another source of error. Since the attribute values are computed from the dependency parse tree, a wrong assignment (mostly for verbs) often makes the entity irrelevant to the computation.

6 Conclusion

Solving math word problems often requires explicit modeling of the word. In this research, we use well-known math formulas to model the word problem and develop an algorithm that learns to map the assertions in the story to the correct formula. Our future plan is to apply this model to general arithmetic problems which require multiple applications of formulas.

7 Acknowledgement

We thank NSF for the DataNet Federation Consortium grant OCI-0940841 and ONR for their grant N00014-13-1-0334 for partially supporting this research.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Daniel G Bobrow. 1964a. Natural language input for a computer problem solving system.

Daniel G. Bobrow. 1964b. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, AFIPS '64 (Fall, part I), pages 591–614, New York, NY, USA. ACM.

Samuel R Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326*.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of ai. *AI Magazine.(To appear)*.

Peter Clark. 2015. Elementary school science and math tests as a driver for ai: Take the aristo challenge! In *AAAI*, pages 4019–4021.

Ido Dagan, Bill Dolan, Bernardo Magnini, and Dan Roth. 2010. Recognizing textual entailment: Rational, evaluation and approaches–erratum. *Natural Language Engineering*, 16(01):105–105.

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

Edward A Feigenbaum and Julian Feldman. 1963. Computers and thought.

Charles R Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Dan A Hinsley, John R Hayes, and Herbert A Simon. 1977. From words to equations: Meaning and representation in algebra word problems. *Cognitive processes in comprehension*, 329.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Walter Kintsch and James G Greeno. 1985. Understanding and solving word arithmetic problems. *Psychological review*, 92(1):109.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. Association for Computational Linguistics.

Hector J Levesque. 2011. The winograd schema challenge.

Hugo Liu and Push Singh. 2004. Conceptneta practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. EMNLP.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Lisbon, Portugal*.

Robert Speer and Catherine Havasi. 2013. Conceptnet 5: A large semantic network for relational knowledge. In *The Peoples Web Meets NLP*, pages 161–176. Springer.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822.

Unravelling Names of Fictional Characters

Katerina Papantoniou
Institute of Computer
Science, FORTH
Heraklion, Greece
papanton@ics.forth.gr

Stasinos Konstantopoulos
Institute of Informatics &
Telecommunications, NCSR 'Demokritos'
Ag. Paraskevi 153 10, Athens, Greece
konstant@iit.demokritos.gr

Abstract

In this paper we explore the correlation between the sound of words and their meaning, by testing if the polarity ('good guy' or 'bad guy') of a character's role in a work of fiction can be predicted by the name of the character in the absence of any other context. Our approach is based on phonological and other features proposed in prior theoretical studies of fictional names. These features are used to construct a predictive model over a manually annotated corpus of characters from motion pictures. By experimenting with different mixtures of features, we identify phonological features as being the most discriminative by comparison to social and other types of features, and we delve into a discussion of specific phonological and phonotactic indicators of a character's role's polarity.

1 Introduction

Could it be possible for fictional characters' names such as 'Dr. No' and 'Hannibal Lecter' to be attributed to positive characters whereas names such as 'Jane Eyre' and 'Mary Poppins' to negative ones? Could someone guess who is the hero and who is the competitor based only on the name of the character and what would be the factors that contribute to such intuition? Literary theory suggests that it should be possible, because fictional character names function as expressions of experience, ethos, teleology, values, culture, ideology, and attitudes of the character.

However, work in literary theory, psychology, linguistics and philosophy has studied fictional names by analysing individual works or small clusters of closely related works, such as those of a particular author. By contrast, we apply tools from computational linguistics at a larger scale aiming to identify more general patterns that are not tied to any specific creator's idiosyncrasies and preferences; in the hope that extracting such patterns can provide valuable insights about how the sound of names and, more generally, words correlates with their meaning.

At the core of our approach is the idea that the names of fictional characters follow (possibly subconsciously) a perception of what a positive or a negative name ought to sound like that is shared between the creator and the audience. Naturally the personal preferences or experiences of the creator might add noise, but fictional characters' names will at least not suffer (or suffer less) from the systematic cultural bias bound to exist in real persons' names.

In the remainder of this paper, we first present the relevant background, including both theoretical work and computational work relevant to peoples' names (Section 2). Based on this theoretical work, we then proceed to formulate a set of features that can be computationally extracted from names, and which we hypothesise to be discriminative enough to allow for the construction of a model that accurately predicts whether a character plays a positive or negative role in a work of fiction (Section 3). In order to test this hypothesis, we constructed a corpus of characters from popular English-language motion pictures. After describing corpus construction and presenting results (Section 4), we proceed to discuss these results (Section 5) and conclude (Section 6).

2 Background

2.1 Onomastics

The procedure of naming an individual, a location or an object is of particular importance and serves

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2154–2163,
Berlin, Germany, August 7-12, 2016.

purposes beyond the obvious purpose of referring to distinct entities. Characteristics such as place of origin, gender, and socioeconomic status can often be guessed from the name or nickname that has been attributed to an individual. *Onomastics*, the study of the origin, history, and use of proper names has attracted scholarly attention as early as antiquity and Plato's 'Cratylos' (Hajdú, 1980).

In fiction and art, in particular, names are chosen or invented without having to follow the naming conventions that are common in many cultures. This allows creators to apply other criteria in selecting a name for their characters, one of which being the intuitions and preconceptions about the character that the name alone implies to the audience. Black and Wilcox (2011) note that writers take informed and careful decisions when attributing names to their characters. Specifically, while care is taken to have names that are easily identifiable and phonologically attractive, or that are important for personal reasons, these are not the only considerations: names are chosen so that they match the personality, the past, and the cultural background of a character.

According to Algeo (2010) behind each name lies a story while Ashley (2003) suggests that a literary name must be treated as a small poem with all the wealth of information that implies. Markey (1982) and Nicolaisen (2008) raised concerns on whether onomastics can be applied to names in art given the different functional roles of names as well as their intrinsic characteristics, namely sensitivity and creativity. 'Redende namen' (significant names) is a widespread theory that seeks the relationship between name and form (Rudnyckyj, 1959). According to this theory, there is a close relationship between the form of a name and its role. This consideration is still prevalent to date as shown by Chen (2008) in her analysis of names in comic books, where names transparently convey the intentions of the creator for the role of each character. Another concern is whether the study of literary names should be examined individually for each creative work or if generalizations can be made (Butler, 2013). However, the scope of most studies is limited to individual projects or creators, creating an opportunity for computational methods that can identify generalizations and patterns across larger bodies of literary work than what is manually feasible.

2.2 Related Work

Although serving radically different purposes and applications than our investigation, various methods for the computational analysis of proper nouns have been developed in natural language processing. Without a doubt, some of the oldest and most mature technologies that exploit the properties of proper nouns are those addressing *named entity recognition and categorization (NERC)*. In this direction, there is a recently ongoing effort for the extension of NERC tools so that they cover the needs of literary texts (Borin et al., 2007; Volk et al., 2009; Kokkinakis and Malm, 2011).

Moving beyond recognition, effort has been made to explore characteristics and relationships of literary characters (Nastase et al., 2007). Typically, however, these efforts take advantage of the context, and very little work tries to extract characteristics of literary characters from their names alone. One example is the application of language identification methods in order to extract the cultural background of proper names (Konstantopoulos, 2007; Bhargava and Kondrak, 2010; Florou and Konstantopoulos, 2011). This work showed that people's names in isolation are more amenable to language identification than common nouns. Konstantopoulos (2007), in particular, reports inconclusive results at pinpointing the discriminative features that are present in people's names but not in other words.

Another relatively recent and related research direction that does not focus on proper nouns investigates elements of euphony mostly by examining phonetic devices. The focus is to identify how the sound of words can foster its effectiveness in terms of persuasion (Guerini et al., 2015) or memorability (Danescu-Niculescu-Mizil et al., 2012).

3 Approach

These earlier attempts relied on the examination of predictive models of n-grams in order to identify the n-grams that are the best discriminants. The aim was that by inspecting these most discriminative n-grams, meaningful patterns would emerge and serve as the vehicle for formulating hypotheses about the correlation between what names sound like and the cultural background of the persons bearing them.

This approaches largely ignored the background in onomastics and literary research. By contrast, we exploit this prior body of theoretical work

ID	Feature	Category	Type
1	words count	phonological	numeric
2	vowels count	phonological	numeric
3	consonants count	phonological	numeric
4	plosives count	phonological	numeric
5	fricatives count	phonological	numeric
6	affricates count	phonological	numeric
7	nasals count	phonological	numeric
8	vowel start	phonological	categorical
9	vowel end	phonological	categorical
10	voice start	phonological	categorical
11	subsequent letters count	phonological	categorical
12	low vowel	phonological	categorical
13	high vowel	phonological	categorical
14	definite article	lexical form	categorical
15	consonance	poetic	numeric
16	assonance	poetic	numeric
17	alliteration	poetic	numeric
18	name and title resemblance	domain	numeric
19	credit index	domain	numeric
20	genre	domain	categorical
21	sentiment soundex wordnet	emotions	numeric
22	sentiment levenshtein wordnet	emotions	numeric
23	gender	social	categorical
24	foreign suffix	social	categorical
25	first name frequency	social	numeric
26	last name frequency	social	numeric
27	full name frequency	social	numeric
28	honor	social	categorical

Table 1: List of features

to define more sophisticated features that directly correspond to theoretical hypotheses. Our empirical experiments are now aimed at identifying the features (and thus hypotheses) that are the most discriminative, rather than at hoping that a coherent hypothesis can be formulated by observing patterns in n-gram features.

In the remainder of this section, we will present these hypotheses and the machine-extracted features that reflect them. The features are also collected in Table 1.

3.1 Emotions

Hypothesis 1 *The (positive or negative) polarity of the sentiment that a character's name evokes is associated with the polarity of the character's role.*

The understanding of how the language transmits emotions has attracted significant research attention in the field of Computational Linguistics. Most of the relevant literature is directed towards calculating sentiment for units at the document or sentence level. These works are usually boosted by semantic dictionaries that provide information about the emotional hue of concepts such as the Linguistic Inquiry and Word Count (LIWC) (Pennebaker et al., 2001), the Harvard General Inquirer (Stone et al., 1966), the WordNet Affect (Strapparava and Valitutti, 2004) and SentiWordNet (Esuli and Sebastiani, 2006). In our task, the absence of context and the inherent arbitrariness in naming (even in fictional names) increases the difficulty in conveying emotional quality to names. More specifically, the intriguing part was to associate fictional names with concepts from a semantic sentiment resource in order to approximate a sentiment value. To achieve this we used SentiWordNet: a linguistic resource that has derived from the annotation of WordNet synsets according to the estimated degree of positive, negative or neutral hue. The overall valence for a given name is calculated as the sum of the valence of its elements (first name, surname). The valence of each name element is the average valence of all SentiWordNet concepts that are associated with it. To associate a name element and a SentiWordNet concept we used the Soundex phonetic distance and the Levenshtein lexicographic distance (Levenshtein, 1966). A heuristic threshold is used to decide whether a name and a SentiWordNet concept are associated.

More formally, the valence $\mathrm{val}(n)$ of a name n comprising name elements e_i is calculated as follows:

$$\mathrm{val}(e_i) = \frac{\sum_{u \in \mathrm{ass}_S(e_i)} \mathrm{swn}(u) + \sum_{v \in \mathrm{ass}_L(e_i)} \mathrm{swn}(v)}{|\mathrm{ass}_S(e_i)| + |\mathrm{ass}_L(e_i)|}$$

$$\mathrm{val}(n) = \sum_i \mathrm{val}(e_i)$$

where $\mathrm{ass}_S(\cdot)$ is the set of SentiWordNet concepts that are Soundex-associated with the given name element, $\mathrm{ass}_L(\cdot)$ the set of SentiWordNet concepts that are Levenshtein-associated with the given name element, and $\mathrm{swn}(\cdot)$ the valence assigned to the given concept by SentiWordNet.

3.2 Stylistic and poetic features

Hypothesis 2 *Assuming Ashley's (2003) and Butler's (2013) position that 'a name can be a whole "poem" in as little as a single word' we assume that stylistic features usually found in poems can be extracted from the names of fictional charac-*

ters, and that such features correlate with the polarity of their roles.

The first quantitative analysis efforts of the poetic style can be found in the 1940s and in the study of the poet and literary critic Josephine Miles (1946; 1967) where she studied the features of poems over time. Despite the great contribution of this work and others that followed, the creation of a framework for quantitative poetic style analysis remained limited to a small number of poems and much of the work was done manually. The work of Kaplan and Blei (2007) is an attempt to automate and analyze large volumes of poems exploring phonological, spelling and syntax features. For our work, we identified the following poetic devices that can be applied to isolated names:

- **Alliteration:** a stylistic literary device identified by the repeated sound of the first consonant in a series of multiple words, or the repetition of the same sounds of the same kinds of sounds at the beginning of words or in stressed syllables of a phrase.
  Examples: Peter **P**arker, **P**eter **P**an
- **Consonance:** a poetic device characterized by the repetition of the same consonant two or more times in short succession.
  Examples: **L**ill**l**ian He**ll**man, F**r**eddy K**r**uege**r**, Hannibal **L**ecter, **K**risten Parke**r**
- **Assonance:** a poetic device characterized by the repetition of the same vowel two or more times in short succession.
  Examples: Bobb**ie** R**i**tch**ie**

3.3 Phonological features

Hypothesis 3 *The presence of specific phonological features can reveal evidence of the role of a character in an artistic artifact.*

Linguistic theory widely adopts the concept of arbitrary relationship between the *signifier* and the *signified* (de Saussure, 1916 1983; Jakobson, 1965). However, an increasing volume of works in various fields investigates the existence of non-arbitrary relations between phonological representation and semantics, a phenomenon known as *phonological iconicity*. Standing from the side of Computational Linguistics and with the intuition that in fictional names the correlation between a word's form and the emotion it expresses will be stronger, we examined a wide range of phonology-related features, shown in Table 1. It should be noted that these features are extracted from the phonetic representation of names derived by applying the spelling-to-phoneme module of the *espeak* speech synthesizer.[1]

3.4 Sociolinguistic features

Hypothesis 4 *We hypothesize that social aspects of names — such as frequency of use or use of foreign names in a given environment — can relate to role of a fictional character. For instance, a 'girl next door' role is more likely to be assigned a very popular female name than a name that sounds hostile or foreign.*

The frequency of names in U.S.A was calculated based on the Social Security Death Index (SSDI), a publicly available database that records deaths of U.S.A citizens since 1936.[2] The same dataset was also used to build a model for recognizing foreign-looking names. More specifically, we trained n-gram language models of order 2–5 against the dataset for both orthographic and phonetic representation using the berkeleylm library (Pauls and Klein, 2011). We then heuristically defined a threshold that correlates well with foreign-looking suffixes. Analogously with the name frequency we extract the gender of each name using a baby names dataset that includes gender information.[3] For unisex names the prevalent gender was picked. Finally, honorific titles (e.g. Professor, Phd, Mr, Mrs etc.) were also extracted from names. Honorific titles are intriguing due to their ambiguous meaning since they can express respect and irony in different contexts.

3.5 Domain features

Hypothesis 5 *We pursued indications to check if domain-related features such as the appearance time of a character in a movie, the movie title or the movie genre is associated (correlates) with the problem under study.*

In this category lies the $feature_{sameastitle}$ since anyone with a quick glance in a list of films would notice that a fictional name often consists of, or is the part of, the movie title, as in, There's Something about Mary, Hannibal, Thelma & Louise, Rocky, etc. On IMDB character names are presented in the form of a list in

[1] Please cf. `http://espeak.sourceforge.net`

[2] Please cf. `https://archive.org/details/DeathMasterFile`

[3] Specifically, we used `https://www.ssa.gov/oact/babynames/state/index.html`

descending order based on screen credits. In the $feature_{creditindex}$ we want to check if the naming process is more assiduous for the roles of protagonists based on this list. In the same direction, we examine the $feature_{genre}$ for a possible correlation between the role of a character and the genre of a film.

4 Experiments and Results

4.1 Data Collection and Annotation

In order to validate our approach, we first need a corpus of names of fictional characters, annotated with the polarity of their role. As such a resource does not exist to the best of our knowledge, we have created it for the purposes of the work described here.

Our decision to use motion pictures rather than other fictional work is motivated by the relative ease of finding annotators familiar with the plot of these works, so that we could get reliable annotations of the polarity of the leading roles. We compiled a list of 409 movies based on the following criteria:

- That they are widely known films, covering all genres of film production. We automatically crosschecked if the candidate movies are included in DBPedia[4] and YAGO[5], as these are indicators that the films are known to the general public.
- That they have received some award or are positively evaluated by users (i.e., have an IMDB rating of 5.0 or higher). The underlying assumption is that this criterion selects major productions where care has been given to even the most minute detail, including the names of the major characters and what these names connote to the audience.
- That they are recent productions, so that annotators can easily recall the plot and the characters.

We then asked volunteers to select any movie from the list that they where very familiar with, and to assign one of *positive*, *negative* or *neutral* to the top-most characters in the credits list, working only as far down the credits list as they felt confident to. The three categories were defined as follows in the annotation guidelines:

Figure 1: Character annotation tool

- Positive: when the role of the character in the plot left a positive impression on you when you saw the movie.
- Negative: when the role of a character left a negative impression on you when you saw the movie.
- Neutral: when the role of the character is important for the plot, but you are in doubt or cannot recall whether it was a positive or a negative role.

Neutral tags are ignored in our experiments. They were foreseen only to allow annotators to skip characters and still have a sense of accomplishment, so that they only make choices that they are confident with.

We used the Hypothes.is[6] open source annotation application. The annotation was carried out by having volunteers install the Hypothes.is Web browser extension and then visit the IMDB[7] page of any of the movies on our list (direct links were provided to them in the guidelines). IMDB was chosen due to its popularity, so that annotators would already be familiar with the online environment. The annotators tagged the character names directly on the IMDB page and the annotations where collected for us by Hypothes.is (Figure 1).

Eight annotators participated in the procedure and provided 1102 positive and 434 negative tags for characters of 202 movies, out of the 409 movies in the original list. Table 2 gives the annotation distribution per movie genre.

The reliability of the annotated collection by means of inter-rater agreement was also measured. For this purpose, various standard agreement measures (Meyer et al., 2014) were calculated, all showing very high agreement among the annotators (Table 3). This demonstrates that the annota-

[4] http://wiki.dbpedia.org
[5] http://www.mpi-inf.mpg.de/yago
[6] https://hypothes.is
[7] http://www.imdb.com

	Original		Resampled	
	Pos	Neg	Pos	neg
Action	262	107	244	102
Adventure	126	63	133	62
Animation	73	22	63	27
Biography	28	6	39	8
Comedy	78	25	23	21
Crime	68	25	81	18
Drama	81	40	76	32
Horror	16	12	28	13
Musical	0	0	0	0
Mystery	20	13	26	17
Sci-Fi	2	0	2	0
Thriller	0	0	0	0
Western	1	2	3	2
Sum	**755/315**	**768/302**		

Table 2: Number of annotations per genre before and after resampling

Measure	Value
Percentage Agreement	0.963
Hubert Kappa Agreement	0.980
Fleiss Kappa Agreement	0.973
Krippendorff Alpha Agreement	0.979

Table 3: Inter-annotator agreement

tion task is well-formulated, but does not guarantee that our classification task is consistent, since the latter will use different information than that used by the annotators. That is to say, the annotators had access to their understanding of the movies' plot to carry out the task, whereas our classification task will be performed over the characters' names alone.

The collection is publicly available, including the guidelines and instructions to the annotators, the source code for the annotation tool, and the source code for the tool that compiles Weka ARFF files from the JSON output of the annotation tool.[8]

4.2 Experimental Design

The experimental design consisted of an iterated approach performing experiments with different sets of features. This process was driven by a preliminary chi-squared analysis in order to exploit feature significance. The algorithms that are used for the experiments are Naive Bayes and J48

[8] https://bitbucket.org/dataengineering/fictionalnames

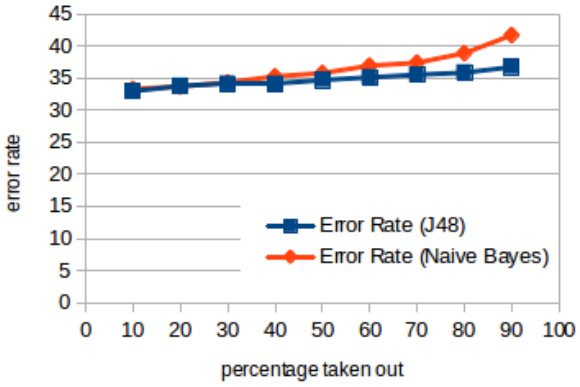


Figure 2: Learning curve for the number of instances

	All		Without domain	
	NB	J48	NB	J48
Recall	0.723	0.824	0.718	0.803
Prec.	0.731	0.822	0.515	0.801
F-score	0.618	0.823	0.6	0.802

Table 4: Comparison of Naive Bayes and J48

(Salzberg, 1994) decision trees. Each experiment is done using a 10-fold cross validation on the available data, using a confidence factor of 0.25 for post-pruning. For all the experiments we used the Weka toolkit (Hall et al., 2009). Due to the imbalance of our dataset in favor of positive classes (see Table 2), we sub-sampled the dataset maintaining the initial genre distribution. We also applied *principal component analysis (PCA)* in order to guarantee the independence of the classification features, as required by the Naive Bayes algorithm. To explore the behavior of the algorithms to the change of trained data we generated the learning curves shown in Figure 2. In both cases the learning curves are well-behaved since the error rate grows monotonically as the training set shrinks. However, the precision, recall, and F-scores achieved by J48 are significantly better that those of Naive Bayes (Table 4).

This preliminary experiment led us to use J48 for the main experiment, where we try different features in order to understand which are the most discriminative ones. These results are collected in Table 5 and discussed immediately below.

5 Discussion of Results

A first observation that can be easily made is that the domain features are good discriminants. As these features exploit information such as credit-

	Rec.	Prec.	F-score
Without domain features	0.803	0.801	0.802
Only domain features	0.725	0.699	0.667
Only phonological features	0.790	0.786	0.787
Without poetic features	0.836	0.832	0.833
Without consonance feature	0.823	0.820	0.821
Without emotions features	0.814	0.810	0.811
Without phonological features	0.798	0.792	0.793
Without social features	0.807	0.803	0.804
All features	0.824	0.822	0.823

Table 5: Performance of J48 for different feature settings

Most frequent in positive characters	
Phoneme n-gram	Examples
/lɪ/	Ned Alleyn (Shakespeare in Love)
/an/	Anouk Rocher (Chocolat)
/aɪ/	Eliza Doolittle (My Fair Lady)
/nɪ/	Linguini (Ratatouille)
/ɪst/	Kevin McCallister (Home Alone)
/ɹəʊ/	Frodo (The Lord of the Rings)
/and/	Dylan Sanders (Charlie's Angels)
/stə/	C.C. Baxter (The Apartment)
Most frequent in negative characters	
Phoneme n-gram	Examples
/ən/	Tom Buchanan (The Great Gatsby)
/əʊ/	Iago (Aladdin)
/tə/	Norrington (Pirates of the Caribbean)
/ɪɪ/	Tom Ripley (The Talented Mr. Ripley)
/mən/	Norman Bates (Psycho)
/mɪs/	Mystique (X-Men)
/ktə/	Hannibal Lecter (Hannibal)

Table 6: Frequent phoneme {2,3}-grams

ing order that is outside the scope of our hypotheses, there were expected to be good discriminants and are included for comparison only.

By comparing the performance of all features (F = 82%), domain-only features (F = 68%), and all-except-domain features (F = 80%), we can immediately understand that our name-intrinsic features are better discriminants than domain features; in fact, name-intrinsic features not just better than domain features, they are by themselves almost as good as domain and name-intrinsic features combined. This is a significant finding, as it validates our core hypothesis that there is a correlation between what fictional character names look and sound like and the role they play in the plot of the fictional work they appear in.

We will now proceed to look in more detail into the different categories of features used, in order to gain further insights about specific discriminants.

5.1 Phonological Features

The phonological features are important separation criteria as evidenced by the drop in performance when they are excluded from the experimental setup (Table 5). Specifically, using all features except phonological features is equivalent to using phonological features alone (about F = 79% in both cases) and slightly worse that using all name-intrinsic features (about F = 80%). By comparison, removing any other category *increases* performance, leading us to believe that all other features are actually adding noise (rather than discriminatory power) to the feature space.

In order to delve more into this category of features, we proceeded with an n-gram analysis (of order 1 through 4) to look for correlations between phonemes. The results clearly demonstrated the positive effect of the number of vowels (normalized by the length of the utterance) to the positive category. As far as the consonants are concerned, voiced (e.g. /ʌ/, /g/, /d/, /w/) seem to relate more to the negative class. Table 7 summarizes a more fine-grained analysis for the consonants based on their categorization.

The environment plays an important role, with specific combinations showing tendencies that are not observed with isolated phonemes. For example, diphoneme /an/ relates to positive class while /ən/ to negative. Table 6 lists some frequent phoneme 2- and 3-gram examples. The position of each phoneme also seems to play an crucial role

Phonemes	Class
/p/, /b/ (bilabial plosive)	P
/l/ (alveolar lateral)	P
/f/, /v/ (labiodental africative)	N
/k/, /g/ (velar plosive)	N
/t/, /d/ (alveolar plosive)	N
/dʒ/, /tʃ/ (affricate)	N
/m/, /n/ (nasal)	N
/ɹ/ (alveolar retroflex)	N

Table 7: Consonants behavior

in the classification task. Specifically, we note that starting with a vowel or a consonant are among the most discriminating features. These observations are consistent to a great extent with work in psychology and literary theory that studied phonological iconicity for common words (Nastase et al., 2007; Auracher et al., 2011; Schmidtke et al., 2014).

Some contradictory conclusions in these works are attributed by researchers to the methodologies applied, while at the same time concerns are raised whether such methodologies can inductively lead to cross-language and general conclusions (Auracher et al., 2011). Table 8 summarizes some of the outcomes of these works.

5.2 Emotion and Affect

The analysis showed that the features that calculate the emotional load of fictional names based on SentiWordNet contribute to the classification task. However, we believe that there is still room for improvement for the performance of this feature mainly towards the optimization of the selection threshold in order to reduce the degree of false positive matches as well as the addition of more lexical resources for example WordNet Affect or LIWC.

5.3 Social Features

The annual publication *It's a Man's (Celluloid) World* examines the representation of female characters every year. According to its 2015 results (Lauzen, 2015), gender stereotypes were abundant with female characters being younger than their male counterparts and more likely to have prosocial goals including supporting and helping others. This bias makes the gender feature discriminative, but in a way that is not linguistically interesting: female characters are simply related to the positive class.

Reference	Description
Taylor and Taylor (1965)	evidence that pleasantness relations are language specific
Fonagy (1961)	sonorants (e.g., /l/,/m/) more common in tender poems, plosives (e.g., /k/,/t/) in aggressive ones
Miall (2001)	Passages about Hell from Miltons "Paradise Lost" were found to contain significantly more front vowels and hard consonants than passages about Eden while the latter contained more medium back vowels
Whissell (1999)	plosives correlate with unpleasant words
Auracher et al. (2011)	nasals (e.g., /m/) relate to sadness, plosives (e.g., /p/) to happiness, parallels across remote languages
Zajonc et al. (1989)	umlaut /y/ causes negative affective states

Table 8: Phonological iconicity studies

A somewhat surprising result was that the foreign suffix feature is not discriminative. The hypothesis that the concept of the 'other' is stereotyped negatively does not seem to be true in our dataset. A closer investigation might identify genres where this hypothesis holds (e.g., war movies), but this would be implicit pragmatic information about the context of the film rather than a linguistically interesting finding.

5.4 Poetic and Stylistic Features

The experimental findings show that literary devices can actually be identified in fictional characters names, but the same findings also indicate that they do not contribute significantly to the classification task. More specifically, consonance is the only stylistic/poetic feature that affects classification.

6 Conclusions and Future Work

In this paper we test the hypothesis that the sound and the form of fictional characters' names correlates with meaning, in our particular case with

the respective characters' role in the work of fiction. We restricted our study to fictional characters since they are not tied to cultural conventions of naming, such as names that run in a family, so that we are able to look for patterns that are perceived as positive or negative by the audience and used as such (consciously or not) by the creator.

Our experiments have verified that features intrinsic to the names and without any reference to the plot or, in general, any other context are discriminative. Furthermore, we have discovered that the most discriminative features are of phonological nature, rather than features that hint at pragmatic information such as the gender or origin of the character. A further contribution of our work is that we ran an annotation campaign and created an annotated corpus of fictional movie characters and their corresponding polarity. This corpus is offered publicly, and can serve experimentation in the digital humanities beyond the scope of the experiments presented here.

Our future research will test the correlation between the polarity and the name of a fictional character beyond the movie domain. It would, for example, be interesting to seek differences between spoken names (as in films) and names that are only meant to be read (as in literature). In addition, using written literature will allow us to compare texts from different periods, pushing earlier than the relatively young age of motion pictures. Character polarity annotations in written literature could be created by, for example, applying sentiment analysis to the full text of the work.

References

[Algeo2010] John Algeo. 2010. Is a theory of names possible? *Names*, 58(2):90–96.

[Ashley2003] Leonard R. N. Ashley. 2003. *Names in Literature*. Bloomington, IN: Authorhouse (formerly 1st Books).

[Auracher et al.2011] Jan Auracher, Sabine Albers, Yuhui Zhai, Gulnara Gareeva, and Tetyana Stavniychuk. 2011. P is for happiness, N is for sadness: Universals in sound iconicity to detect emotions in poetry. *Discourse Processes*, 48(1):1–25.

[Bhargava and Kondrak2010] Aditya Bhargava and Grzegorz Kondrak. 2010. Language identification of names with SVMs. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 693–696, Los Angeles, California, June. Association for Computational Linguistics.

[Black and Wilcox2011] Sharon Black and Brad Wilcox. 2011. 188 unexplainable names: Book of Mormon names no fiction writer would choose. *Religious Educator*, 12(2).

[Borin et al.2007] Lars Borin, Dimitrios Kokkinakis, and Leif-Jöran Olsson. 2007. Naming the past: Named entity and animacy recognition in 19th century Swedish literature. In *ACL 2007 Workshop on Language Technology for Cultural Heritage Data (LaTeCH 2007)*, pages 1–8.

[Butler2013] James Odelle Butler. 2013. *Name, Place, and Emotional Space: Themed Semantics in Literary Onomastic Research*. Ph.D. thesis, University of Glasgow.

[Chen2008] Lindsey N. Chen. 2008. Ethnic marked names as a reflection of United States isolationist attitudes in Uncle $crooge comic books. *Names*, 56(1):19–22.

[Danescu-Niculescu-Mizil et al.2012] Cristian Danescu-Niculescu-Mizil, Justin Cheng, Jon M. Kleinberg, and Lillian Lee. 2012. You had me at hello: How phrasing affects memorability. *CoRR*, abs/1203.6360.

[de Saussure1916 1983] Ferdinand de Saussure. [1916] 1983. *Course in General Linguistics*. Duckworth, London. (translation Roy Harris).

[Esuli and Sebastiani2006] Andrea Esuli and Fabrizio Sebastiani. 2006. SENTIWORDNET: A publicly available lexical resource for opinion mining. In *Proceedings of the 5th Conference on Language Resources and Evaluation (LREC'06)*, pages 417–422.

[Florou and Konstantopoulos2011] Eirini Florou and Stasinos Konstantopoulos. 2011. A quantitative and qualitative analysis of Nordic surnames. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011), May 11-13, 2011, Riga, Latvia*, volume 11 of *NEALT Proceedings Series*.

[Fonagy1961] Ivan Fonagy. 1961. *Communication in Poetry*. William Clowes.

[Guerini et al.2015] Marco Guerini, Gözde Özbal, and Carlo Strapparava. 2015. Echoes of Persuasion: The Effect of Euphony in Persuasive Communication. *CoRR*, abs/1508.05817.

[Hajdú1980] Mihály Hajdú. 1980. The history of Onomastics. *Onomastica Uralica*, 2:7–45.

[Hall et al.2009] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November.

[Jakobson1965] Roman Jakobson. 1965. Quest for the Essence of Language. *Diogenes*, 13(51):21–37.

[Kaplan and Blei2007] David M. Kaplan and David M. Blei. 2007. A computational approach to style in American poetry. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, pages 553–558, October.

[Kokkinakis and Malm2011] Dimitrios Kokkinakis and Mats Malm. 2011. Character profiling in 19th century fiction. In *Workshop: Language Technologies for Digital Humanities and Cultural Heritage in conjunction with the Recent Advances in Natural Language Processing (RANLP). Hissar, Bulgaria.*

[Konstantopoulos2007] Stasinos Konstantopoulos. 2007. What's in a name? In Petya Osenova, Erhard Hinrichs, and John Nerbonne, editors, *Proceedings of Computational Phonology Workshop, International Conf. on Recent Advances in NLP, (RANLP), Borovets, Bulgaria, September 2007*.

[Lauzen2015] Martha Lauzen. 2015. It's a man's (celluloid) world: On-screen representations of female characters in the top 100 films of 2011. Technical report, San Diego State University Center for the Study of Women in Television and Film, School of Theatre, Television and Film, San Diego State University, San Diego, CA.

[Levenshtein1966] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, feb.

[Markey1982] T. L. Markey. 1982. Crisis and cognition in onomastics. *Names*, 30(3):129–142.

[Meyer et al.2014] Christian M. Meyer, Margot Mieskes, Christian Stab, and Iryna Gurevych. 2014. DKPro Agreement: An Open-Source Java Library for Measuring Inter-Rater Agreement. In *Proceedings of the 25th International Conference on Computational Linguistics: System Demonstrations (COLING)*, pages 105–109, Dublin, Ireland, August.

[Miall2001] David Miall. 2001. Sounds of contrast: An empirical approach to phonemic iconicity. *Poetics*, 29(1):55–70.

[Miles1946] Josephine Miles. 1946. Major adjectives in English poetry: From Wyatt to Auden. *University of California Publications in English*, 12.

[Miles1967] Josephine Miles. 1967. *Style and Proportion: The Language of Prose and Poetry*. Little, Brown and Co., Boston.

[Nastase et al.2007] Vivi Nastase, Marina Sokolova, and Jelber Sayyad Shirabad. 2007. Do happy words sound happy? A study of the relation between form and meaning for English words expressing emotions. In *Recent Advances in Natural Language Processing (RANLP 2007)*.

[Nicolaisen2008] William F. H. Nicolaisen. 2008. On names in literature. *Nomina*, 31:89–98.

[Pauls and Klein2011] Adam Pauls and Dan Klein. 2011. Faster and smaller n-gram language models. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

[Pennebaker et al.2001] JW Pennebaker, ME Francis, and RJ Booth. 2001. *Linguistic inquiry and word count [computer software]*. Mahwah, NJ: Erlbaum Publishers.

[Rudnyckyj1959] Jaroslav B. Rudnyckyj. 1959. Function of proper names in literary works. *Internationalen Vereinigung für moderne Sprachen und Literaturen*, 61:378–383.

[Salzberg1994] Steven L. Salzberg. 1994. C4.5: Programs for machine learning. *Machine Learning*, 16(3):235–240.

[Schmidtke et al.2014] David S. Schmidtke, Markus Conrad, and Jacobs Arthur M. 2014. Phonological iconicity. *Frontiers in Psychology*, 12.

[Stone et al.1966] Philip J. Stone, Dexter C. Dunphy, Marshall S. Smith, and Daniel M. Ogilvie. 1966. *The General Inquirer: A Computer Approach to Content Analysis*. MIT Press, Cambridge, MA.

[Strapparava and Valitutti2004] Carlo Strapparava and Alessandro Valitutti. 2004. WordNet-Affect: An affective extension of WordNet. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1083–1086. ELRA.

[Taylor and Taylor1965] I. K. Taylor and M. M. Taylor. 1965. Another look at phonetic symbolism. *Psychological Bulletin*, 65.

[Volk et al.2009] Martin Volk, Noah Bubenhofer, Adrian Althaus, and Maya Bangerter. 2009. Classifying named entities in an Alpine heritage corpus. *Künstliche Intelligenz*, pages 40–43.

[Whissell1999] Cynthia Whissell. 1999. Phonosymbolism and the emotional nature of sounds: Evidence of the preferential use of particular phonemes in texts of differing emotional tone. *Perceptual and Motor Skills*, 89(1):19–48, August.

[Zajonc et al.1989] R. B. Zajonc, Sheila T. Murphy, and Marita Inglehart. 1989. Feeling and facial efference: Implications of the vascular theory of emotion. *Psychological Review*, 96(3):395–416, July.

Most *babies* are *little* and most *problems* are *huge*: Compositional Entailment in Adjective-Nouns

Ellie Pavlick
University of Pennsylvania
epavlick@seas.upenn.edu

Chris Callison-Burch
University of Pennsylvania
ccb@cis.upenn.edu

Abstract

We examine adjective-noun (AN) composition in the task of recognizing textual entailment (RTE). We analyze behavior of ANs in large corpora and show that, despite conventional wisdom, adjectives do not always restrict the denotation of the nouns they modify. We use natural logic to characterize the variety of entailment relations that can result from AN composition. Predicting these relations depends on context and on commonsense knowledge, making AN composition especially challenging for current RTE systems. We demonstrate the inability of current state-of-the-art systems to handle AN composition in a simplified RTE task which involves the insertion of only a single word.

1 Overview

The ability to perform inference over utterances is a necessary component of natural language understanding (NLU). Determining whether one sentence reasonably implies another is a complex task, often requiring a combination of logical deduction and simple common-sense. NLU tasks are made more complicated by the fact that language is compositional: understanding the meaning of a sentence requires understanding not only the meanings of the individual words, but also understanding how those meanings combine.

Adjectival modification is one of the most basic types of composition in natural language. Most existing work in NLU makes a simplifying assumption that adjectives tend to be *restrictive*– i.e. adding an adjective modifier limits the set of things to which the noun phrase can refer. For example, the set of *little dogs* is a subset of the set of *dogs*, and we cannot in general say that *dog* entails *little dog*. This assumption has been exploited by high-performing RTE systems (MacCartney and Manning, 2008; Stern and Dagan, 2012), as well as used as the basis for learning new entailment rules (Baroni et al., 2012; Young et al., 2014).

However, this simplified view of adjectival modification often breaks down in practice. Consider the question of whether *laugh* entails *bitter laugh* in the following sentences:

1. Again his *laugh* echoed in the gorge.
2. Her *laugh* was rather derisive.

In (1), we have no reason to believe the man's laugh is bitter. In (2), however, it seems clear from context that we are dealing with an unpleasant person for whom *laugh* entails *bitter laugh*. Automatic NLU should be capable of similar reasoning, taking both context and common sense into account when making inferences.

This work aims to deepen our understanding of AN composition in relation to automated NLU. The contributions of this paper are as follows:

- We conduct an empirical analysis of ANs and their entailment properties.
- We define a task for directly evaluating a system's ability to predict compositional entailment of ANs in context.
- We benchmark several state-of-the-art RTE systems on this task.

2 Recognizing Textual Entailment

The task of recognizing textual entailment (RTE) (Dagan et al., 2006) is commonly used to evaluate the state-of-the-art of automatic NLU. The RTE task is: given two utterances, a *premise* (p) and a *hypothesis* (h), would a human reading p typically infer that h is most likely true? Systems are expected to produce either a binary (YES/NO) or trinary (ENTAILMENT/CONTRADICTION/UNKNOWN) output.

The type of knowledge tested in the RTE task has shifted in recent years. While older datasets mostly captured logical reasoning (Cooper et al., 1996) and lexical knowledge (Giampiccolo et al., 2007) (see Examples (1) and (2) in Table 1), the recent datasets have become increasingly reliant on common-sense knowledge of scenes and events (Marelli et al., 2014). In Example (4) in Table 1, for which the gold label is ENTAILMENT, it is perfectly reasonable to assume the dogs are playing. However, this is not necessarily true that *running* entails *playing*– maybe the dogs are being

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2164–2173,
Berlin, Germany, August 7-12, 2016.

(1)	FraCas	p	No delegate finished the report on time.	Quantifiers
		h	Some Scandinavian delegate finished the report on time.	(*no* → ¬*some*)
(2)	RTE2	p	Trade between China and India is expected to touch $20 bn this year. . .	Definitions
		h	There is a profitable trade between China and India.	(*$20 bn* → *profitable*)
(3)	NA	p	Some delegates finished the report on time.	Implicature
		h	Not all of the delegates finished the report on time.	(*some* → ¬*all*)
(4)	SICK	p	A couple of white dogs are running along a beach.	Common Sense
		h	Two dogs are playing on the beach.	(*running* → *playing*)

Table 1: Examples of sentence pairs coming from various RTE datasets, and the types of inference highlighted by each. While linguistic phenomena like implicature (3) have yet to be explicitly included in RTE tasks, commonsense inferences like those in (4) (from the SICK dataset) have become a common part of NLU tasks like RTE, question answering, and image labeling.

chased by a bear and are running for their lives! Example (4) is just one of many RTE problems which rely on intuition rather than strict logical inference.

Transformation-based RTE. There have been an enormous range of approaches to automatic RTE– from those based on theorem proving (Bjerva et al., 2014) to those based on vector space models of semantics (Bowman et al., 2015a). Transformation-based RTE systems attempt to solve the RTE problem by identifying a sequence of *atomic edits* (MacCartney, 2009) which can be applied, one by one, in order to transform p into h. Each edit can be associated with some entailment relation. Then, the entailment relation that holds between p and h overall is a function of the entailment relations associated with each atomic edit. This approach is appealing in that it breaks potentially complex p/h pairs into a series of bite-sized pieces. Transformation-based RTE is widely used, not only in rule-based approaches (MacCartney and Manning, 2008; Young et al., 2014), but also in statistical RTE systems (Stern and Dagan, 2012; Padó et al., 2014).

MacCartney (2009) defines an *atomic edit* applied to a linguistic expression as the deletion DEL, insertion INS, or substitution SUB of a subexpression. If x is a linguistic expression and e is an atomic edit, than $e(x)$ is the result of applying the edit e to the expression x. For example:

$$\begin{array}{lcl} x & = & a_1\ \textit{girl}_2\ \textit{in}_3\ a_4\ \textit{red}_5\ \textit{dress}_6 \\ e & = & \text{DEL}(\textit{red}, 5) \\ e(x) & = & a_1\ \textit{girl}_2\ \textit{in}_3\ a_4\ \textit{dress}_5 \end{array}$$

We say that the entailment relation that holds between x and $e(x)$ is *generated* by the edit e. In the above example, we would say that e generates a *forward entailment* ($\sqsubset$) since *a girl in a red dress* entails *a girl in a dress*.

3 Natural Logic Entailment Relations

Natural logic (MacCartney, 2009) is a formalism that describes entailment relationships between natural language strings, rather than operating over mathematical formulae. Natural logic enables both light-weight representation and robust inference, and is an increasingly popular choice for NLU tasks (Angeli and Manning, 2014; Bowman et al., 2015b; Pavlick et al., 2015). There are seven "basic entailment relations" described by natural logic, five of which we explore here.[1]

These five relations, as they might hold between an AN and the head N, are summarized in Figure 1. The *forward entailment* relation is the restrictive case, in which the AN (*brown dog*) is a subset of (and thus entails) the N (*dog*) but the N does not entail the AN (*dog* does not entail *brown dog*). The symmetric *reverse entailment* can also occur, in which the N is a subset of the set denoted by the AN. An example of this is the AN *possible solution*: i.e. all actual *solutions* are *possible solutions*, but there are an abundance of *possible solutions* that are not and will never be actual *solutions*. In the *equivalence* relation, AN and N denote the same set (e.g. the *entire universe* is the same as the *universe*), whereas in the *alternation* relation, AN and N denote disjoint sets (e.g. a *former senator* is not a *senator*). In the *independence* relation, the AN has no determinable entailment relationship to the N (e.g. an *alleged criminal* may or may not be a *criminal*).

4 Simplified RTE Task

The focus of this work is to determine the entailment relation that exists between an AN and its head N in a given context. To do this, we define a simplified entailment task identical to the normal RTE task, with the constraint that p and h differ only by one atomic edit e as defined in Section 2. We look only at insertion INS(A) and deletion DEL(A), where A must be a single adjective.

We use a 3-way entailment classification where the possible labels are ENTAILMENT, CONTRADICTION, and UNKNOWN. This allows us to recover the basic entailment relation from Section 3: by determining the labels associated with the INS operation and the DEL

[1] We omit two relationships: *negation* and *cover*. These relations require that the sets denoted by the strings being compared are "exhaustive." In this work, this requirement would be met when everything in the universe is either an instance of the noun or it is an instance of the adjective-noun (or possibly both). This is a hard constraint to meet, and we believe that the interesting relations that result from AN composition are adequately captured by the remaining 5 relations.

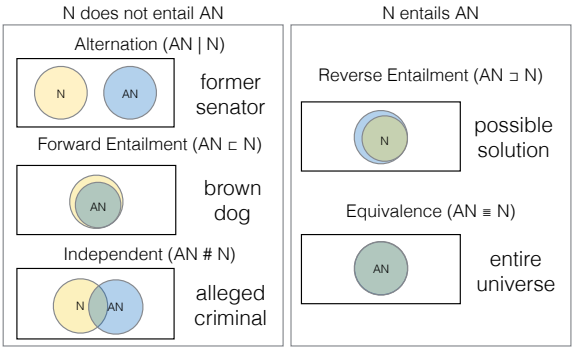


Figure 1: Different entailment relations that can exist between an adjective-noun and the head noun. The best-known case is that of *forward entailment*, in which the AN denotes a subset of the N (e.g. *brown dog*). However, many other relationships may exist, as modeled by natural logic.

operation, we can uniquely identify each of the five relations (Table 2).

	INS	DEL
Equivalence	ENTAILMENT	ENTAILMENT
Forward Entail.	ENTAILMENT	UNKNOWN
Reverse Entail.	UNKNOWN	ENTAILMENT
Independence	UNKNOWN	UNKNOWN
Alternation	CONTRADICTION	CONTRADICTION

Table 2: Entailment generated by INS(A) or DEL(A) for possible relations holding between AN and N. Both INS and DEL are required to distinguish all five entailment relations.

4.1 Limitations

Modeling denotations of ANs and N. We note that this task design does not directly ask about the relationship between the sets denoted by the AN and by the N (as shown in Figure 1). Rather than asking "Is this instance of AN an instance of N?" we ask "Is this statement that is true of AN also true of N?" While these are not the same question, they are often conflated in NLP, for example, in information extraction, when we use statements about ANs as justification for extracting facts about the head N (Angeli et al., 2015). We focus on the latter question and accept that this prevents us from drawing conclusions about the actual set theoretic relation between the denotation of AN and the denotation of N. However, we are able to draw conclusions about the practical entailment relation between statements about the AN and statements about the N.

Monotonicity. In this simplified RTE task, we assume that the entailment relation that holds overall between p and h is attributable wholly to the atomic edit (i.e. the inserted or deleted adjective). This is an over-simplification. In practice, several factors can cause the entailment relation that holds between the sentences overall to differ from the relation that holds between the AN and the N. For example, quantifiers and other downward-monotone operators can block or reverse entailments (*brown dog* $\rightarrow$ *dog*, but *no brown dog* $\not\rightarrow$ *no dog*). While we make some effort to avoid selecting such sentences for our analysis (Section 5.3), fully identifying and handling such cases is beyond the scope of this paper. We acknowledge that monotone operators and other complicating factors (e.g. multiword expressions) might be present in our data, but we believe, based on manual inspection, that they not frequent enough to substantially effect our analyses.

5 Experimental Design

To build an intuition about the behavior of ANs in practice, we collect human judgments of the entailments generated by inserting and deleting adjectives from sentences drawn from large corpora. In this section, we motivate our design decisions, before carrying out our full analysis in Section 6.

5.1 Human judgments of entailment

People often draw conclusions based on "assumptions that seem plausible, rather than assumptions that are known to be true" (Kadmon, 2001). We therefore collect annotations on a 5-point scale, ranging from 1 (definite contradiction) to 5 (definite entailment), with 2 and 4 capturing likely (but not certain) contradiction/entailment respectively. We recruit annotators on Amazon Mechanical Turk. We tell each annotator to assume that the premise "is true, or describes a real scenario" and then, using their best judgement, to indicate how likely it is, on a scale of 1 to 5, that the hypothesis "is also true, or describes the same scenario." They are given short descriptions and several examples of sentence pairs that constitute each score along the 1 to 5 scale. They are also given the option to say that "the sentence does not make sense," to account for poorly constructed p/h pairs, or errors in our parsing. We use the mean score of the three annotators as the true score for each sentence pair.

Inter-annotator agreement. To ensure that our judgements are reproducible, we re-annotate a random 10% of our pairs, using the same annotation setup but a different set of annotators. We compute the intra-class correlation (ICC) between the scores received on the first round of annotation, and those received in the second pass. ICC is related to Pearson correlation, and is used to measure consistency among annotations when the group of annotators measuring each observation is not fixed, as opposed to metrics like Fleiss's κ which assume a fixed set of annotators. On our data, the ICC is 0.77 (95% CI 0.73 - 0.81) indicating very high agreement. These twice-annotated pairs will become our test set in Section 7.

5.2 Data

Selecting contexts. We first investigate whether, in naturally occurring data, there is a difference between contexts in which the author uses the AN and contexts in which the author uses only the (unmodified) N. In other words, in order to study the effect of an A (e.g. *financial*) on the denotation of an N (e.g. *system*), is it better to look at contexts like (a) below, in which the author originally used the AN *financial system*, or to use contexts like (b), in which the author used only the N *system*?

(a) The TED spread is an indication of investor confidence in the U.S. *financial system*.

(b) Wellers hopes the *system* will be fully operational by 2015.

We will refer to contexts like (a) as *natural* contexts, and those like (b) as *artificial*. We take sample of 500 ANs from the Annotated Gigaword corpus (Napoles et al., 2012), and choose three natural and three artificial contexts for each. We generate p/h pairs by deleting/inserting the A for the natural/artificial contexts, respectively, and collect human judgements on the effect of the INS(A) operation for both cases.

Figure 2 displays the results of this pilot study. In sentences which contain the AN naturally, there is a clear bias toward judgements of "entailment." That is, in contexts when an AN appears, it is often the case that this A is superfluous: the information carried by the A is sufficiently entailed by the context that removing it does not remove information. Sentences (a) and (b) above provide intuition: in the case of sentence (a), trigger phrases like *investor confidence* make it clear that the *system* we are discussing is the *financial system*, whether or not the adjective *financial* actually appears. No such triggers exist in sentence (b).

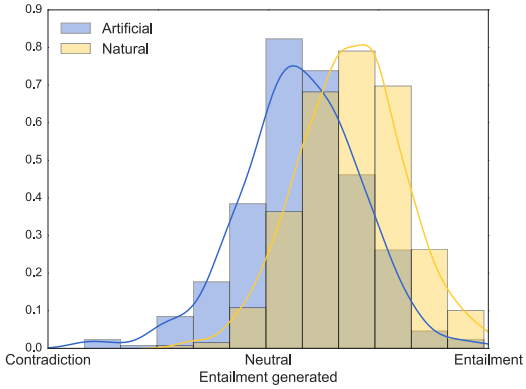


Figure 2: p/h pairs derived from natural contexts result in a notable bias toward judgements of "entailment" for the INS(A) operation, compared to p/h pairs derived from artificial contexts.

Selecting ANs. We next investigate whether the frequency with which an AN is used effects its tendency to entail/be entailed by the head N. Again, we run a small pilot study. We choose 500 ANs stratified across different levels of frequency of occurrence in order to determine if sampling the most frequent ANs introduces bias into our annotation. We see no significant relationship between the frequency with which an AN appears and the entailment judgements we received.

5.3 Final design decisions

As a result of the above pilot experiments, we proceed with our study as follows. First, we use only artificial contexts, as we believe this will result in a greater variety of entailment relations and will avoid systematically biasing our judgements toward entailments. Second, we use the most frequent AN pairs, as these will better represent the types of ANs that NLU systems are likely to encounter in practice.

We look at four different corpora capturing four different genres: Annotated Gigaword (Napoles et al., 2012) (**News**), image captions (Young et al., 2014) (**Image Captions**), the Internet Argument Corpus (Walker et al., 2012) (**Forums**), and the prose fiction subset of GutenTag dataset (Brooke et al., 2015) (**Literature**). From each corpus, we select the 100 nouns which occur with the largest number of unique adjectives. Then, for each noun, we take the 10 adjectives with which the noun occurs most often. For each AN, we choose 3 contexts[2] in which the N appears unmodified, and generate p/h pairs by inserting the A into each.

We collect 3 judgements for each p/h pair. Since this task is subjective, and we want to focus our analysis on clean instances on which human agreement is high, we remove pairs for which one or more of the annotators chose the "does not make sense" option and pairs for which we do not have at least 2 out of 3 agreement (i.e. at least two workers must have chosen the same score on the 5-point scale). In the end, we have a total of 5,560 annotated p/h pairs[3] coming roughly evenly from our 4 genres.

6 Empirical Analysis

Figure 3 shows how the entailment relations are distributed in each genre. In Image Captions, the vast majority of ANs are in a *forward entailment* (restrictive) relation with their head N. In the other genres, however, a substantial fraction (36% for Forums) are in equivalence relations: i.e. the AN denotes the same set as is denoted by the N alone.

When does N entail AN? If it is possible to insert adjectives into a sentence without adding new information, when does this happen? When is adjectival

[2] As a heuristic, we skip sentences containing obvious downward-monotone operators, e.g. *not*, *every* (Section 4).

[3] Our data is available at `http://www.seas.upenn.edu/~nlp/resources/AN-composition.tgz`

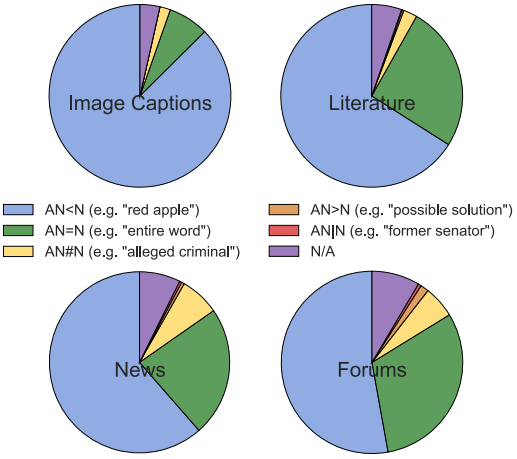


Figure 3: Basic entailment relations assigned to ANs according to the 5,560 p/h pairs our data.

modification not restrictive? Based on our qualitative analysis, two clear patterns stand out:

1) **When the adjective is prototypical of the noun it modifies.** In general, we see that adding adjectives which are seen as attributes of the "prototypical" instance of the noun tend to generate entailments. E.g. people are generally comfortable concluding that *beach→sandy beach*. The same adjective may be prototypical and thus entailed in the context of one noun, but generate a contradiction in the context of another. E.g. if someone has a *baby*, it is probably fine to say they have a *little baby*, but if someone has *control*, it would be a lie to say they have *little control* (Figure 4).[4]

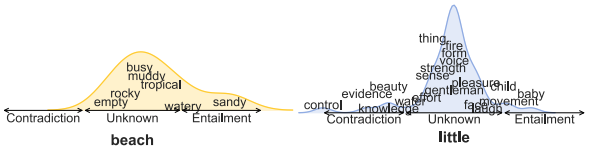


Figure 4: Inserting adjectives that are seen as "prototypical" of the noun tends to generate entailments. E.g., *beach* generally entails *sandy beach*.

2) **When the adjective invokes a sense of salience or importance.** Nouns are assumed to be salient and relevant. E.g. *answers* are assumed (perhaps naively) to be *correct*, and *problems* are assumed (perhaps melodramatically) to be *current* and *huge*. Inserting adjectives like *false* or *empty* tend to generate contradictions (Figure 5).

What do the different natural logic relations look like in practice? Table 3 shows examples of ANs and

[4]These curves show the distribution over entailment scores associated with the INS(A) operation. Yellow curves show, for a single N, the distribution over all the As that modify it. Blue curves show, for a single A, the distribution over all the Ns it modifies.

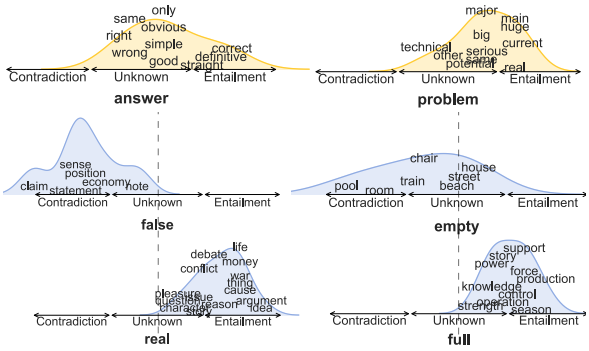


Figure 5: Unless otherwise specified, nouns are considered to be salient and relevant. *Answers* are assumed to be *correct*, and *problems* to be *current*.

contexts exhibiting each of the basic entailment relations. Some entailment inferences depend entirely on contextual information (Example 2a) while others arise from common-sense inference (Example 2b). Many of the most interesting examples fall into the *independence* relation. Recall from Section 3 that independence, in theory, covers ANs such as *alleged criminal*, in which the AN may or may not entail the N. In practice, the cases we observe falling into the independence relation tend to be those which are especially effected by world knowledge. In Example 3, *local economy* is considered to be independent of *economy* when used in the context of *President Obama*: i.e. the assumption that the president would be discussing the national economy is so strong that even when the president says *the local economy is improving*, people do not take this to mean that he has said *the economy is improving*.

Undefined entailment relations. Our annotation methodology– i.e. inferring entailment relations based on the entailments generated by INS and DEL edits– does not enforce that all of the ANs fit into one of the five entailment relations defined by natural logic. Specifically, we observe many instances (∼5% of p/h pairs) in which INS is determined to generate a contradiction, while DEL is said to generate an entailment. In terms of set theory, this is equivalent to the (non-sensical) setting in which "every AN is an instance of N, but no N is an instance of AN." On inspection, these again represent cases in which common-sense assumptions dominate the inference. In Example 6, when given the premise *Bush travels to Michigan to discuss the economy*, annotators are confident enough that *economy* does not entail *Japanese economy* (why on earth would Bush travel to Michigan to discuss the Japanese economy?) that they label the insertion of *Japanese* as generating a contradiction. However, when presented with the p/h in the opposite direction, annotators agree that the *Japanese economy* does indeed entail the *economy*. These examples highlight the flexibility with which humans perform natural language inference, and the need for automated systems to

(1)	AN $\sqsubset$ N	He underwent a [successful] operation on his leg at a Lisbon hospital in December.
(2a)	AN $\equiv$ N	The [deadly] attack killed at least 12 civilians.
(2b)	AN $\equiv$ N	The [entire] bill is now subject to approval by the parliament.
(3)	AN # N	President Obama cited the data as evidence that the [local] economy is improving.
(4)	AN $\sqsupset$ N	The [militant] movement was crushed by the People's Liberation Army.
(5)	AN | N	Red numbers spelled out their [perfect] record: 9-2.
(6)	AN ? N	Bush travels Monday to Michigan to make remarks on the [Japanese] economy.

Table 3: Examples of ANs in context exhibiting each of the different entailment relations. Note that these are "artificial" contexts (Section 5.2), meaning the adjective was not originally a part of the sentence.

be equally flexible.

Take aways. Our analysis in this section results in three key conclusions about AN composition. 1) Despite common assumptions, adjectives do not always restrict the denotation of a noun. Rather, adjectival modification can result in a range of entailment relations, including equivalence and contradiction. 2) There are patterns to when the insertion of an adjective is or is not entailment-preserving, but recognizing these patterns requires common-sense and a notion of "prototypical" instances of nouns. 3) The entailment relation that holds between an AN and the head N is highly context dependent. These observations describe sizable obstacles for automatic NLU systems. Common-sense reasoning is still a major challenge for computers, both in terms of how to learn world knowledge and in how to represent it. In addition, context-sensitivity means that entailment properties of ANs cannot be simply stored in a lexicon and looked-up at run time. Such properties make AN composition an important problem on which to focus NLU research.

7 Benchmarking Current SOTA

We have highlighted why AN composition is an interesting and likely challenging phenomenon for automated NLU systems. We now turn our investigation to the performance of state-of-the-art RTE systems, in order to quantify how well AN composition is currently handled.

The Add-One Entailment Task. We define the "Add-One Entailment" task to be identical to the normal RTE task, except with the constraint that the premise p and the hypothesis h differ only by the atomic insertion of an adjective: $h = e(p)$ where e=INS(A) and A is a single adjective. To provide a consistent interface with a range of RTE systems, we use a binary label set: NON-ENTAILMENT (which encompasses both CONTRADICTION and UNKNOWN) and ENTAILMENT. We want to test on only straightforward examples, so as not to punish systems for failing to classify examples which humans themselves find difficult to judge. In our test set, therefore, we label pairs with mean human scores ≤ 3 as NON-ENTAILMENT, pairs with scores ≥ 4 as ENTAILMENT, and throw away the pairs which fall into the ambiguous range in between.[5] Our resulting train, dev, and test sets contain 4,481, 510, and 387 pairs, respectively. These splits cover disjoint sets of ANs– i.e. none of the ANs appearing in test were seen in train. Individual adjectives and/or nouns can appear in both train and test. The dataset consists of roughly 85% NON-ENTAILMENT and 15% ENTAILMENT. Inter-annotator agreement achieves 93% accuracy.

7.1 RTE Systems

We test a variety of state-of-the-art RTE systems, covering several popular approaches to RTE. These systems are described in more detail below.

Classifier-based. The Excitement Open RTE platform (Magnini et al., 2014) includes a suite of RTE systems, including baseline systems as well as feature-rich supervised systems which provide state-of-the-art performance on the RTE3 datasets (Giampiccolo et al., 2007). We test two systems from Excitement: the simple Maximum Entropy (**MaxEnt**) model which uses a suite of dense, similarity-based features (e.g. word overlap, cosine similarity), and the more sophisticated Maximum Entropy model (**MaxEnt+LR**) which uses the same similarity-based features but additionally incorporates features from external lexical resources such as WordNet (Miller, 1995) and VerbOcean (Chklovski and Pantel, 2004). We also train a standard unigram model (**BOW**).

Transformation-based. The Excitement platform also includes a transformation-based RTE system called **BIUTEE** (Stern and Dagan, 2012). The BIUTEE system derives a sequence of edits that can be used to transform the premise into the hypothesis. These edits are represented using feature vectors, and the system searches over edit sequences for the lowest cost "proof" of either entailment or non-entailment. The feature weights are set by logistic regression during training.

Deep learning. Bowman et al. (2015a) recently reported very promising results using deep learning ar-

[5] For our training and dev sets, we include all pairs, considering scores < 3.5 as NON-ENTAILMENT and scores ≥ 3.5 as ENTAILMENT. We tried removing "ambiguous" pairs from the training and dev sets as well, but it did not improve the systems' performances on the test set.

chitectures and large training data for the RTE task. We test the performance of those same implementations on our Add-One task. Specifically, we test the following models: a basic Sum-of-words model (**Sum**), which represents both p and h as the sum of their word embeddings, an **RNN** model, and an **LSTM** model. We also train a bag-of-vectors model (**BOV**), which is simply a logistic regression whose features are the concatenated averaged word embeddings of p and h.

For the LSTM, in addition to the normal training setting– i.e. training only on the 5K Add-One training pairs– we test a transfer-learning setting (**Transfer**). In transfer learning, the model trains first on a large general dataset before fine-tuning its parameters on the smaller set of target-domain training data. For our Transfer model, we train first on the 500K pair SNLI dataset (Bowman et al., 2015a) until convergence, and then fine-tune on the 5K Add-One pairs. This setup enabled Bowman et al. (2015a) to train a high-performance LSTM for the SICK dataset, which is of similar size to our Add-One dataset (∼5K training pairs).

7.2 Results

Out of the box performances. To calibrate expectations, we first report the performance of each of the systems on the datasets for which they were originally designed. For the Excitement systems, this is the RTE3 dataset (Table 6a). For the deep learning systems, this is the SNLI dataset (Table 6b). For the deep learning systems, in addition to reporting performance when trained on the SNLI corpus (500K p/h pairs), we report the performance in a reduced training setting in which systems only have access to 5K p/h pairs. This is equivalent to the amount of data we have available for the Add-One task, and is intended to give a sense of the performance improvements we should expect from these systems given the size of the training data.

	RTE3
Majority	51.3
BOW	51.0
Edit Dist.	61.9
MaxEnt+LR	63.6
BIUTEE	65.6

(a) Systems from Magnini et al. (2014) on RTE3.

	SNLI 500K / 5K
Majority	65.7
BOV	74.4 / 71.5
RNN	82.1 / 67.0
Sum	85.3 / 69.2
LSTM	86.2 / 68.0

(b) Systems from Bowman et al. (2015a) on SNLI.

Figure 6: Performance of SOTA systems on the datasets for which they were originally developed.

7.3 Performance on Add-One RTE.

Finally, we train each of the systems on the 5,000 Add-One p/h pairs in our dataset and test on our held-out set of 387 pairs. Figure 7 reports the results in terms of accuracy and precision/recall for the ENTAILMENT class. The baseline strategy of predicting the majority class for each adjective, based on the training data, reaches close to human performance (92% accuracy). Given the simplicity of the task (p and h differ by a single word), this baseline strategy should be achievable. However, none of the systems tested come close to this level of performance, suggesting that they fail to learn even the most-likely entailment generated by adjectives (e.g. that INS(*brown*) probably generates NON-ENTAILMENT and INS(*possible*) probably generates ENTAILMENT). The best performing system is the RNN, which achieves 87% accuracy, only two points above the baseline of always guessing NON-ENTAILMENT.

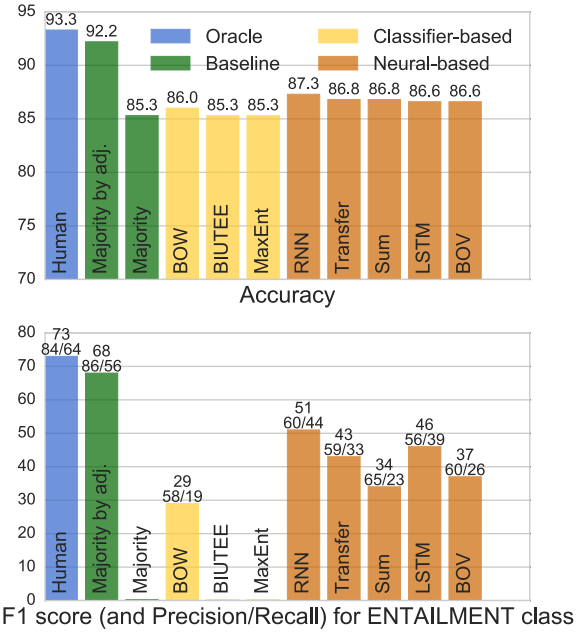


Figure 7: Performances of all systems on AddOne RTE task. The strategy of predicting the majority class for each adjective– based on the training data– reaches near human performance. None of the systems tested come close to human levels, indicating that the systems fail even to memorize the most-likely class for each adjective in training.

8 Related Work

Past work, both in linguistics and in NLP, has explored different classes of adjectives (e.g. privative, intensional) as they relate to entailment (Kamp and Partee, 1995; Partee, 2007; Boleda et al., 2013; Nayak et al., 2014). In general, prior studies have focused on modeling properties of the adjectives alone, ignoring the context-dependent nature of AN/N entailments– i.e. in prior work *little* is always restrictive, whether it is modifying *baby* or *control*. Pustejovsky (2013) offer a preliminary analysis of the contextual complexities surrounding adjective inference, which reinforces many of the observations we have made here. Hartung and Frank (2011) analyze adjectives in terms of the properties they modify but don't address them from an entailment perspective. Tien Nguyen et al. (2014) look at the

adjectives in the restricted domain of computer vision.

Other past work has employed first-order logic and other formal representations of adjectives in order to provide compositional entailment predictions (Amoia and Gardent, 2006; Amoia and Gardent, 2007; McCrae et al., 2014). Although theoretically appealing, such rigid logics are unlikely to provide the flexibility needed to handle the type of common-sense inferences we have discussed here. Distributional representations provide much greater flexibility in terms of representation (Baroni and Zamparelli, 2010; Guevara, 2010; Boleda et al., 2013). However, work on distributional AN composition has so far remained out-of-context, and has mostly been evaluated in terms of overall "similarity" rather than directly addressing the entailment properties associated with composition.

9 Conclusion

We have investigated the problem of adjective-noun composition, specifically in relation to the task of RTE. AN composition is capable of producing a range of natural logic entailment relationship, at odds with commonly-used heuristics which treat all adjectives a restrictive. We have shown that predicting these entailment relations is dependent on context and on world knowledge, making it a difficult problem for current NLU technologies. When tested, state-of-the-art RTE systems fail to learn to differentiate entailment-preserving insertions of adjectives from non-entailing ones. This is an important distinction for carrying out human-like reasoning, and our results reveal important weaknesses in the representations and algorithms employed by current NLU systems. The Add-One Entailment task we have introduced will allow ongoing RTE research to better diagnose systems' abilities to capture these subtleties of ANs, which that have practical effects on natural language inference.

Acknowledgments

This research was supported by a Facebook Fellowship, and by gifts from the Alfred P. Sloan Foundation, Google, and Facebook. This material is based in part on research sponsored by the NSF grant under IIS-1249516 and DARPA under number FA8750-13-2-0017 (the DEFT program). The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes. The views and conclusions contained in this publication are those of the authors and should not be interpreted as representing official policies or endorsements of DARPA and the U.S. Government.

We would like to thank Sam Bowman for helping us to replicate his prior work and ensure a fair comparison. We would also like to thank the anonymous reviewers for thoughtful comments, and the Amazon Mechanical Turk annotators for their contributions.

References

Marilisa Amoia and Claire Gardent. 2006. Adjective based inference. In *Proceedings of the Workshop KRAQ'06 on Knowledge and Reasoning for Language Processing*, pages 20–27. Association for Computational Linguistics.

Marilisa Amoia and Claire Gardent. 2007. A first order semantic approach to adjectival inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 185–192, Prague, June. Association for Computational Linguistics.

Gabor Angeli and Christopher D. Manning. 2014. NaturalLI: Natural logic inference for common sense reasoning. In *Empirical Methods in Natural Language Processing (EMNLP)*, October.

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China, July. Association for Computational Linguistics.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193, Cambridge, MA, October. Association for Computational Linguistics.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 23–32, Avignon, France, April. Association for Computational Linguistics.

Johannes Bjerva, Johan Bos, Rob van der Goot, and Malvina Nissim. 2014. The meaning factory: Formal semantics for recognizing textual entailment and determining semantic similarity. *SemEval 2014*, page 642.

Gemma Boleda, Marco Baroni, Louise McNally, and Nghia Pham. 2013. Intensionality was only alleged: On adjective-noun composition in distributional semantics. In *Proceedings of the 10th International Conference on Computational Semantics*, pages 35–46.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015a. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September. Association for Computational Linguistics.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2015b. Learning distributed word representations for natural logic reasoning. In *2015 AAAI Spring Symposium Series*.

Julian Brooke, Adam Hammond, and Graeme Hirst. 2015. GutenTag: an NLP-driven tool for digital humanities research in the Project Gutenberg corpus. In *Proceedings of the Fourth Workshop on Computational Linguistics for Literature*, pages 42–47, Denver, Colorado, USA, June. Association for Computational Linguistics.

Timothy Chklovski and Patrick Pantel. 2004. VerbOcean: Mining the web for fine-grained semantic verb relations. In *EMNLP*, volume 2004, pages 33–40.

Robin Cooper, Dick Crouch, Jan Van Eijck, Chris Fox, Johan Van Genabith, Jan Jaspars, Hans Kamp, David Milward, Manfred Pinkal, Massimo Poesio, et al. 1996. A framework for computational semantics. Technical report, Technical Report LRE 62-051 D-16, The FraCaS Consortium.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The PASCAL recognizing textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190. Springer.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pages 1–9, Prague, June. Association for Computational Linguistics.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 33–37, Uppsala, Sweden, July. Association for Computational Linguistics.

Matthias Hartung and Anette Frank. 2011. Exploring supervised LDA models for assigning attributes to adjective-noun phrases. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 540–551, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Nirit Kadmon. 2001. *Formal Pragmatics: Semantics, Pragmatics, Presupposition, and Focus. Willey*. Blackwell. Oxford.

Hans Kamp and Barbara Partee. 1995. Prototype theory and compositionality. *Cognition*, 57(2):129–191.

Bill MacCartney and Christopher D. Manning. 2008. Modeling semantic containment and exclusion in natural language inference. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 521–528.

Bill MacCartney. 2009. *Natural language inference*. Ph.D. thesis, Citeseer.

Bernardo Magnini, Roberto Zanoli, Ido Dagan, Kathrin Eichler, Guenter Neumann, Tae-Gil Noh, Sebastian Padó, Asher Stern, and Omer Levy. 2014. The Excitement Open Platform for textual inferences. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 43–48, Baltimore, Maryland, June. Association for Computational Linguistics.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223, Reykjavik, Iceland, May. ACL Anthology Identifier: L14-1314.

John P. McCrae, Francesca Quattri, Christina Unger, and Philipp Cimiano. 2014. Modelling the semantics of adjectives in the ontology-lexicon interface. In *Proceedings of the 4th Workshop on Cognitive Aspects of the Lexicon (CogALex)*, pages 198–209, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

George A. Miller. 1995. WordNet: A lexical database for English. *Communications of the ACM*, 38(11):39–41, November.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated gigaword. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 95–100.

Neha Nayak, Mark Kowarsky, Gabor Angeli, and Christopher D. Manning. 2014. A dictionary of nonsubsective adjectives. Technical Report CSTR 2014-04, Department of Computer Science, Stanford University, October.

Sebastian Padó, Tae-Gil Noh, Asher Stern, Rui Wang, and Roberto Zanoli. 2014. Design and realization of a modular architecture for textual entailment. *Journal of Natural Language Engineering*.

Barbara Partee. 2007. Compositionality and coercion in semantics: The dynamics of adjective meaning. *Cognitive foundations of interpretation*, pages 145–161.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. 2015. Adding semantics to data-driven paraphrasing. In *Association for Computational Linguistics*, Beijing, China, July. Association for Computational Linguistics.

James Pustejovsky. 2013. Inference patterns with intensional adjectives. In *Proceedings of the 9th Joint ISO - ACL SIGSEM Workshop on Interoperable Semantic Annotation*, pages 85–89, Potsdam, Germany, March. Association for Computational Linguistics.

Asher Stern and Ido Dagan. 2012. BIUTEE: A modular open-source system for recognizing textual entailment. In *Proceedings of the ACL 2012 System Demonstrations*, pages 73–78, Jeju Island, Korea, July. Association for Computational Linguistics.

Dat Tien Nguyen, Angeliki Lazaridou, and Raffaella Bernardi. 2014. Coloring objects: Adjective-noun visual semantic compositionality. In *Proceedings of the Third Workshop on Vision and Language*, pages 112–114, Dublin, Ireland, August. Dublin City University and the Association for Computational Linguistics.

Marilyn A. Walker, Jean E. Fox Tree, Pranav Anand, Rob Abbott, and Joseph King. 2012. A corpus for research on deliberation and debate. In *LREC*, pages 812–817.

Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. 2014. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics (TACL)*, 2(Feb):67–78.

Modeling Stance in Student Essays

Isaac Persing and **Vincent Ng**
Human Language Technology Research Institute
University of Texas at Dallas
Richardson, TX 75083-0688
{persingq,vince}@hlt.utdallas.edu

Abstract

Essay stance classification, the task of determining how much an essay's author agrees with a given proposition, is an important yet under-investigated subtask in understanding an argumentative essay's overall content. We introduce a new corpus of argumentative student essays annotated with stance information and propose a computational model for automatically predicting essay stance. In an evaluation on 826 essays, our approach significantly outperforms four baselines, one of which relies on features previously developed specifically for stance classification in student essays, yielding relative error reductions of at least 11.3% and 5.3%, in micro and macro F-score, respectively.

1 Introduction

State-of-the-art automated essay scoring engines such as E-*rater* (Attali and Burstein, 2006) do not grade essay content, focusing instead on providing diagnostic trait feedback on categories such as grammar, usage, mechanics, style and organization. Hence, persuasiveness and other content-dependent dimensions of argumentative essay quality are largely ignored in existing automated essay scoring research. While full-fledged content-based essay scoring is still beyond the reach of state-of-the-art essay scoring engines, recent work has enabled us to move one step closer to this ambitious goal by analyzing essay content, attempting to determine the argumentative structure of student essays (Stab and Gurevych, 2014) and the persuasiveness of the arguments made in these essays (Persing and Ng, 2015).

Stance classification is an important first step in determining how persuasive an argumentative student essay is because persuasiveness depends on how well the author argues *w.r.t. the stance she takes* using the supporting evidence she provides. For instance, if her stance is *Agree Somewhat*, a persuasive argument would involve explaining what reservations she has about the given proposition. As another example, an argumentative essay in which the author takes a *neutral* stance or the author presents evidence that does not support the stance she claims to take should receive a low persuasiveness score.

Given the important role played by stance classification in determining an essay's persuasiveness, our goal in this paper is to examine stance classification in argumentative student essays. While there is a large body of work on stance classification[1], stance classification in argumentative essays is largely under-investigated and is different from previous work in several respects. First, in automated essay grading, the majority of the essays to be assessed are written by students who are learners of English. Hence our stance classification task could be complicated by the authors' lack of fluency in English. Second, essays are longer and more formally written than the text typically used in previous stance classification research (e.g., debate posts). In particular, a student essay writer typically expresses her stance on the essay's topic in a thesis sentence/clause, while a debate post's author may never even explicitly express her stance. Although the explicit expression of stance in essays seems to make our task easier,

[1]Previous approaches to stance classification have focused on three discussion/debate settings, namely congressional floor debates (Thomas et al., 2006; Bansal et al., 2008; Balahur et al., 2009; Yessenalina et al., 2010; Burfoot et al., 2011), company-internal discussions (Agrawal et al., 2003; Murakami and Raymond, 2010), and online social, political, and ideological debates (Wang and Rosé, 2010; Biran and Rambow, 2011; Walker et al., 2012; Abu-Jbara et al., 2013; Hasan and Ng, 2013; Boltužić and Šnajder, 2014; Sobhani et al., 2015; Sridhar et al., 2015).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2174–2184,
Berlin, Germany, August 7-12, 2016.

Prompt	Prompt Parts
Most university degrees are theoretical and do not prepare students for the real world. They are therefore of very little value.	1) Most university degrees are theoretical. 2) Most university degrees do not prepare students for the real world. 3) Most university degrees are of very little value.
The prison system is outdated. No civilized society should punish its criminals: it should rehabilitate them.	1) The prison system is outdated. 2) No civilized society should punish its criminals. 3) Civilized societies should rehabilitate criminals.

Table 1: Some examples of essay prompts and their associated parts.

identifying stancetaking text in the midst of non-stancetaking sentences in a potentially long essay, as we will see, is by no means a trivial task.

To our knowledge, the essay stance classification task has only been attempted by Faulkner (2014). However, the version of the task we address is different from his. First, Faulkner only performed two-class stance classification: while his corpus contains essays labeled with *For* (Agree), *Against* (Disagree), and *Neither*, he simplified the task by leaving out the arguably most difficult-to-identify stance, *Neither*. In contrast, we perform *fine-grained* stance classification, where we allow essay stance to take one of six values: *Agree Strongly*, *Agree Somewhat*, *Neutral*, *Disagree Somewhat*, *Disagree Strongly*, and *Never Addressed*, given the practical need to perform fine-grained stance classification in student essays, as discussed above. Second, given that many essay prompts are composed of multiple simpler propositions (e.g., the prompt "Most university degrees are theoretical and do not prepare students for the real world" has two parts, "Most university degrees are theoretical" and "Most university degrees do not prepare students for the real world."), we manually split such prompts into *prompt part*s and determine the stance of the author w.r.t. each part, whereas Faulkner assigned an overall stance to a given prompt regardless of whether it is composed of multiple propositions. The distinction is important because an analysis of our annotations described in Section 2 shows that essay authors take different stances w.r.t. different prompt parts in 49% of essays, and in 39% of essays, authors even take stances with different polarities w.r.t. different prompt parts.

In sum, our contributions in this paper are two-fold. First, we propose a computational model for essay stance classification that outperforms four baselines, including our re-implementation of Faulkner's approach. Second, in order to stimulate further research on this task, we make our annotations publicly available. Since progress on this task is hindered in part by the lack of a publicly annotated corpus, we believe that our data set will be a valuable resource for the NLP community.

2 Corpus

We use as our corpus the 4.5 million word International Corpus of Learner English (ICLE) (Granger et al., 2009), which consists of more than 6000 essays on a variety of different topics written by university undergraduates from 16 countries and 16 native languages who are learners of English as a Foreign Language. 91% of the ICLE texts are written in response to prompts that trigger argumentative essays, and thus are expected to take a stance on some issue. We select 11 such prompts, and from the subset of argumentative essays written in response to them, we select 826 essays to annotate for training and testing our stance classification system.[2] Table 1 shows two of the 11 topics selected for annotation.

We pair each of the 826 essays with each of the prompt parts to which it responds, resulting in 1,593 *instance*s.[3] We then familiarize two human annotators, both of whom are native speakers of English, with the stance definitions in Table 2 and ask them to assign each instance the stance label they believe the essay's author would have chosen if asked how strongly she agrees with the prompt part. We additionally furnish the annotators with descriptions of situations that might cause an author to select the more ambiguous classes. For example, an author might choose Agree Somewhat if she appears to mostly agree with the prompt part, but qualifies her opinion in a way that is not captured by the prompt part's bluntness (e.g. an author who claims the prison system in a lot of countries is outdated would Agree Somewhat with the first part of Table 1's second prompt). Or she may choose Disagree Somewhat if she appears to dis-

[2] See our website at `http://www.hlt.utdallas.edu/~persingq/ICLE/` for the complete list of essay stance annotations.

[3] We do not segment the essays' texts according to which prompt part is being responded to. Each (entire) essay is viewed as a response to all of its associated prompt parts.

Stance	Definition
Agree Strongly (885)	The author seems to agree with and care about the claim.
Agree Somewhat (148)	The author generally agrees with the claim, but might be hesitant to choose "Agree Strongly".
Neutral (28)	The author agrees with the claim as much as s/he disagrees with it.
Disagree Somewhat (91)	The author generally disagrees with the claim, but might be hesitant to choose "Disagree Strongly".
Disagree Strongly (416)	The author seems to disagree with and care about the claim.
Never Addressed (25)	A stance cannot be inferred because the proposition was never addressed.

Table 2: Stance label counts and definitions.

agree with the prompt part, but mentions the disagreement only in passing because she does not care much about the topic.

To ensure consistency in annotation, we randomly select 100 essays (187 instances) for annotation by both annotators. Their labels agree in 84.5% of the instances, yielding a Cohen's (1960) Kappa of 0.76. Each case of disagreement is resolved through discussion between the annotators.

3 Baseline Stance Classification Systems

In this section, we describe four baseline systems.

3.1 Agree Strongly Baseline

Given the imbalanced stance distribution shown in Table 2, we create a simple but by no means weak baseline, which predicts that every instance has most frequent class label (Agree Strongly), regardless of the prompt part or the essay's contents.

3.2 N-Gram Baseline

Previous work on stance classification, which assumes that stance-annotated training data is available for every topic for which stance classification is performed, has shown that the N-Gram baseline is a strong baseline. Not only is this assumption unrealistic in practice, but it has led to undesirable consequences. For instance, the proposition "feminists have done more harm to the cause of women than good" elicits much more disagreement than normal. So, if instances from this proposition appeared in both the training and test sets, the unigram feature "feminist" would be strongly correlated with the disagreement classes even though intuitively it tells us nothing about stance. This partly explains why the N-Gram baseline was strong in previous work (Somasundaran and Wiebe, 2010). In light of this problem, we perform leave-one-out cross validation where we partition the instances by *prompt*, leaving the instances created for one prompt out in each test set.

To understand how strong n-grams are when evaluated in our leave-one-prompt-out cross-validation setting, we employ them as features in our second baseline. Specifically, we train a multiclass classifier on our data set using a feature set composed solely of unigram, bigram, and trigram features, each of which indicates the number of times the corresponding n-gram is present in the associated essay.

3.3 Duplicated Faulkner Baseline

While it is true that no system exists for solving our exact problem, the system proposed by Faulkner (2014) comes fairly close. Hence, as our third baseline, we train a multiclass classifier on our data set for fine-grained essay stance classification using the two types of features proposed by Faulkner, as described below.

Part-of-speech (POS) generalized dependency subtrees. Faulkner first constructs a lexicon of stance words in the style of Somasundaran and Wiebe (2010). The lexicon consists of (1) the set of stemmed first unigrams appearing in all stance-annotated text spans in the Multi-Perspective Question Answering (MPQA) corpus (Wiebe et al., 2005), and (2) the set of boosters (clearly, decidedly), hedges (claim, estimate), and engagement markers (demonstrate, evaluate) from the appendix of Hyland (2005). He then manually removes from this list any words that appear not to be stancetaking, resulting in a 453 word lexicon.

Stance words target propositions, which Faulkner notes, usually contain some opinion-bearing language that can serve as a proxy for the targeted proposition. In order to find the locations in an essay where a stance is being taken, he first finds each stance word in the essay. Then he finds the shortest path from the stance word to an opinion word in the sentence's dependency tree, using the MPQA subjectivity lexicon of opinion words (Wiebe et al., 2005). If this nearest opinion word appears in the stance word's immediate or embedded clause, he creates a binary feature by concatenating all the words in the dependency path, POS generalizing all words other than the stance and opinion word, and finally prepending

"not" if the stance word is adjacent to a negator in the dependency tree. Thus given the sentence "I **can** only say that this statement is completely *true*." he would add the feature *can-V-true*, which suggests agreement with the prompt.

Prompt topic words. Recall that for the previous feature type, a feature was generated whenever an opinion word occurred in a stance word's immediate or embedded clause. Each content word in this clause is used as a binary feature if its similarity with one of the prompt's content words meets an empirically determined threshold.

3.4 N-Gram+Duplicated Faulkner Baseline

To build a stronger baseline, we employ as our fourth baseline a classifier trained on both n-gram features and duplicated Faulkner's features.

4 Our Approach

Our approach to stance classification is a learning-based approach where we train a multiclass classifier using four types of features: n-gram features (Section 3.2), duplicated Faulkner's features (Section 3.3), and two novel types of features, stancetaking path-based features (Section 4.1) and knowledge-based features (Section 4.2).

4.1 Stancetaking Path-Based Features

Recall that, in order to identify his POS generalized dependency subtrees, Faulkner relies on two lexica, a lexicon of stancetaking words and a lexicon of opinion-bearing words. He then extracts a feature any time words from the two lexica are syntactically close enough. A major problem with this approach is that the lexica are so broad that nearly 80% of sentences in our corpus contain text that can be identified as stancetaking using this method. Intuitively, an essay may state its stance w.r.t. a prompt part in a thesis or conclusion sentence, but most of essay's text will be at most tangentially related to any particular prompt part. For this reason, we propose to identifying stancetaking text to target *only text that appears directly related to the prompt part*. Below we first show how we identify and stance-labeling relevant stancetaking dependency paths, and then describe the features we derive from these paths.

4.1.1 Identifying relevant stancetaking paths

As noted above, we first identify stancetaking text that appears directly related to the prompt part.

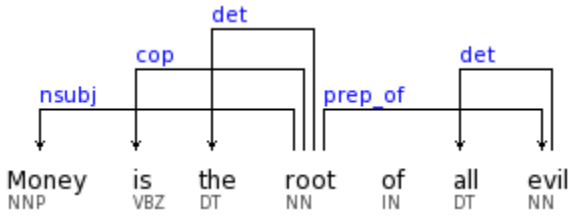


Figure 1: Automatic dependency parse of a prompt part.

To begin, we note that the prompt parts themselves must express a stance on a topic if they can be agreed or disagreed with. By examining the dependency parses[4] of the prompt parts, we can recognize elements of how stancetaking text is structured. From the prompt part shown in Figure 1, for example, we notice that the important words that express a stance in the sentence are "money", "root", and "evil". By analyzing the dependency structure in this and other prompt parts, we discovered that stancetaking text often consists of (1) a **subject** word, which is the child in an nsubj or nsubjpass relation, (2) a **governor** word which is the subject's parent, and (3) an **object**, which is a content word from which there is a (not always direct) dependency path from the governor. We therefore abstract a stance in an essay as a dependency path from a subject to an object that passes through the governor. Thus, the stancetaking dependency path we identify from the prompt part shown in Figure 1 could be represented as money-root-evil.

The obvious problem with identifying stancetaking text in this way is that nearly all sentences contain this kind of stancetaking structure, and just as with Faulkner's dependency paths, there is little reason to believe that any particular path is relevant to an instance's prompt part. Does this mean that nearly all sentences are stancetaking? We would argue that they can be, as even sentences that appear on their face to be mere statements of fact with no apparent value judgment can be viewed as taking a stance on the factuality of the statement, and people often disagree about the factuality of statements. For this reason, after we have identified a stancetaking path, we must determine whether the stance being expressed is relevant to the prompt part before extracting features from it.

[4]Dependency parsing, POS tagging, and lemmatization are performed automatically using the Stanford CoreNLP system (Manning et al., 2014)

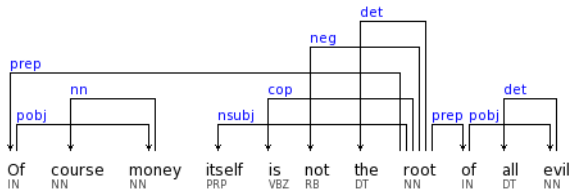


Figure 2: Automatic dependency parse of an essay sentence.

For this reason, we ignore all stancetaking paths that do not meet the following three relevance conditions. First, the lemma of the path's governor must match the lemma of a governor in the prompt part. Second, the lemma of the path's object must match the lemma of some content word[5] in the prompt part. Finally, the containing sentence must not contain a question mark or a quotation, as such sentences are usually rhetorical in nature. We do not require that the subject word match the prompt part's subject word because this substantially reduces coverage for various reasons. For one, of the three words (subject, governor, object), the subject is the word most likely to be replaced with some other word like a pronoun, and possibly because the essays were written by non-native English speakers, automatic coreference resolution cannot reliably identify these cases. We also do not fully trust that the subject identified by the dependency parser will reliably match the subject we are looking for. Given these constraints, we can automatically identify the "itself-root-of-evil" dependency path in Figure 2 as a relevant stancetaking path.

4.1.2 Stance-labeling the paths

Next, we determine whether a stancetaking path identified in the previous step appears to agree or disagree with the prompt part.

To begin, we count the number of negations occurring in the prompt part. Any word like "no", "not", or "none" counts as a negation unless it begins a non-negation phrase like "no doubt" or "not only".[6] Thus, the count of negations in the prompt part in Figure 1 is 0.

After that, we count the number of times the identified stancetaking path is negated. Because these paths occur in student essays and are therefore often not as simply-stated as the prompt parts, this is a little bit more complicated than just counting the containing sentence's negations since the sentence may contain a lot of additional material. To do this, we construct a list of all the dependency nodes in the stancetaking path as well as all of their dependency tree children. We then remove from this list any node that, in the sentence, occurs after the last node in the stancetaking path. The total negation count we are looking for is the number of nodes in this list that correspond to negation words (unless the negation word begins a negation phrase). Thus, because the word "not" is the child of "root" in the path "itself-root-of-evil" we identified in Figure 2, we consider this path to have been negated one time.

Finally, we sum the prompt part negations and the stancetaking path negations. If this sum is even, we believe that the relevant stancetaking path agrees with the prompt part in the instance. If it is odd, however (as in the case of the prompt part and stancetaking text in the dependency tree figures), we believe that it disagrees with the prompt part. To illustrate why we are concerned with whether this sum is even, consider the following examples. If both the prompt part and the stancetaking text are negated, both disagree with the opposite of the prompt part's stance. Thus, they agree with each other, and their negation sum is even (2). If the stancetaking path was negated twice, however, the sum would be odd (3) due to the stance path's double negations canceling each other out, and the stancetaking path would disagree with the prompt part.

4.1.3 Deriving path-based features

We extract four features from the relevant stancetaking dependency paths identified and stance-labeled so far, as described below.

The first feature encodes the count of relevant stancetaking paths that appear to agree with the prompt part. The second feature encodes the count of relevant stancetaking paths that appear to disagree with the prompt part. While we expect these first two features to be correlated with the agreement and disagreement classes, respectively, they may not be sufficient to distinguish between agreeing and disagreeing instances. It is possible, for example, that both features may be greater than zero in a single instance if we have identified one stancetaking path that appears to agree

[5]For our purpose, a content word (1) is a noun, pronoun, verb, adjective, or adverb, (2) is not a stopword, and (3) is at the root, is a child in a dobj or pobj relation, or is the child in a conj relation whose parent is the child in a dobj or pobj relation in the dependency tree.

[6]See our website at `http://www.hlt.utdallas.edu/~persingq/ICLE/` for our list of manually constructed negation words and non-negation phrases.

with the prompt part and another stancetaking path that appears to disagree with the prompt part. It is not clear whether this situation is indicative of only the Neutral class, or perhaps it indicates partial (Somewhat) (Dis)Agreement, or maybe our method of detecting disagreement is not reliable enough, and it therefore makes sense, when we get these conflicting signals, to ignore them entirely and just assign the instance to the most frequent (Agree Strongly) class. For that matter, if neither feature is greater than zero, does this mean that the instance Never Addressed the prompt part, or does it instead mean that our method for identifying stancetaking paths doesn't have high enough recall to work on all instances? We let our learner sort these problems out by adding two more binary features to our instances, one which indicates that both of the first two features are zero, and one that indicates whether both are greater than zero.

4.2 Knowledge-Based Features

Our second feature type is composed of five linguistically informed binary features that correspond to five of the six classes in our fine-grained stance classification task. Intuitively, if an instance has one of these features turned on, it should be assigned to the feature's corresponding class.

1. Neutral. Stancetaking text indicating neutrality tends to be phrased somewhat differently than stancetaking text indicating any other class. In particular, neutral text often makes claims that are about the prompt part's subject, but which are tangential to the proposition expressed in the prompt part. For this reason, we search the essay for words that match the prompt part's subject lemmatically.

After identifying a sentence that is about the prompt part's subject in this way, we check whether the sentence begins with any neutral indicating phrase.[7] If we find a sentence that both begins with a neutral phrase and is about the prompt part's subject, we turn the Neutral feature on. Thus, sentences like the following can be captured: "*In all probability* university students wonder whether or not they spend their time uselessly in studying through four or five years in order to take their *degree*."

2. (Dis)Agree Somewhat. In order to set the values of the features associated with the Somewhat classes, we first identify relevant stancetaking paths as described above. We then trim the list of paths by removing any path whose governor or subject does not have a hedge word as an adverb modifier child in the dependency tree.[8] Thus, we are able to determine that the essay containing the sentence "There is *nearly* no place left for dream and imagination" is likely to belong to one of the Somewhat classes w.r.t. the prompt part "There is no longer a place for dreaming and imagination."

The question now is how to determine which (if any) of the Somewhat classes it should belong to. We analyze all the paths from the list for negation in much the same way we described above, but with one major difference. We hypothesize that when taking a Somewhat stance, students are more likely to explicitly state that the stance being taken is their opinion rather than stating the stance bluntly without attribution. For example, one Disagree Somewhat essay includes the sentence, "*I* **never** *believed* these people were honest if saying that money is just the root of all evil." In order to determine that this sentence contains an indication of the Disagree Somewhat class, we need to account for the negation that occurs at the beginning, far away from the stancetaking path (money-root-of-evil). To do this, we semantically parse the sentence using SEMAFOR (Das et al., 2010). Each of the *semantic frames* detected by SEMAFOR describes an event that occurs in a sentence, and the event's *frame elements* may be the people or other entities that participate in the event. One of the semantic frames detected in this example sentence describes a Believer (I) and the content of his or her belief (all the text after "believed"). Because the sentence includes a semantic frame that (1) contains a first person (I, we) Cognizer, Speaker, Perceiver, or Believer element, (2) contains an element that covers all the text in the dependency path (a Content frame element, in this case), and (3) the word that triggers the frame ("believed") has a negator child in the dependency tree, we add one to this relevant stancetaking path's negation count. This makes this hedged stancetaking path's negation count odd, so we believe that this sentence likely disagrees with its instance's prompt part somewhat. If we find a hedged stancetaking

[7] We construct a list of neutral phrases for introducing another person's ideas from a writing skills website (`http://www.myenglishteacher.eu/question/other-ways-to-say-according-to/`).

[8] See our website at `http://www.hlt.utdallas.edu/~persingq/ICLE/` for our manually constructed list of hedge words.

path with an odd negation count, we turn on the Disagree Somewhat feature. Similarly, if we find a hedged stancetaking path with an even negation count, we turn on the Agree Somewhat feature.

3. (Dis)Agree Strongly. When we believe there is strong evidence that an instance should belong to one of the Strongly classes, we turn on the corresponding (Dis)Agree Strongly feature. In particular, if we find a relevant stancetaking path that appears to agree with the prompt part (as described in Section 4.1.2), but do not find any such path that appears to disagree with it, we turn on the Agree Strongly feature. Similarly, if we find a relevant stancetaking path that appears to disagree with the prompt part, but do not find a relevant stancetaking path that appears to agree with it, we turn on the Disagree Strongly feature.

5 Evaluation

5.1 Experimental Setup

Data partition. All our results are obtained via leave-one-prompt-out cross-validation experiments. So, in each fold experiment, we partition the instances from our 11 prompts into a training set (10 prompts) and a test set (1 prompt).

Evaluation metrics. We employ two metrics to evaluate our systems: (1) micro F-score, which treats each instance as having equal weight; and (2) macro F-score, which treats each class as having equal weight.[9] To gain insights into how different systems perform on different classes, we additionally report per-class F-scores.

Training. We train the baselines and our approach using two learning algorithms, MALLET's (McCallum, 2002) implementation of maximum entropy (MaxEnt) classification and our own implementation of the one nearest neighbor (1NN) algorithm using the cosine similarity metric. Note that these two learners have their own strengths and weaknesses: in comparison to 1NN, MaxEnt is better at exploiting high-dimensional features but less robust to skewed class distributions. For the baseline systems, we select the learner by performing cross validation on the training folds to maximize the average of micro and macro F-scores in each fold experiment.

When training our approach, we perform exhaustive feature selection to determine which subset of the four sets of features (i.e., n-gram, duplicated Faulkner, path-based, and knowledge-based features) should be used. Specifically, we select the feature groups and learner jointly by performing cross validation on the training folds, choosing the combination yielding the highest average of micro and macro F-scores in each fold experiment. To prevent any feature type from dominating the others, to each feature we apply a weight of one divided by the number of features having its type.

Testing. In case of a tie when applying 1NN, the tie is broken by selecting the class appearing higher in Table 2.

5.2 Results and Discussion

Results on fine-grained essay stance classification are shown in Table 3. The first four rows show our baselines' performances. Among the four baselines, Always Agree Strongly performs best w.r.t. micro F-score, obtaining a score of 55.6%, whereas Duplicated Faulkner performs best w.r.t. macro F-score, obtaining a score of 15.6%. Despite its poor performance, Duplicated Faulkner is a state-of-the-art approach on this task. Its poor performance can be attributed to three major factors. First, it was intended to identify only Agree and Disagree instances (note that Faulkner simply removed neutral instances from his experimental setup), which should not prevent them from performing well w.r.t. micro F-score. Second, it is far too permissive, generating features from a large majority of sentences while relevant sentences are far rarer. Third, while it does succeed at predicting Disagree Strongly far more frequently than either of the other baselines that excludes the Faulkner feature set, the problem's class skewness means that a learner is much more likely to be punished for predicting minority classes, which are more difficult to predict with high precision.

The fact that it makes an attempt to solve the problem rather than relying on class skewness for good performance makes Duplicated Faulkner a more interesting baseline than either N-Gram or Always Agree Strongly, even though both technically outperform it w.r.t. micro F-score. Similarly, the statistically significant improvements in micro and macro F-score our approach achieves over the best baselines are more impressive when taking the skewness problem into consideration.

The results of our approach, which has access

[9] Since stance classification is a multiclass, single-label task, micro F-score, precision, recall, and accuracy are all equivalent.

	System	Micro-F	Macro-F	A+	A−	Neu	D−	D+	Nev
1	Always Agree Strongly	55.6	11.9	71.4	.0	.0	.0	.0	.0
2	N-Gram	55.4	12.0	71.3	.0	.0	.0	.5	.0
3	Duplicated Faulkner	50.8	15.6	66.8	4.0	.0	.0	22.9	.0
4	N-Gram + Duplicated Faulkner	53.4	15.4	69.1	2.5	.0	.0	20.6	.0
5	Our approach	**60.6**	**20.1**	73.6	.0	.0	2.1	44.8	.0

Table 3: Cross-validation results for fine-grained essay stance classification, including per-class F-scores for Agree Strongly (A+), Agree Somewhat (A−), Neutral (Neu), Disagree Somewhat (D−), Disagree Strongly (D+), and Never Addressed (Nev).

to all four feature groups, are shown in row 5 of the table. It obtains micro and macro F-scores of 60.6% and 20.1%, which correspond to statistically significant relative error reductions over the best baselines of 11.3% and 5.3%, respectively.[10]

Recall that we turned on one of our knowledge-based features only when we believed there was strong evidence that an instance belonged to its associated class. To get an idea of how useful these features are, we calculate the precision, recall, and F-score that would be obtained for each class if we treated our knowledge-based features as heuristic classifiers. The respective precisions, recalls, and F-scores we obtained are: 0.66/0.28/0.40 (A+), 0.50/0.02/0.04 (A−), 0.00/0.00/0.00 (Neu), 0.50/0.01/0.02 (D−), and 0.63/0.31/0.42 (D+). Since the rule predictions are encoded as features for the learner, they may not necessarily be used by the learner even if the underlying rules are precise. For instance, despite the rule's high precision on the Agree Somewhat class, the learner did not make use of its predictions due to its low coverage.

5.3 Additional Experiments

Since all the systems we examined fared poorly on identifying Somewhat classes, one may wonder how these systems would perform if we considered a simplified version of the task where we merged each Somewhat class with the corresponding Strongly class. In particular, since Faulkner's approach was originally not designed to distinguish between Strongly and Somewhat classes, it may seem fairer to compare our approach against Duplicated Faulkner on the four-class essay stance classification task, where stance can take one of four values: Agree (created by merging Agree Strongly and Agree Somewhat), Disagree (created by merging Disagree Strongly and Disagree Somewhat), Neutral, and Never Addressed.

In the results for the different systems on this four-class stance classification task, shown in Table 4, we see that the same patterns we noticed in the six-class version of the task persist. The approaches' relative order w.r.t. micro and macro F-score remains the same, though they are adjusted upwards due to the problem's increased simplicity. Our approach's performance on Agree increases (compared to Agree Strongly) because Agree is a bigger class, making predictions of the class safer. Our approach's performance decreases on Disagree (compared to Disagree Strongly) since it is not good at predicting Disagree Somewhat instances which are part of the class.

5.4 Error Analysis

To gain additional insights into our approach, we analyze its six major sources of error below.

Stances not presented in a straightforward manner. As an example, consider "To my opinion this technological progress triggers off the imagination in a certain way." To identify this sentence as strongly disagreeing with the proposition "there is no longer a place for dreaming and imagination", we need to understand (1) the world knowledge that technological progress is occurring, (2) that "triggers off the imagination in a certain way" means that the technological progress coincides with imagination occurring, (3) that if imagination is occurring, there must be "a place for dreaming and imagination", and (4) that the prompt part is negated. In general, in order to construct reliable features to increase our coverage of essays that express their stance like this, we would need additional world knowledge and a deeper understanding of the text.

Rhetorical statements occasionally misidentified as stancetaking. For example, our method

[10] All significance tests are approximate randomization tests with $p < 0.01$. Boldfaced results are significant w.r.t. micro F-score for the Always Agree Strongly baseline, and macro F-score w.r.t. the Duplicated Faulkner baseline.

	System	Micro-F	Macro-F	A	Neu	D	Nev
1	Always Agree Strongly	64.8	19.7	78.7	.0	.0	.0
2	N-Gram	64.3	19.7	78.2	.0	.8	.0
3	Duplicated Faulkner	62.3	25.1	75.1	.0	25.2	.0
4	N-Gram + Duplicated Faulkner	62.6	23.7	75.8	.0	19.0	.0
5	Our approach	**67.6**	**29.1**	78.5	.0	38.0	.0

Table 4: Cross-validation results for four-class essay stance classification for Agree (A), Neutral (Neu), Disagree (D), and Never Addressed (Nev).

for identifying stancetaking paths misidentifies "I am going to discuss the topic that television is the opium of the masses in modern society" as stancetaking. To handle this, we need to incorporate more sophisticated methods for detecting rhetorical statements than those we are using (e.g., ignoring sentences ending in question marks).

Negation expressed without negation words. Our techniques for capturing negation are unable to detect when negation is expressed without the use of simple negation words. For example, "In this sense money is the root of life" should strongly disagree with "money is the root of all evil". The author replaced "life" with "evil", and detecting that this constitutes something like negation would require semantic knowledge about words that are somehow opposite of each other.

Insufficient feature/heuristic coverage of the Disagree Strongly class. Our stancetaking path-based features that we identified as intuitively having a connection to the Disagree Strongly class together cover only 51% of Disagree Strongly instances, meaning that it is in principle impossible for our system to identify the remaining 49%. However, our decision to incorporate only features that are expected to have fairly high precision for some class was intentional, as the lesson we learned from the Faulkner-based system is that it is difficult to learn a good classifier for stance classification using a large number of weakly or non-predictive features. To solve this problem, we would therefore need to exploit other aspects of strongly disagreeing essays that act as reliable predictors of the class.

Rarity of minority class instances. It is perhaps not surprising that our learning-based approach performs poorly on the minority classes. Even though the knowledge-based features were designed in part to improve the prediction of minority classes, our results suggest that the resulting features were not effectively exploited by the learners. To address this problem, one could employ a hybrid rule-based and learning-based approach where we use our machine-learned classifier to classify an instance only if it cannot be classified by any of these rules.

Lack of obvious similarity between instances of the same class. For example, if the most straightforward stancetaking sentence in an Agree Somewhat instance reads something like this, "In conclusion, I will not go to such extremes as to declare nihilistically that university does not prepare me for the real world in the least", (given the prompt part "Most university degrees do not prepare us for real life"), and we somehow managed to identify the instance's class as Agree Somewhat, what would the instance have in common with other Agree Somewhat instances? Given the numerous ways of expressing a stance, we believe a deeper understanding of essay text is required in order automatically detect how instances like this are similar to instances of the same class, and such similarities are required for learning in general.

6 Conclusion

We examined the new task of fine-grained essay stance classification, in which we determine stance for each prompt part and allow stance to take one of six values. We addressed this task by proposing two novel types of features, stancetaking path-based features and knowledge-based features. In an evaluation on 826 argumentative essays, our learning-based approach, which combines our novel features with n-gram features and Faulkner's features, significantly outperformed four baselines, including our re-implementation of Faulkner's system. Compared to the best baselines, our approach yielded relative error reductions of 11.3% and 5.3%, in micro and macro F-score, respectively. Nevertheless, accurately predicting the Somewhat, Neutral, and Never Addressed stances remains a challenging task. To stimulate further research on this task, we make all of our stance annotations publicly available.

Acknowledgments

We thank the three anonymous reviewers for their detailed comments. This work was supported in part by NSF Grants IIS-1219142 and IIS-1528037. Any opinions, findings, conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of NSF.

References

Amjad Abu-Jbara, Ben King, Mona Diab, and Dragomir Radev. 2013. Identifying opinion subgroups in arabic online discussions. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 829–835.

Rakesh Agrawal, Sridhar Rajagopalan, Ramakrishnan Srikant, and Yirong Xu. 2003. Mining newsgroups using networks arising from social behavior. In *Proceedings of the 12th International Conference on World Wide Web*, pages 529–535.

Yigal Attali and Jill Burstein. 2006. Automated essay scoring with E-rater v.2.0. *Journal of Technology, Learning, and Assessment*, 4(3).

Alexandra Balahur, Zornitsa Kozareva, and Andrés Montoyo. 2009. Determining the polarity and source of opinions expressed in political debates. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 468–480.

Mohit Bansal, Claire Cardie, and Lillian Lee. 2008. The power of negative thinking: Exploiting label disagreement in the min-cut classification framework. In *Proceedings of the 22nd International Conference on Computational Linguistics: Companion volume: Posters*, pages 15–18.

Or Biran and Owen Rambow. 2011. Identifying justifications in written dialogs. In *Proceedings of the 2011 IEEE Fifth International Conference on Semantic Computing*, pages 162–168.

Filip Boltužić and Jan Šnajder. 2014. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58.

Clinton Burfoot, Steven Bird, and Timothy Baldwin. 2011. Collective classification of congressional floor-debate transcripts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1506–1515.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37.

Dipanjan Das, Nathan Schneider, Desai Chen, and Noah A. Smith. 2010. Probabilistic frame-semantic parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 948–956.

Adam Faulkner. 2014. Automated classification of stance in student essays: An approach using stance target information and the Wikipedia link-based measure. In *Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference*.

Sylviane Granger, Estelle Dagneaux, Fanny Meunier, and Magali Paquot. 2009. *International Corpus of Learner English (Version 2)*. Presses universitaires de Louvain.

Kazi Saidul Hasan and Vincent Ng. 2013. Stance classification of ideological debates: Data, models, features, and constraints. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1348–1356.

Ken Hyland. 2005. *Metadiscourse: Exploring interaction in writing*. Continuum Discourse. Continuum, London.

Andrew Kachites McCallum. 2002. MALLET: A Machine Learning for Language Toolkit. `http://mallet.cs.umass.edu`.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.

Akiko Murakami and Rudy Raymond. 2010. Support or oppose? classifying positions in online debates from reply activities and opinion expressions. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 869–875.

Isaac Persing and Vincent Ng. 2015. Modeling argument strength in student essays. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 543–552.

Parinaz Sobhani, Diana Inkpen, and Stan Matwin. 2015. From argumentation mining to stance classification. In *Proceedings of the 2nd Workshop on Argumentation Mining*, pages 67–77.

Swapna Somasundaran and Janyce Wiebe. 2010. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, pages 116–124.

Dhanya Sridhar, James Foulds, Bert Huang, Lise Getoor, and Marilyn Walker. 2015. Joint models of disagreement and stance in online debate. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 116–125.

Christian Stab and Iryna Gurevych. 2014. Identifying argumentative discourse structures in persuasive essays. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 46–56.

Matt Thomas, Bo Pang, and Lillian Lee. 2006. Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 327–335.

Marilyn Walker, Pranav Anand, Rob Abbott, and Ricky Grant. 2012. Stance classification using dialogic properties of persuasion. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 592–596.

Yi-Chia Wang and Carolyn P. Rosé. 2010. Making conversational structure explicit: Identification of initiation-response pairs within online discussions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 673–676.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39(2–3):165–210.

Ainur Yessenalina, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1046–1056.

A New Psychometric-inspired Evaluation Metric for Chinese Word Segmentation

Peng Qian Xipeng Qiu* Xuanjing Huang
Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
School of Computer Science, Fudan University
825 Zhangheng Road, Shanghai, China
{pqian11, xpqiu, xjhuang}@fudan.edu.cn

Abstract

Word segmentation is a fundamental task for Chinese language processing. However, with the successive improvements, the standard metric is becoming hard to distinguish state-of-the-art word segmentation systems. In this paper, we propose a new psychometric-inspired evaluation metric for Chinese word segmentation, which addresses to balance the very skewed word distribution at different levels of difficulty[1]. The performance on a real evaluation shows that the proposed metric gives more reasonable and distinguishable scores and correlates well with human judgement. In addition, the proposed metric can be easily extended to evaluate other sequence labelling based NLP tasks.

1 Introduction

Word segmentation is a fundamental task for Chinese language processing. In recent years, Chinese word segmentation (CWS) has undergone great development, which is, to some degree, driven by evaluation conferences of CWS, such as SIGHAN Bakeoffs (Emerson, 2005; Levow, 2006; Jin and Chen, 2008; Zhao and Liu, 2010). The current state-of-the-art methods regard word segmentation as a sequence labeling problem (Xue, 2003; Peng et al., 2004). The goal of sequence labeling is to assign labels to all elements in a sequence, which can be handled with supervised learning algorithms, such as maximum entropy (ME) (Berger et al., 1996), conditional random fields (CRF) (Lafferty et al., 2001) and Perceptron (Collins, 2002).

Benefiting from the public datasets and feature engineering, Chinese word segmentation achieves quite high precision after years of intensive research. To evaluate a word segmenter, the standard metric consists of **precision** p, **recall** r, and an evenly-weighted **F-score** f_1.

However, with the successive improvement of performance, state-of-the-art segmenters are hard to be distinguished under the standard metric. Therefore, researchers also report results with some other measures, such as out-of-vocabulary (OOV) recall, to show their strengths besides p, r and f_1.

Furthermore, although state-of-the-art methods have achieved high performances on p, r and f_1, there exists inconsistence between the evaluation ranking and the intuitive feelings towards the segmentation results of these methods. The inconsistence is caused by two reasons:

(1) The high performance is due to the fact that the distribution of difficulties of words is unbalanced. The proportion of trivial cases is very high, such as '的 ('s)', '我们 (we)', which results in that the non-trivial cases are relatively despised. Therefore, a good measure should have a capability to balance the skewed distribution by weighting the test cases.

(2) Human judgement depends on difficulties of segmentations. A segmenter can earn extra credits when correctly segmenting a difficult word than an easy word. Conversely, a segmenter can take extra penalties when wrongly segmenting an easy word than a difficult word.

Taking a sentence and two predicted segmentations as an example:
S : 白藜芦醇 是 一 种 酚类 物质
(Trans: Resveratrol is a kind of phenols material.)
P1: 白 藜芦 醇 是 一种 酚类 物质
P2: 白藜 芦醇 是 一 种 酚类物 质

We can see that the two segmentations have the

*Corresponding author.

[1]We release the word difficulty of the popular word segmentation datasets at http://nlp.fudan.edu.cn/data/ .

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2185–2194,
Berlin, Germany, August 7-12, 2016.

same scores in p, r and f_1. But intuitively, P1 should be better than P2, since P2 is worse even on the trivial cases, such as '酚类 (phenols)' and '物质 (material)'.

Therefore, we think that an appropriate evaluation metric should not only provide an all-around quantitative analysis of system performances, but also explicitly reveal the strengths and potential weaknesses of a model.

Inspired by psychometrics, we propose a new evaluation metric for Chinese word segmentation in this paper. Given a labeled dataset, not all words have the same contribution to judge the performance of a segmenter. Based on psychometric research (Lord et al., 1968), we assign a difficulty value to each word. The difficulty of a word is automatically rated by a committee of segmenters, which are diversified by training on different datasets and features. We design a balanced precision, recall to pay different attentions to words according to their difficulties.

We also give detailed analysis on a real evaluation of Chinese word segmentation with our proposed metric. The analysis result shows that the new metric gives a more balanced evaluation result towards the human intuition of the segmentation quality. We will release the weighted datasets focused this paper to the academic community.

Although our proposed metric is applied to Chinese word segmentation for a case study, it can be easily extended to other sequence labelling based NLP tasks.

2 Standard Evaluation Metric

The standard evaluation usually uses three measures: precision, recall and balanced F-score.

Precision p is defined as the number of correctly segmented words divided by the total number of words in the automatically segmented corpus.

Recall r is defined as the number of correctly segmented words divided by the total number of words in the gold standard, which is the manually annotated corpus.

F-score f_1 the harmonic mean of precision and recall.

Given a sentence, the gold-standard segmentation of a sentence is $w_1, \cdots, W_N$, N is the number of words. The predicted segmentation is $w'_1, \cdots, w'_{N'}$, N' is the number of words. Among that, the number of words correctly identified by the predicted segmentation is c, and the number of incorrectly predicted words is e.

p, r and f_1 are defined as follows:

$$p = \frac{c}{N'}, \tag{1}$$

$$r = \frac{c}{N}, \tag{2}$$

$$f_1 = \frac{2 \times p \times r}{p + r}. \tag{3}$$

As a complement to these metrics, researchers also use the recall of out-of-vocabulary (OOV) words to measure the segmenter's performance in detecting unknown words.

3 A New Psychometric-inspired Evaluation Metric

We involve the basic idea from psychometrics and improve the evaluation metric by assigning weights to test cases.

3.1 Background Theory

This work is inspired by the test theory in psychometrics (Lord et al., 1968). Psychologists, as well as educators, have studied the way of analyzing items in a psychological test, such as IQ test. The general idea is that test cases should be given different weights, which reflects the effectiveness of a certain item to a certain measuring object.

Similarly, we consider an evaluation task as a kind of special psychological test. The psychological traits, or the ability of the model, is not an explicit characteristics. We propose that the test cases for NLP task should also be assigned a real value to account for the credits that the tagger earned from answering the test case.

In analogy to the way of computing difficulty in psychometrics, the difficulty of a target word w_i is defined as the error rate of a committee in the case of word segmentation.

Given a committee of K base segmenters, we can get K segmentations for sentence $w_1, \cdots, W_N$. We use a mark $m_i^k \in \{0, 1\}$ to indicate whether word w_i is correctly segmented by the k-th segmenter.

The number of words c^k correctly identified by the k-th segmenter is

$$c^k = \sum_{i=1}^{N} m_i^k. \tag{4}$$

Thus, we can calculate the degree of difficulty of each word w_i.

$$d_i = \frac{1}{K}\sum_{k=1}^{K}(1 - m_i^k). \quad (5)$$

This methodology of measuring test item difficulty is also widely applied in assessing standardized exams such as TOEFL (Service, 2000).

3.2 Psychometric-Inspired Evaluation Metric

Since the distribution of the difficulties of words is very skew, we design a new metric to balance the weights of different words according to their difficulties. In addition, we also should keep strictly a fair rule for rewards and punishments.

Intuitively, if the difficulty of a word is high, a correct segmentation should be given an extra rewards; otherwise, if the difficulty of a word is low, it is reasonable to give an extra punishment to a wrong segmentation.

Our new metric of precision, recall and balanced F-score is designed as follows.

Balanced Recall Given a new predicted segmentation, the mark $m_i \in \{0, 1\}$ indicates whether word w_i is correctly segmented. d_i is the degree of difficulty of word w_i.

According to the difficulties of each word, we can calculated the reward recall r_{reward} which biased for the difficult cases.

$$r_{reward} = \frac{\sum_{i=1}^{N} d_i \times m_i}{\sum_{i=1}^{N} d_i}, \quad (6)$$

where $r'_{reward} \in [0, 1]$ is biased recall, which places more attention on the difficult cases and less attention on the easy cases.

Conversely, we can calculated another punishment recall $r_{punishment}$ which biased for the easy cases.

$$r_{punishment} = \frac{\sum_{i=1}^{N} (1 - d_i) \times m_i}{\sum_{i=1}^{N} (1 - d_i)}, \quad (7)$$

where $r_{punishment} \in [0, 1]$ is biased recall, which places more attention on the easy cases and less attention on the difficult cases.

$r_{punishment}$ can be interpreted as a punishment as bellows.

$$r_{punishment} = \frac{\sum_{i=1}^{N} (1 - d_i) \times m_i}{\sum_{i=1}^{N} (1 - d_i)}, \quad (8)$$

$$= 1 - \frac{\sum_{i=1}^{N} (1 - d_i) \times (1 - m_i)}{\sum_{i=1}^{N} (1 - d_i)}. \quad (9)$$

From Eq (9), we can see that an extra punishment is given to wrong segmentation for low difficult word. In detailed, for a word w_i that is easy to segment, its weights $(1 - d_i)$ is relative higher. When its segmentation is wrong, $m_i = 0$. Therefore, $(1 - d_i) \times (1 - m_i) = (1 - d_i)$ will be larger, which results to a smaller final score.

To balance the reward and punishment, a balanced recall r_b is used, which is the harmonic mean of r_{reward} and $r_{punishment}$.

$$r_b = \frac{2 \times r_{punishment} \times r_{reward}}{r_{punishment} + r_{reward}} \quad (10)$$

Balanced Precision Given a new predicted segmentation, the mark $m'_i \in \{0, 1\}$ to indicate whether segment s'_i is correctly segmented. d'_i is the degree of difficulty of segment s'_i, which is an average difficulty of the corresponding gold words.

Similar to balanced recall, we use the same way to calculate balance precision p_b. Here N' is the number of words in the predicted segmentation. d'_i is the weight for the predicted segmentation unit w'_i. It equals to the word difficulty of the corresponding word w that cover the right boundary of w'_i in the gold segmentation.

$$p_{reward} = \frac{\sum_{i=1}^{N} (1 - d_i) \times m_i}{\sum_{i=1}^{N'} (1 - d'_i)}, \quad (11)$$

$$p_{punishment} = \frac{\sum_{i=1}^{N} (1 - d_i) \times m_i}{\sum_{i=1}^{N'} (1 - d'_i)}, \quad (12)$$

$$p_b = \frac{2 \times p_{reward} \times p_{punishment}}{p_{reward} + p_{punishment}}. \quad (13)$$

$$(14)$$

Balanced F-score The final balanced F-score is

$$f_b = \frac{2 \times p_{balanced} \times r_{balanced}}{p_{balanced} + r_{balanced}}. \quad (15)$$

4 Committee of Segmenters

It is infeasible to manually judge the difficulty of each word in a dataset. Therefore, an empirical method is needed to rate each word. Since the difficulty is also not derivable from the observation of the surface forms of the text, we use a committee of automatic segmenters instead. To keep fairness

F1	$C_iT_0, (i = -1, 0, 1)$
	$C_{i:i+1}T_0, (i = -1, 0)$
	$T_{-1,0}$
F2	$C_iT_0, (i = -2, -1, 0, 1, 2)$
	$C_{i:i+1}T_0, (i = -2, -1, 0, 1)$
	$T_{-1,0}$
F3	$C_iT_0, (i = -2, -1, 0, 1, 2)$
	$C_{i:i+1}T_0, (i = -2, -1, 0, 1)$
	$C_{i:i+2}T_0, (i = -2, -1, 0)$
	$T_{-1,0}$

Table 1: Feature templates. C represents a Chinese character, and T represents the character-based tag in set {B, M, E, S}. The subscript indicates its position relative to the current character, whose subscript is 0. $C_{i:j}$ represents the subsequence of characters form relative position i to j.

and justice of the committee, we need a large number of diversified committee members.

Thus, the grading result of committee is fair and accurate, avoiding the laborious human annotation and the deviation caused by the subjective factor of the artificial judgement.

4.1 Building the Committee

Base Segmenters The committee is composed of a series of base segmenters, which are based on discriminative character-based sequence labeling method. Each character is labeled as one of {B, M, E, S} to indicate the segmentation. 'B' indicates the beginning character of a word. 'M' indicates the middle character of a word. 'E' indicates the end character of a word. 'S' indicates that the word consists of only a single character.

Diversity of Committee To objectively assess the difficulty of a word, we need to maintain a large enough committee with diversity.

To encourage diversity among committee members, we train them with different datasets and features. Specifically, each base segmenter adopts one of three types of feature templates (shown in Table 1), and are trained on randomly sampling training sets. To keep a large diversity, we set sampling ratio to be 10%, 20% and 30%. In short, each base segmenter is constructed with a random combination of the candidate feature template and the sampling ratio for training dataset.

Size of Committee To obtain a valid and reliable assessment for a word, we need to choose the

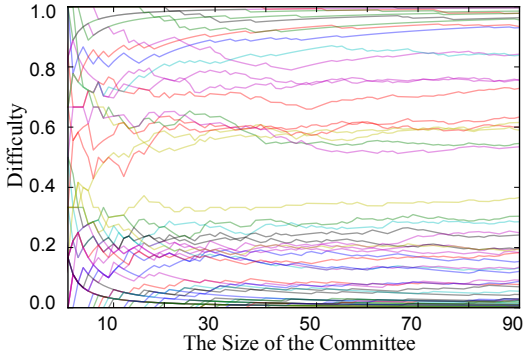


Figure 1: Judgement of difficulty against the committee size. Each line represents a sampled word.

appropriate size of committee. For a given test case, the judgement of its difficulty should be relatively stable. We analyze how the judgement of its difficulty changes as the size of committee increases.

Figure 2 show PKU data from SIGHAN 2005 (Emerson, 2005) the difficulty is stable when the sample size is large enough.

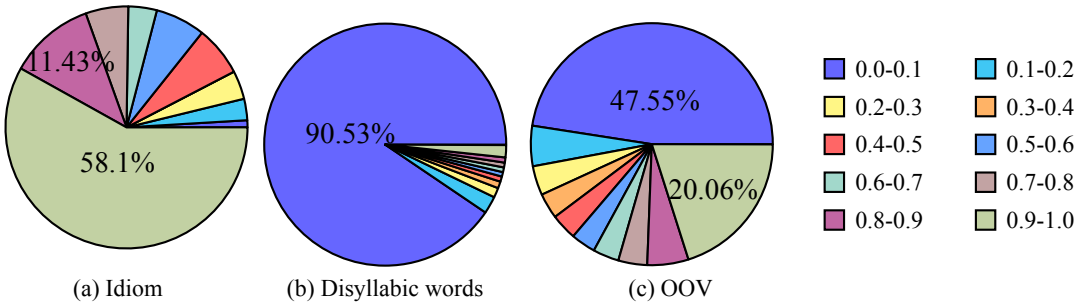
4.2 Interpreting Difficulty with Linguistic Features

Since we get the difficulty for each word empirically, we naturally want to know whether the difficulty is explainable, as what TOEFL researchers have done (Freedle and Kostin, 1993; Kostin, 2004). We would like to know whether the variation of word difficulty can be partially expalined by a series of traceable linguistic features.

Based on the knowledge about the characteristics of Chinese grammar and the practical experiences of corpus annotation, we consider the following surface linguistic features. In order to explicitly display the relationship between the linguistic predictors and the distribution of the word difficulty at a micro level, we divide the difficulty scale into ten discrete intervals and calculate the distributions of these linguistic features on different ranges of difficulty.

Here, we interpret the difficulties of the words from the perspective of three important linguistic features:

Idiom In Chinese, the 4-character idioms have special linguistic structures. These structure usually form a different pattern that is hard for the machine algorithm to understand. Therefore,



(a) Idiom (b) Disyllabic words (c) OOV

Figure 2: Difficulty distribution of (a) idioms, (b) dysyllabic words and (c) Out-of-vocabulary words from PKU dataset. Similar pattern has also been found in other datasets.

it is reasonable to hypothesize that the an idiom phrase is more likely to be a difficult word for word segmentation task. We can see from Figure 2a that 58.1% of idioms have a difficulty at (0.9,1]. The proportion does increase with the degree of difficulty, which corresponds with the human intuition.

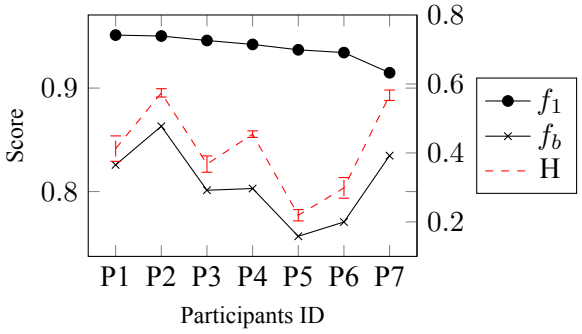
Dissyllabic Word Disyllabic word is a word formed by two consecutive Chinese characters. We can see from Figure 2b that the frequency of disyllabic words has a negative correlations with the degree of difficulty. This is an interesting result. It means that a two-syllable word pattern is easy for a machine algorithm to recognize. This is consistent with the lexical statistics (Yip, 2000), which shows that dissyllabic words account for 64% of the common words in Chinese.

Out-of-vocabulary Word Processing out-of-vocabulary (OOV) word is regarded as one of the key factors to the improvement of model performance. Since these words never occur in the training dataset, it is for sure that the word segmentation system will find it hard to correctly recognize these words from the contexts. We can see from Figure 2c that OOV generally has high difficulty. However, a lot of OOV is relatively easy for segmenters.

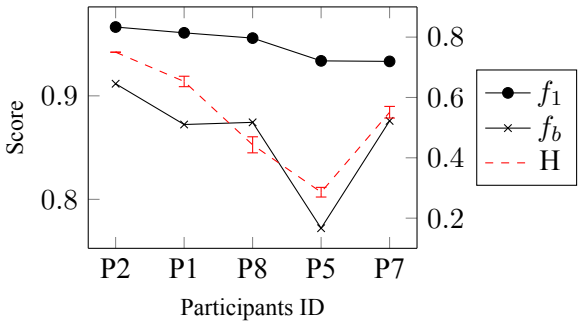
All the linguistic predictors above prove that the degree of difficulty, namely the weight for each word, is not only rooted in the foundation of test theory, but also correlated with linguistic intuition.

5 Evaluation with New Metric

Here we demonstrate the effectiveness of the proposed method in a real evaluation by re-analyzing the submission results from NLPCC



(a) Closed Track



(b) Open Track

Figure 3: Comparisons of standard metric and our new metric for the closed track and the open track of NLPCC 2015 Weibo Text Word Segmentation Shared Task. The black lines for f_1 and f_b are plotted against the left y-axis. The red lines for human judgement scores are plotted against the right y-axis.

2015 Shared Task[2] of Chinese word segmentation. The dataset of this shared task is collected from micro-blog text. For convenience, we use WB to represent this dataset in the following discussions.

We select the submissions of all 7 participants from the closed track and the submissions of all

[2]Conference on Natural Language Processing and Chinese Computing. http://tcci.ccf.org.cn/conference/2015/

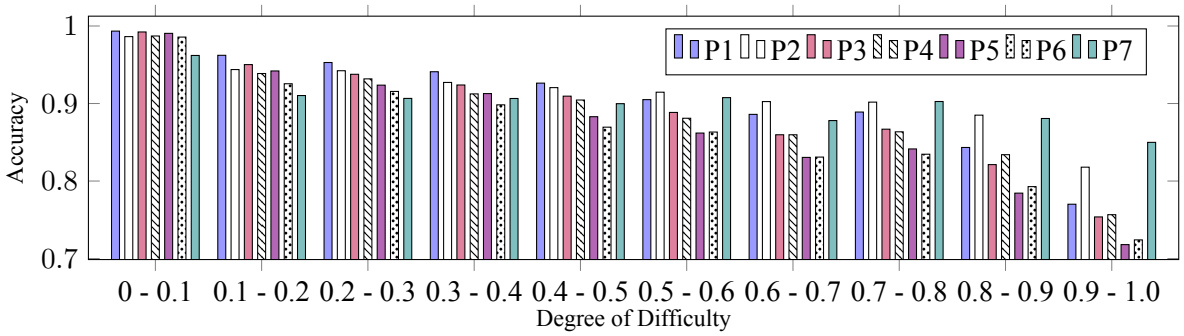


Figure 4: Accuracies of different participants in Closed Track by different difficulties on WB dataset.

5 participants from the open track. In the closed track, participants could only use information found in the provided training data. In the open track, participants could use the information which should be public and be easily obtained.

We compare the standard precision, recall and F-score with our new metric. The result is displayed in Figure 3. Considering the related privacy issues, we will refer to the participants as P1, P2, etc. The order of these participants in the sub-figures is sorted according to the original ranking given by the standard metric in each track. The same ID number refers to the same participants.

It is interesting to see that the proposed metric gives out significantly different rankings for the participants, compared to the original rankings. Based on the standard metric, Participant 1 (P1) ranks the top in closed track while P7 is ranked as the worst in both tracks. However, P2 ranks first under the evaluation of the new metric in the Closed track. P7 also get higher ranking than its original one.

5.1 Correlation with Human Judgement

To tell whether the standard metric or the proposed metric is more reasonable, we asked three experts to evaluate the quality of the submissions from the participants. We randomly selected 50 test sentences from the WB dataset. For each test sentence, we present all the submitted candidate segmentation results to the human judges in random order. Then, the judges are asked to choose the best candidate(s) with the highest segmentation quality as well as the second-best candidate(s) among all the submissions. Human judges had no access to the source of the sentences.

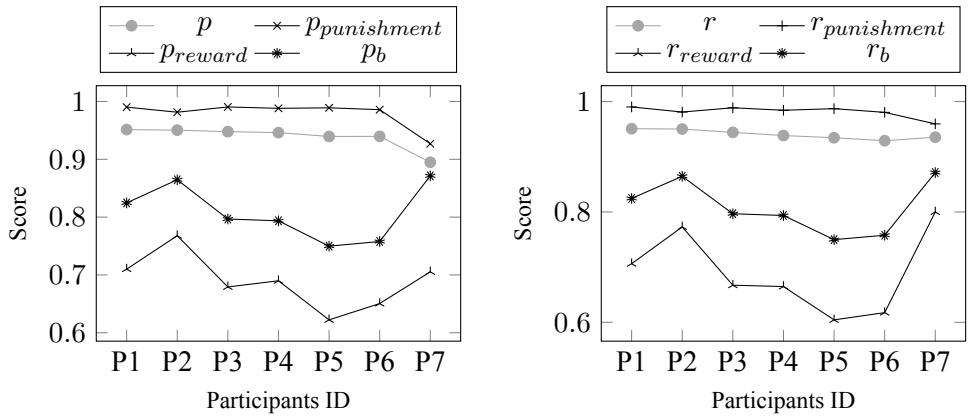
Once we collect the human judgement of the segmentation quality, we can compute the score for each participants. If a candidate segmentation result from a certain participant is ranked first for n times, then this participants earned n point. If second for m times, then this participants earned $\frac{m}{2}$ points. Then we can get the probability of a participants being ranked the best or sub-best by computing $\frac{n+\frac{m}{2}}{50}$. Finally, we get the human-intuition-based gold ranking of the participants through the means of scores from all the human judges.

It is worth noticing that the ranking result of our proposed metric correlates with the human judgements better than that of the standard metric, as is shown in Figure 3. The Pearson correlation between our proposed metric and human judgements are 0.9056 ($p = 0.004$) for closed session and 0.8799 ($p = 0.04$) for open session while the Pearson correlation between standard metric and human judgements are only 0.096 ($p = 0.836$) for closed session and 0.670 ($p = 0.216$). This evidence strongly supports that the proposed method is a good approximate of human judgements.

5.2 Detailed Analysis

Since we have empirically got the degree of difficulty for each word in the test dataset, we can compute the distribution of the difficulty for words that have been correctly segmented. We divided the whole range of difficulty into 10 intervals. Then, we count the ratio of the correct segmented units for each difficulty interval. In this way, we can quantitatively measure to what extent the segmentation system performs on difficult test cases and easy test cases.

As is shown in Figure 4, P7 works better on difficult cases than other systems, but the worst on easy cases. This explains why P7 gets good rank based on the new evaluation metric. Besides,



(a) Standard and Weighted Precision

(b) Standard and Weighted Recall

Figure 5: Comparisons of standard and weighted precision and recall on NLPCC Closed Track.

if we compare P1 and P2, we will notice that P2 performs just slightly worse than P1 on easy cases, but much better than P1 on difficult cases. Therefore, conventional evaluation metric rank P1 as the top system because the P1 gets a lot of credits from a large portion of easy cases. Unlike conventional metric, our new metric achieves balance between hard cases and easy cases and ranks P2 as the top system.

The experiment result indicates that the new metric can reveal the implicit difference and improvement of the model, while standard metric cannot provide us with such a fine-grained result.

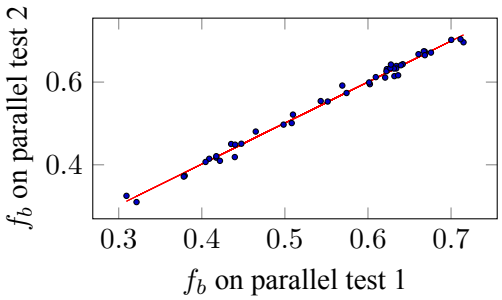


Figure 6: Correlation between the evaluation results f_b of two parallel testsets with the proposed metrics on a collection of models. The Pearson correlation is 0.9961, $p = 0.000$.

5.3 Validity and Reliability

Jones (1994) concluded some important criteria for the evaluation metrics of NLP system. It is very important to check the validity and reliability of a new metric.

Previous section has displayed the validity of the proposed evaluation metric by comparing the evaluation results with human judgements. The evaluation results with our new metric correlated with human intuition well.

Regarding reliability, we perform the parallel-test experiment. We randomly split the test dataset into two halves. These two halves have similar difficulty distribution and, therefore, can be considered as a parallel test. Then different models, including those used in the first experiment, are evaluated on the first half and the second half. The results in Figure 6 shows that the performances of different models with our proposed evaluation metric are significantly correlated in two parallel tests.

5.4 Visualization of the Weight

As is known, there might be some annotation inconsistency in the dataset. We find that most of the cases with high weight are really valuable difficult test cases, such as the visualized sentences from WB dataset in Figure 7. In the first sentence, the word ‘BMW 族’ (NOUN.People who take bus, metro and then walk to the destination) is an OOV word and contains English characters. The weight of this word, as expected, is very high. In the second sentence, the word ‘素不相识’ (VERB.not familiar with each other) is a 4-character Chinese idiom. the conjunction word ‘就算’ (CONJ.even if) has structural ambiguity. It can also be decomposed into a two-word phrase ‘就’ (ADV.just) and ‘算’ (VERB.count). From the visualization of the weight, we can see that these difficult words are all given high weights.

Data	Corpus Size	p	r	$f1$	p_b	r_b	f_b
PKU	20%	90.04	89.90	89.97	45.22	43.37	44.28
	50%	92.87	91.58	92.22	54.24	49.12	51.55
	80%	94.07	92.21	93.13	61.80	54.74	58.05
	100%	94.03	92.91	93.47	64.22	59.16	61.59
MSR	20%	92.93	92.58	92.76	45.76	44.13	44.93
	50%	95.22	95.18	95.20	63.00	62.22	62.60
	80%	95.68	95.74	95.71	67.26	66.96	67.11
	100%	96.19	96.02	96.11	70.80	69.45	70.12
NCC	20%	87.32	86.37	86.84	42.16	40.23	41.17
	50%	89.34	89.03	89.19	50.31	49.26	49.78
	80%	91.42	91.10	91.26	60.48	59.25	59.86
	100%	92.00	91.77	91.89	63.72	62.70	63.20
SXU	20%	89.70	89.31	89.50	43.53	42.35	42.93
	50%	93.04	92.42	92.73	56.21	54.27	55.23
	80%	94.45	93.94	94.19	64.55	62.50	63.51
	100%	94.89	94.61	94.75	68.10	66.63	67.36

Table 2: Model evaluation with standard metric and our new metric. Models vary in the amount of training data and feature types.

6 Comparisons on SIGHAN datasets

In this section, we give comparisons on SIGHAN datasets. We use four simplified Chinese datasets: PKU and MSR (SIGHAN 2005) as well as NCC and SXU (SIGHAN 2008).

For each dataset, we train four segmenters with varying abilities, based on 20%, 50%, 80% and 100% of training data respectively. The used feature template is F2 in Table 1.

Table 2 shows the different evaluation results with standard metric and our balanced metric. We can see that the proposed evaluation metric generally gives lower and more distinguishable score, compared with the standard metric.

7 Related work

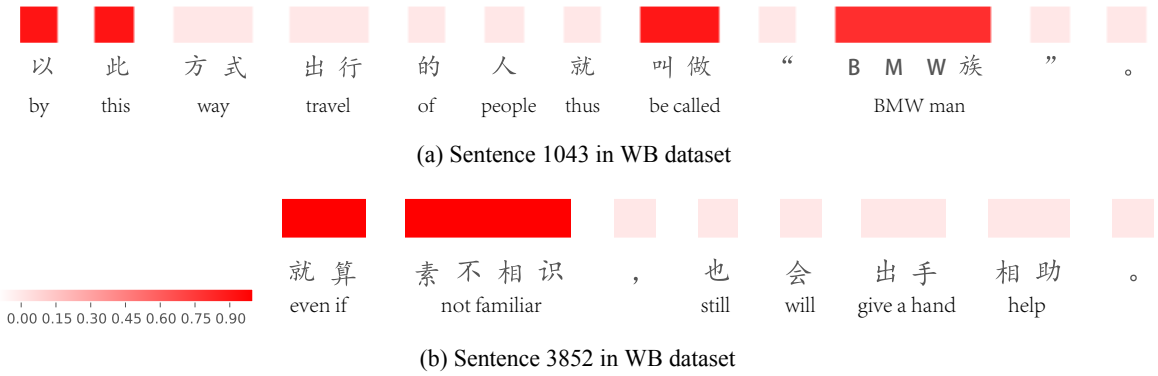
Evaluation metrics has been a focused topic for a long time. Researchers have been trying to evaluate various NLP tasks towards human intuition (Papineni et al., 2002; Graham, 2015a; Graham, 2015b). Previous work (Fournier and Inkpen, 2012; Fournier, 2013; Pevzner and Hearst, 2002) mainly deal with the near-miss error case on the context of text segmentation. Much attention has been given to different penalization for the error. These work criticize that traditional metrics such as precision, recall and F-score, consider all the error similar. In this sense, some studies aimed at assigning different penalization to the word. We think that these explorations can be regarded as the foreshadowing of our evaluation metric that balances reward and punishment.

Our paper differs from previous research in that we take the difficulty of the test case into consideration, while previous works only focus on the variation of error types and penalisation. We involve the basic idea from psychometrics and improve the evaluation with a balance between difficult cases and easy cases, reward and punishment.

We would like to emphasize that our weighted evaluation metric is not a replacement of the traditional precision, recall, and F-score. Instead, our new weighted metrics can reveal more details that traditional evaluation may not be able to present.

8 Conclusion

In this paper, we put forward a new psychometric-inspired method for Chinese word segmentation evaluation by weighting all the words in test dataset based on the methodology applied to psychological tests and standardized exams. We empirically analyze the validity and reliability of the new metric on a real evaluation dataset. Experiment results reveal that our weighted evaluation metrics gives more reasonable and distinguishable scores and



(a) Sentence 1043 in WB dataset

(b) Sentence 3852 in WB dataset

Figure 7: Visualising the word weight of WB dataset.

correlates well with human judgement. We will release the weighted datasets to the academic community.

Additionally, the proposed evaluation metric can be easily extended to word segmentation task for other languages (e.g. Japanese) and other sequence labelling-based NLP tasks, with just tiny changes. Our metric also points out a promising direction for the researchers to take into the account of the biased distribution of test case difficulty and focus on tackling the hard bones of natural language processing.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

A.L. Berger, V.J. Della Pietra, and S.A. Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

T. Emerson. 2005. The second international Chinese word segmentation bakeoff. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing*, pages 123–133. Jeju Island, Korea.

Chris Fournier and Diana Inkpen. 2012. Segmentation similarity and agreement. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 152–161. Association for Computational Linguistics.

Chris Fournier. 2013. Evaluating text segmentation using boundary edit distance. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 1702–1712.

Roy Freedle and Irene Kostin. 1993. The prediction of toefl reading item difficulty: Implications for construct validity. *Language Testing*, 10(2):133–170.

Yvette Graham. 2015a. Improving evaluation of machine translation quality estimation. In *Proceedings of the 53th Annual Meeting on Association for Computational Linguistics*.

Yvette Graham. 2015b. Re-evaluating automatic summarization with bleu and 192 shades of rouge. In *Proceedings of EMNLP*.

C. Jin and X. Chen. 2008. The fourth international Chinese language processing bakeoff: Chinese word segmentation, named entity recognition and Chinese pos tagging. In *Sixth SIGHAN Workshop on Chinese Language Processing*, page 69.

Karen Sparck Jones. 1994. Towards better nlp system evaluation. In *Proceedings of the workshop on Human Language Technology*, pages 102–107. Association for Computational Linguistics.

Irene Kostin. 2004. Exploring item characteristics that are related to the difficulty of toefl dialogue items. *ETS Research Report Series*, 2004(1):i–59.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*.

Gina-Anne Levow. 2006. The third international chinese language processing bakeoff: Word segmentation and named entity recognition. In *Proceedings of the Fifth SIGHAN Workshop on Chinese Language Processing*, pages 108–117, Sydney, Australia, July.

Frederic M Lord, Melvin R Novick, and Allan Birnbaum. 1968. *Statistical Theories of Mental Test Scores.* Addison-Wesley.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

F. Peng, F. Feng, and A. McCallum. 2004. Chinese segmentation and new word detection using conditional random fields. *Proceedings of the 20th international conference on Computational Linguistics*.

Lev Pevzner and Marti A Hearst. 2002. A critique and improvement of an evaluation metric for text segmentation. *Computational Linguistics*, 28(1):19–36.

Educational Testing Service. 2000. *Computer-Based TOEFL Score User Guide*. Princeton, NJ.

N. Xue. 2003. Chinese word segmentation as character tagging. *Computational Linguistics and Chinese Language Processing*, 8(1):29–48.

Po-Ching Yip. 2000. *The Chinese Lexicon: A Comprehensive Survey*. Psychology Press.

H. Zhao and Q. Liu. 2010. The cips-sighan clp 2010 chinese word segmentation bakeoff. In *Proceedings of the First CPS-SIGHAN Joint Conference on Chinese Language Processing*.

Temporal Anchoring of Events for the TimeBank Corpus

Nils Reimers†‡, **Nazanin Dehghani**†‡ *, **Iryna Gurevych**†‡
†Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
‡ Research Training Group AIPHES
Technische Universität Darmstadt
http://www.ukp.tu-darmstadt.de

Abstract

Today's extraction of temporal information for events heavily depends on annotated temporal links. These so called TLINKs capture the relation between pairs of event mentions and time expressions. One problem is that the number of possible TLINKs grows quadratic with the number of event mentions, therefore most annotation studies concentrate on links for mentions in the same or in adjacent sentences. However, as our annotation study shows, this restriction results for 58% of the event mentions in a less precise information when the event took place.

This paper proposes a new annotation scheme to anchor events in time. Not only is the annotation effort much lower as it scales linear with the number of events, it also gives a more precise anchoring when the events have happened as the complete document can be taken into account. Using this scheme, we annotated a subset of the TimeBank Corpus and compare our results to other annotation schemes. Additionally, we present some baseline experiments to automatically anchor events in time. Our annotation scheme, the automated system and the annotated corpus are publicly available.[1]

1 Introduction

In automatic text analysis, it is often important to precisely know when an event occurred. A user might be interested in retrieving news articles that discuss certain events which happened in a given time period, for example articles discussing car bombings in the 1990s. The user might not only be interested in articles from that time period, but also in more recent articles that cover events from that period. Knowing when an event happened is also essential for time aware summarization, automated timeline generation as well as automatic knowledge base creation. In many cases, time plays a crucial role for facts stored in a knowledge base, for example for the facts when a person was born or died. Also, some facts are only true for a certain time period, like being the president of a country. Event extraction can be used to automatically infer many facts for knowledge bases, however, to be useful, it is crucial that the date when the event happened can precisely be extracted.

The TimeBank Corpus (Pustejovsky et al., 2003) is a widely used corpus using the TimeML specifications (Saurí et al., 2004) for the annotations of event mentions and temporal expressions. In order to anchor events in time, the TimeBank Corpus uses the concept of temporal links (TLINKs) that were introduced by Setzer (2001). A TLINK states the temporal relation between two events or an event and a time expression. For example, an event could happen *before*, *simultaneous*, or *after* a certain expression of time. The TimeBank Corpus served as dataset for the shared tasks TempEval-1, 2 and 3 (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013).

In this paper we describe a new approach to anchor every event in time. Instead of using temporal links between events and temporal expressions, we consider the event time as an argument of the event mention. The annotators are asked to write down the date when an event happened in a normalized format for every event mention. The annotation effort is for this reason identical

* Guest researcher from the School of Electrical and Computer Engineering, University of Tehran.

[1] https://www.ukp.tu-darmstadt.de/data/timeline-generation/temporal-anchoring-of-events/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2195–2204,
Berlin, Germany, August 7-12, 2016.

to the number of event mentions, i.e. for a document with 200 event mentions, the annotators must perform 200 annotations. When annotating the event mentions, the annotators are asked to take the complete document into account. Section 3 presents our annotation scheme, and section 4 gives details about the conducted annotation study.

The number of possible TLINKs scales quadratic with the number of events and temporal expressions. Some documents of the TimeBank Corpus contain more than 200 events and temporal expressions, resulting in more than 20.000 possible TLINKs. Hand-labeling all links is extremely time-consuming and even when using transitive closures and computational support, it is not feasible to annotate all possible TLINKs for a larger set of documents. Therefore, all annotation studies limited the number of TLINKs to annotate. For example, in the original TimeBank Corpus, only links that are salient were annotated. Which TLINKs are salient is fairly vague and results in a comparably low reported inter-annotator agreement. Furthermore, around 62% of all events do not have any attached TLINK, i.e. for most of the events in the original TimeBank Corpus, no temporal statement can be made.

In contrast to the sparse annotation of TLINKs used in the TimeBank Corpus, the TimeBank-Dense Corpus (Cassidy et al., 2014) used a dense annotation and all temporal links for events and time expressions in the same sentence and in directly succeeding sentences were annotated. For a subset of 36 documents with 1729 events and 289 time expressions, they annotated 12,715 temporal links, which is around 6.3 links per event and time expression. Besides the large effort needed for a dense annotation, a major downside is the limitation that events and time expressions must be in the same or in adjacent sentences. Our annotation study showed that in 58.72% of the cases the most informative temporal expression is more than one sentence apart from the event mention. For around 25% of the events, the most informative temporal expression is even five or more sentences away. Limiting the TLINKs to pairs that are at most one sentence apart poses the risk that important TLINKs are not annotated and consequently cannot be learned by automated systems.

A further drawback of TLINKs is that it can be difficult or even impossible to encode temporal information that originates from different parts in the text. Given the sentence:

> *December 30th, 2015 - During New Year's Eve, it is traditionally very busy in the center of Brussels and people gather for the fireworks display. But the upcoming [display]$_{Event}$ was canceled today due to terror alerts.*

For a human it is simple to infer the date for the event *display*. But it is not possible to encode this knowledge using TLINKs, as the date is not explicitly mentioned in the text.

To make our annotations comparable to the dense TLINK annotation scheme of the TimeBank-Dense Corpus (Cassidy et al., 2014), we annotated the same documents and compare the results in section 5. For 385 out of 872 events (44.14%), our annotation scheme results in a more precise value on which date an event happened.

Section 6 presents a baseline system to extract event times. For a subset of events, it achieves an F_1-score of 49.01% while human agreement for these events is 80.50%.

2 Previous Annotation Work

The majority of corpora on events uses sparse temporal links (TLINKs) to enable anchoring of events in time. The original TimeBank (Pustejovsky et al., 2003) only annotated salient temporal relations. The subsequent TempEval competitions (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013) are based on the original TimeBank annotations, but tried to improve the coverage and added some further temporal links for mentions in the same sentence. The MEANtime corpus (van Erp et al., 2015) applied a sparse annotation and only temporal links between events and temporal expressions in the same and in succeeding sentences were annotated. The MEANtime corpus distinguishes between main event mentions and subordinated event mentions, and the focus for TLINKs was on main events.

More dense annotations were applied by Bramsen et al. (2006), Kolomiyets et al. (2012), Do et al. (2012) and by Cassidy et al. (2014). While Bramsen et al., Kolomiyets et al., and Do et al. only annotated some temporal links, Cassidy et al. annotated all Event-Event, Event-Time, and Time-Time pairs in the same sentence as well as in directly succeeding sentences leading to the densest annotation for the TimeBank Corpus.

A drawback of the previous annotation works is the limitation that only links between expressions in the same or in succeeding sentences are annotated. In case the important temporal expression, that defines when the event occurred, is more than one sentence away, the TLINK will not be annotated. Consequently, retrieving the information when the event occurred is not possible. Increasing this window size would result in a significantly increased annotation effort as the number of links grows quadratic with the number of expressions.

Our annotation is the first for the TimeBank Corpus that does not try to annotate the quadratic growing number of temporal links. Instead, we consider the event time as an argument of the individual event mention and it is annotated directly by the annotators. This reduces the annotation effort by 85% in comparison to the TimeBank-Dense Corpus. This allows an annotator to annotate significant more documents in the same time. Also, all temporal information, independent where it is mentioned in the document, can be taken into account resulting in a much more precise anchoring of events in time, as section 5 shows.

3 Event Time Annotation Scheme

The annotation guidelines for the TimeBank Corpus (Saurí et al., 2004) define an *event* as a cover term for situations that *happen* or *occur*. Events can be *punctual* or last for *a period of time*. They also consider as events those predicates describing *states* or *circumstances* in which something holds true. For the TimeBank Corpus, the smallest extent of text (usually a single word) that expresses the occurrence of an event is annotated.

The aspectual type of the annotated events in the TimeBank Corpus can be distinguished into *achievement events*, *accomplishment events*, and *states* (Pustejovsky, 1991). An achievement is an event that results into an instantaneous change of some sort. Examples of *achievement events* are *to find*, *to be born*, or *to die*. *Accomplishment events* also result into a change of some sort, however, the change spans over a longer time period. Examples are *to build something* or *to walk somewhere*. States on the other hand do not describe a change of some sort, but that something holds true for some time, for example, *being sick* or *to love someone*. The aspectual type of an event does not only depend on the event itself, but also on the context in which the event is expressed.

Our annotation scheme was created with the goal of being able to create a knowledge base from the extracted events in combination with their event times. Punctual events are a single dot on the time axis while events that last for a period of time have a begin and an end point. It can be difficult to distinguish between punctual events and events with a short duration. Furthermore, the documents typically do not report precise starting and ending times for events, hence we decided to distinguish between events that happened at a *Single Day* and *Multi-Day Events* that span over multiple days. We used days as the smallest granularity for the annotation as none of the annotated articles contained any information on the hour, the minute or the second when the event happened. In case a corpus contains this information, the annotation scheme could be extended to include this information as well.

For *Single Day Events*, the event time is written in the format *YYYY-MM-DD*. For *Multi-Day Events*, the annotator annotates the *begin point* and the *end point* of the event. In case no statement can be made on when an event happened, the event will be annotated with the label *not applicable*. This applies only to 0.67% of the annotated events in the TimeBank Corpus which is mainly due to annotation errors in the TimeBank Corpus.

> *He was **sent** into space on May 26, 1980. He **spent** six days aboard the Salyut 6 spacecraft.*

The first event in this text, ***sent***, will be annotated with the event time *1980-05-26*. The second event, ***spent***, is a Multi-Day Event and is annotated with the event time *beginPoint=1980-05-26* and *endPoint=1980-06-01*.

In case the exact event time is not stated in the document, the annotators are asked to narrow down the possible event time as precisely as possible. For this purpose, they can annotate the event time with *after YYYY-MM-DD* and *before YYYY-MM-DD*.

> *In 1996 he was **appointed** military attache at the Hungarian embassy in Washington. [...] McBride was **part** of a seven-member crew aboard the Orbiter Challenger in October 1984*

The event ***appointed*** is annotated *after 1996-01-01 before 1996-12-31* as the event must have happened sometime in 1996. The Multi-Day Event

part is annotated with *beginPoint=after 1984-10-01 before 1984-10-31* and *endPoint=after 1984-10-01 before 1984-10-31*.

To speed up the annotation process, annotators were allowed to write *YYYY-MM-xx* to express that something happened sometime within the specified month and *YYYY-xx-xx* to express that the event happened sometime during the specified year. Annotators were also allowed to annotate events that happened at the Document Creation Time with the label *DCT*.

The proposed annotation scheme requires that event mentions are already annotated. For our annotation study we used the event mentions that were already defined in the TimeBank Corpus. In contrast to the annotation of TLINKs, temporal expressions must not be annotated in the corpus.

4 Annotation Study

The annotation study was performed on the same subset of documents as used by the TimeBank-Dense Corpus (Cassidy et al., 2014) with the event mentions that are present in the TempEval-3 dataset (UzZaman et al., 2013). Cassidy et al. selected 36 random documents from the TimeBank Corpus (Pustejovsky et al., 2003). These 36 documents include a total of 1498 annotated events. This allows to compare our annotations to those of the TimeBank-Dense Corpus (see section 5).

Each document has been independently annotated by two annotators according to the annotation scheme introduced above. We used the freely available WebAnno (Yimam et al., 2013). To speed up the annotation process, the existent temporal expressions that are defined in the TimeBank Corpus were highlighted. These temporal expressions are in principle not required to perform our annotations, but the highlighting of them helps to determine the event time. Figure 1 depicts a sample annotation made by WebAnno. The two annotators were trained on 15 documents distinct from the 36 documents annotated for the study. During the training stage, the annotators discussed the decisions they have made with each other.

After both annotators completed the annotation task, the two annotations were curated by one person to derive one final annotation. The curator examined the events where the annotators disagreed and decided on the final annotation. The final annotation might be a merge of the two provided annotations.

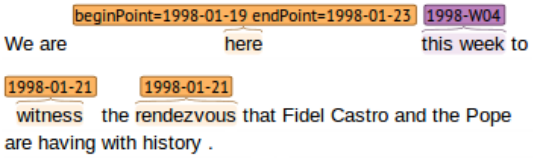


Figure 1: Sample Annotation made with WebAnno. The violet annotations are existing annotations of temporal expressions from the TimeBank Corpus. The span for the beige annotations, the event mentions, come also from the TimeBank Corpus. Our annotators added the value for the event time for those beige annotations.

4.1 Inter-Annotator-Agreement

We use Krippendorff's α (Krippendorff, 2004) with the nominal metric to compute the Inter-Annotator-Agreement (IAA). The nominal metric considers all distinct labels equally distant from one another, i.e. partial agreement is not measured. The annotators must therefore completely agree.

Using this metric, the Krippendorff's α for the 36 annotated documents is $\alpha = 0.617$. Cassidy et al. (2014) reported a Kappa agreement between $0.56 - 0.64$ for their annotation of TLINKs. Comparing these numbers is difficult, as the annotation tasks were different. According to Landis and Koch (1977), these numbers lie on the border of a moderate and a substantial level of agreement.

4.2 Disagreement Analysis

In 648 out of 1498 annotated events, the annotators disagreed on the event time. In 42.3% of the disagreements, the annotators disagreed on whether the event mention is a Single Day Event or a Multi-Day Event. Such disagreement occurs when it is unclear from the text whether the event lasted for one or for several days. For example, an article reported on a meeting and due to a lack of precise temporal information in the document, one annotator assumed that the meeting lasted for one day, the other that it lasted for several days. A different source for the disagreement has been the annotation of states. They can either be annotated with the date where the text gives evidence that they hold true, or they can be annotated as a Multi-Day Event that begins before that date and ends after that date.

Different annotations for Multi-Day Events account for 231 out of the 648 disagreements (35.6%). In this category, the annotators disagreed

on the begin point in 110 cases (47.6%), on the end point in 57 cases (24.7%) and on the begin as well as on the end point in 64 cases (27.7%). The Krippendorff's α for all begin point annotations is 0.629 and for all end point annotations it is 0.737.

A disagreement on Single Day Events was observed for 143 event mentions and accounts for 22.1% of the disagreements. The observed agreement for Single Day Events is 80.5% or $\alpha = 0.799$. Most disagreements for Single Day Events were whether the event occurred on the same date as the document was written or if it occurred before the document was written.

4.3 Measuring Partial Agreement

One issue of the strict nominal metric is that it does not take partial agreement into account. In several cases, the two annotators agreed in principle on the event time, but might have labeled it slightly differently. One annotator might have taken more clues from the text into account to narrow down when an event has happened. One annotator for example, has annotated an event with the label *after 1998-08-01 before 1998-08-31*. The second annotator has taken an additional textual clue into account, which was that the event must have happened in the first half of August 1998 and annotated it as *after 1998-08-01 before 1998-08-15*. Even though both annotators agree in principle, when using the nominal metric it would be considered as a distinct annotation.

To measure this effect, we created a relaxed metric to measure mutual exclusivity:

$$d_{ME}(a,b) = \begin{cases} 1 \text{ if } a \text{ and } b \text{ are mutual exclusive} \\ 0 \text{ else} \end{cases}$$

The metric measures whether two annotations can be satisfied at the same time. Given the event happened on August 5th, 1998, then the two annotations *after 1998-08-01 before 1998-08-31* and *after 1998-08-01 before 1998-08-15* would both be satisfied. In contrast, the two annotations *after 1998-02-01* and *before 1997-12-31* can never be satisfied at the same time and are therefore mutual exclusive.

Out of the 648 disagreements, 71 annotations were mutually exclusive. Computing the Krippendorff's α with the above metric yields a value of $\alpha_{ME} = 0.912$.

4.4 Annotation Statistics

Table 1 gives an overview of the assigned labels. Around 58.21% of the events are either instantaneous events or their duration is at most one day. 41.12% of the events are Multi-Day Events that take place over multiple days. While for Single Day Events there is a precise date for 55.73% of the events, the fraction is much lower for Multi-Day Events. In this category, only in 19.81% of the cases the begin point is precisely mentioned in the article and only in 15.75% of the cases, the end point is precisely mentioned.

The most prominent label for Single Day Events is the Document Creation Time (DCT). 48.28% of Single Day Events happened on the day the article was created, 33.49% of these events happened at least one day before the DCT and 17.43% of the mentions refer to future events. This distribution shows, that the news articles and TV broadcast transcripts from the TimeBank Corpus mainly report on events that happened on the same day.

For Multi-Day Events, the distribution looks different. In 76.46% of the cases, the event started in the past, and in 65.10% of the cases, it is still ongoing.

4.5 Most Informative Temporal Expression

Not all temporal expressions in a text are of the same relevance for an event. In fact, in many cases only a single temporal expression is of importance, which is the expression stating when the event occurred. Our annotations allow us to determine *most informative* temporal expression for an event. We define the most informative temporal expression as the expression that has been used by the annotator to determine the event time. We checked for all annotations whether the event date can be found as a temporal expression in the document and computed the distance to the closest one with a matching value. The distance is measured as the number of sentences. 421 out of 1498 events happened on the Document Creation Time and were excluded from this computation. The Document Creation Time is provided as additional metadata in the TimeBank Corpus, and it is often not explicitly mentioned in the document text.

Figure 2 shows the distance between the most informative temporal expression and the event mention. In 23.68% of the cases, the time expression is in the same sentence, and in 17.59% of the cases, the time expression is either in the

	# Events	%
Single Day Events	**872**	**58.21%**
with precise date	486	55.73%
after + before	145	16.63%
after	124	14.22%
before	117	13.42%
past events	292	33.49%
events at DCT	421	48.28%
future events	152	17.43%
Multi-Day Events	**616**	**41.12%**
precise begin point	122	19.81%
precise end point	97	15.75%
begins in the past	471	76.46%
begins on the DCT	38	6.17%
begins in the future	105	17.05%
ends in the past	179	29.06%
ends on the DCT	26	4.22%
ends in the future	401	65.10%
Not applicable	**10**	**0.67%**

Table 1: Statistic on the annotated event times. Single Day Events happen on a single day, Multi-Day Events take place over multiple days. The event time can either be precise or the annotators used *before* and *after* to narrow down the event time, e.g. the event has happened in a certain month and year. DCT = Document Creation Time.

next or in the previous sentence. It follows that in 58.72%, of the cases the most informative time expression cannot be found in the same or in the preceding or succeeding sentence. This is important to note, as previous shared tasks like TempEval-1,-2, and -3 (Verhagen et al., 2007; Verhagen et al., 2010; UzZaman et al., 2013) and previous annotation studies like the TimeBank-Dense Corpus (Cassidy et al., 2014) only considered the relation between event mentions and temporal expressions in the same and in adjacent sentences. However, for the majority of events, the most informative temporal expression is not in the same or in the preceding / succeeding sentence.

For 7.31% of the annotated events, no matching temporal expression was found in the document. Those were mainly events where the event time was inferred by the annotators from multiple temporal expressions in the document. An example is that the year of the event was mentioned in the beginning of the document and the month of the event was mentioned in a later part of the document.

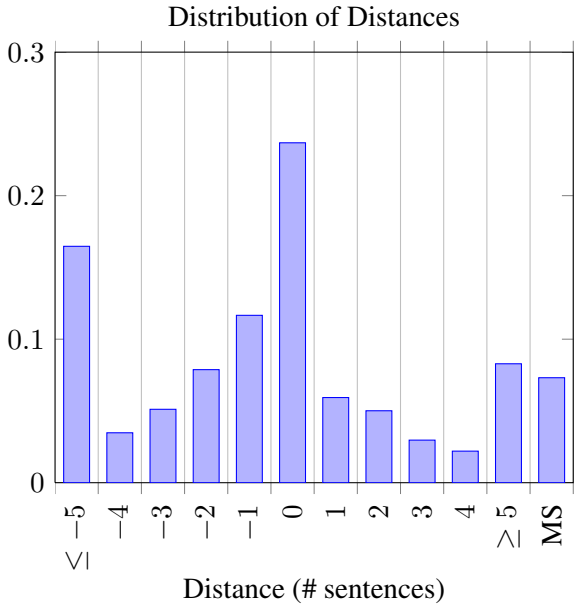


Figure 2: Distribution of distances in sentences between the event mention and the most informative temporal expression. For 58.72% of the event mentions, the most informative time expression is not in the same or in the previous/next sentence. For 7.3% of the mentions, the time expression originates from multiple sources (MS).

5 Comparison of Annotation Schemes

Depending on the application scenario and the text domain, the use of TLINKs or the proposed annotation scheme may be advantageous. TLINKs have the capability to capture the temporal order of events, even when temporal expressions are completely absent in a document, which is often the case for novels. The proposed annotation scheme has the advantage that temporal information, independent where and in which form it is mentioned in the document, can be taken into account. However, the proposed scheme requires that the events can be anchored on a time axis, which is easy for news articles and encyclopedic text but hard for novels and narratives.

In this section, we evaluate the application scenario of temporal knowledge base population and time-aware information retrieval. For temporal knowledge base population, it is important to derive the date for facts and events as precisely as possible (Surdeanu, 2013). Those facts can either be instantaneous, e.g. a person died, or they can last for a longer time like a military conflict.

Similar requirements are given for time-aware information retrieval, where it can be important to know at which point in time something occurred (Kanhabua and Nørvåg, 2012).

We use the TimeBank-Dense Corpus (Cassidy et al., 2014) with its TLINKs annotations and compare those to our event time annotations. The TimeBank-Dense Corpus annotated all TLINKs between Event-Event, Event-Time, and Time-Time pairs in the same sentence and between succeeding sentences as well as all Event-DCT and Time-DCT pairs. Six different link types were defined: `BEFORE`, `AFTER`, `INCLUDES`, `IS_INCLUDED`, `SIMULTANOUS`, and `VAGUE`, where `VAGUE` encodes that the annotators where not able to make a statement on the temporal relation of the pair.

We studied how well the event time is captured by the dense TLINK annotation. We used transitive closure rules as described by Chambers et al. (2014) to deduct also TLINKs for pairs that were not annotated. For example, when $event_1$ happened before $event_2$ and $event_2$ happened before $date_1$, we can infer that $event_1$ happened before $date_1$. Using this transitivity allows to infer relations for pairs that are more than one sentence apart. For all annotated events, we evaluated all TLINKs, including the TLINKs inferred from the transitivity rules, and derived the event time as precisely as possible. We then computed how precise the inferred event times are in comparison to our annotations. Preciseness is measured in the number of days. An event that is annotated with *1998-02-13* has the preciseness of 1 day. If the inferred event time from the TLINKs is *after 1998-02-01 and before 1998-02-15*, then the preciseness is 15 days. A more precise anchoring is preferred.

The TimeBank-Dense Corpus does not have a link type to mark that an event has started or ended at a certain time point. This makes the TLINK annotation impractical for the durative events that span over multiple days. According to our annotation study, 41.12% of the events in the TimeBank Corpus last for longer time periods. For these 41.12%, it cannot be inferred from when to when the events lasted.

In 487 out of the 872 Single Day Events (55.85%), the TLINKs give a result with the same precision as our annotations. For 198 events (22.71%), our annotation is more precise, i.e. the time window where the event might have happened is smaller. For 187 events (21.44%), no event time could be inferred from the TLINKs. This is due to the fact that there was no link to any temporal expression even when transitivity was taken into account.

For the 487 events where the TLINKs resulted in an event time as precise as our annotation, the vast majority of them were events that happened at the Document Creation Time. As depicted in Table 1, 421 events happened at DCT. For those events the precise date can directly be derived from the annotated link between each event mention and the DCT. For all other events that did not happen at the Document Creation Time, the TLINKs result for the most cases in a less precise anchoring in time and for around a fifth of these cases in no temporal anchoring at all while we do anchor them.

We can conclude, that even a dense TLINK annotation gives suboptimal information on when events have happened, and due to the restriction that TLINKs are only annotated in the same and in adjacent sentences, a lot of relevant temporal information gets lost.

6 Automated Event Time Extraction

In this section, we present a baseline system for automatic event time extraction. The system uses temporal relations in which the event is involved and anchors the event to the most precise time. For this purpose, we have defined a two-step process to determine the events' time. Given a set of documents in which the events and time expressions are already annotated, the system first obtains a set of possible times for each of the events. Second, the most precise time is selected or generated for each event.

For the first step, we use the multi-pass architecture introduced by Chambers et al. (2014) that was trained and evaluated on the TimeBank-Dense Corpus (Cassidy et al., 2014). Chambers et al. describe multiple rules and machine learning based classifiers to extract relations between events and temporal expressions. This architecture extracts temporal relations of the type `BEFORE`, `AFTER`, `INCLUDES`, `IS_INCLUDED`, and `SIMULTANOUS`. The classifiers are combined into a precision-ranked cascade of sieves. The architecture presented by Chambers et al. does not produce temporal information that an event has started or ended at a certain time point and can

therefore only be used for Single Day Events.

We use these sieves to add the value of the temporal expression and the corresponding relation to a set of possible times for each event. In fact, for each event we generate a set of <relation, time> tuples in which the event is involved.

> *Police **confirmed Friday** that the body found along a highway*

For example, the one sieve adds [IS_INCLUDED, $Friday_{1998-02-13}$] and a second sieve adds [BEFORE, $DCT_{1998-02-14}$] to the set of possible event times for the *confirmed* event.

Applying the sequence of the sieves will obtain all various temporal links for each event. In the next step, if the event has a relation of type SIMULTANEOUS, IS_INCLUDED or INCLUDES, the system sets the event time to the value of the time expression. If the event has a relation of type BEFORE and/or AFTER, the system narrows down the event time as precisely as possible. If the sieve determines the relation type as VAGUE, the set of possible event times remains unchanged.

Algorithm 1 demonstrates how the event time is selected or generated from a set of possible times.

Algorithm 1 Automatic Event Time Extraction

```
function EventTime(times)
    if times is empty then
        return 'Not Available'                ▷ the event has no non-vague relation
    end if
    min_before_time = DATE.MAX_VALUE
    max_after_time = DATE.MIN_VALUE
    for [relation, time] in times do
        if relation is SIMULTANEOUS or IS_INCLUDED or INCLUDES then
            return time
        else if relation is BEFORE and time < min_before_time then
            min_before_time = time
        else if relation is AFTER and time > max_after_time then
            max_after_time = time
        end if
    end for
    event_time = AFTER + max_after_time + BEFORE + min_before_time
    return event_time
end function
```

Applying the proposed method on the TimeBank-Dense Corpus, we obtained some value for the event time for 593 of 872 (68%) Single Day Events. For 359 events (41%), the system generates the event time with the same precision as our annotations. Table 2 gives statistics of the automatically obtained event times.

To evaluate the output of the proposed system, we evaluated how precise the automatically obtained event times are in comparison with our annotations. Table 3 shows for 41% of events, the proposed system generates the same event time as our annotations. For 21% events our annotation is more precise, i.e. the time window where the event might have happened is smaller. For 47 events (5.38%), the system infers an event time that is in conflict with the human annotation, for example a disagreement if an event happened before or after DCT. Considering event times that have the same preciseness as our annotations as true positives, the precision of the proposed system is 60.54% and the recall is 41.17% for Single Day Events. As presented in section 4, human annotators agree in 80.50% of the cases on the label for Single Day Events. The less precise and non-inferred event times are mainly due to the fact that temporal expressions, that are more than one sentence apart, are not taken into account by the sieve architecture.

Single Day Events	# Events	%
with precise date	260	29.82%
after + before	16	1.84%
after	99	11.35%
before	218	25%
not available	279	31.99%

Table 2: Statistics on the automatically obtained event times for events happened on a single day. The obtained event time can either be precise or the system used *before* and *after* to narrow down the event time. For 279 events, the system cannot infer any event time.

Obtained event time	# Events	%
same as human annotation	359	41.17%
less precise	187	21.44%
conflicting annotations	47	5.38%
cannot infer event time	279	31.99%
Precision		**60.54%**
Recall		**41.17%**
F_1-Score		**49.01%**
Human F_1-Score		**80.50%**

Table 3: Evaluation results of proposed system in comparison with our annotations.

In this work we focused on the automated anchoring of Single Day Events and presented a baseline system that relies on the work of Chambers et al. (2014). The F_1-score with 49.01% is in comparison to the human score of 80.50% comparatively low. However, only in 5.38% of the cases, the automatically inferred event time is plain wrong. In the most cases, no event time could be inferred (31.99%) or it was less precise

than the human annotation (21.44%).

Extending the described approach to Multi-Day-Events is not straight forward. The TimeBank-Dense Corpus and consequently the system by Chambers et al. does not include a TLINK type to note that an event has started or ended at a certain date, hence, extracting the begin point and end point for Multi-Day-Events is not possible. A fundamental adaption of the system by Chambers et al. would be required.

In contrast to Single Day Events, extracting the event time for Multi-Day Events requires more advanced logic. The start date of the event must be before the end date of the event. The relation to events that are included in the Multi-Day Events must be checked to avoid inconsistencies. The development of an automated system for Multi-Day Events is subject of our ongoing work.

7 Conclusion

We presented a new annotation scheme for anchoring events in time and annotated a subset of the TimeBank Corpus (Pustejovsky et al., 2003) using this annotation scheme. The annotation guidelines as well as the annotated corpus are publicly available.[2] In the performed annotation study, the Krippendorff's α inter-annotator agreement was considerably high at $\alpha = 0.617$. The largest disagreement resulted from events in which it was not explicitly mentioned when the event happened. Using a more relaxed measure for Krippendorff's α which only assigns a distance to mutual exclusive annotations, the agreement changed to $\alpha_{ME} = 0.912$. We can conclude that after little training, annotators are able to perform the annotation with a high agreement.

The effort for annotating TLINKs on the other hand scales quadratic with the number of events and temporal expressions. This imposes the often used restriction that only temporal links between events and temporal expressions in the same or in succeeding sentences are annotated. Even with this restriction, the annotation effort is quite significant, as on average 6.3 links per mention must be annotated. As Figure 2 depicts, in more than 58.72% of the cases the most informative temporal expression is more than one sentence apart from the event mention. As a consequence, inferring from TLINKs when an event happened is less precise as temporal information that is more than one sentence away can often not be taken into account.

For the 872 Single Day Events, the correct event time could be inferred from the TLINKs only in 487 cases. For 187 Single Day Events, no event time at all could be inferred, as no temporal expression was within the one sentence window of that event.

A drawback of the proposed scheme is the lack of temporal ordering of events beyond the smallest unit of granularity, which was in our case one day. The scheme is suitable to note that several events occurred at the same date, but their order on that date cannot be encoded. In case the temporal ordering is important for the application scenario, the annotation scheme could be extended and TLINKs could be annotated for events that fall on the same date. Another option is to increase the granularity, but this requires that the information in the documents also allow this more precise anchoring.

[2] https://www.ukp.tu-darmstadt.de/data/timeline-generation/temporal-anchoring-of-events/

Acknowledgement

This work has been supported by the German Research Foundation as part of the Research Training Group Adaptive Preparation of Information from Heterogeneous Sources (AIPHES) under grant No. GRK 1994/1 and by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under grant No. I/82806. Additional support was provided by the German Federal Ministry of Education and Research (BMBF) as a part of the Software Campus program under the promotional reference 01-S12054.

References

Philip Bramsen, Pawan Deshpande, Yoong Keok Lee, and Regina Barzilay. 2006. Inducing Temporal Graphs. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 189–198, Stroudsburg, PA, USA. Association for Computational Linguistics.

Taylor Cassidy, Bill McDowell, Nathanael Chambers, and Steven Bethard. 2014. An Annotation Framework for Dense Event Ordering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 501–506, Baltimore, Maryland, USA. Association for Computational Linguistics.

Nathanael Chambers, Taylor Cassidy, Bill McDowell, and Steven Bethard. 2014. Dense Event Ordering

with a Multi-Pass Architecture. *Transactions of the Association for Computational Linguistics*, 2:273–284.

Quang Xuan Do, Wei Lu, and Dan Roth. 2012. Joint Inference for Event Timeline Construction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 677–687, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nattiya Kanhabua and Kjetil Nørvåg. 2012. Learning to Rank Search Results for Time-sensitive Queries. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2463–2466, New York, NY, USA. ACM.

Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. 2012. Extracting Narrative Timelines As Temporal Dependency Structures. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 88–97, Stroudsburg, PA, USA. Association for Computational Linguistics.

Klaus Krippendorff. 2004. *Content Analysis: An Introduction to Its Methodology (second edition)*. Sage Publications.

J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics*, 33(1):159–174.

James Pustejovsky, Patrick Hanks, Roser Sauri, A. See, Robert Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, D. Day, Lisa Ferro, and Marcia Lazo. 2003. The TIMEBANK Corpus. In *Proceedings of Corpus Linguistics 2003*, pages 647–656, Lancaster, UK.

James Pustejovsky. 1991. The Syntax of Event Structure. *Cognition 41 (1991)*, pages 47–81.

Roser Saurí, Jessica Littman, Robert Gaizauskas, Andrea Setzer, and James Pustejovsky. 2004. TimeML Annotation Guidelines, Version 1.2.1.

Andrea Setzer. 2001. *Temporal Information in Newswire Articles: An Annotation Scheme and Corpus Study*. Ph.D. thesis, University of Sheffield, Sheffield, UK.

Mihai Surdeanu. 2013. Overview of the TAC 2013 Knowledge Base Population Evaluation: English Slot Filling and Temporal Slot Filling. In *Proceedings of the TAC-KBP 2013 Workshop*, Gaithersburg, Maryland, USA.

Naushad UzZaman, Hector Llorens, Leon Derczynski, Marc Verhagen, James F. Allen, and James Pustejovsky. 2013. SemEval-2013 Task 1: TempEval-3: Evaluating Time Expressions, Events, and Temporal Relations. In *Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013)*, pages 1–9, Atlanta, Gerogia, USA.

Marieke van Erp, Piek Vossen, Rodrigo Agerri, Anne-Lyse Minard, Manuela Speranza, Ruben Urizar, Egoitz Laparra, Itziar Aldabe, and German Rigau. 2015. Annotated Data, version 2. Technical report, Amsterdam, Netherlands.

Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Graham Katz, and James Pustejovsky. 2007. SemEval-2007 Task 15: TempEval Temporal Relation Identification. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, SemEval '07, pages 75–80, Stroudsburg, PA, USA. Association for Computational Linguistics.

Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. 2010. SemEval-2010 Task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, SemEval '10, pages 57–62, Stroudsburg, PA, USA. Association for Computational Linguistics.

Seid Muhie Yimam, Iryna Gurevych, Richard Eckart de Castilho, and Chris Biemann. 2013. WebAnno: A Flexible, Web-based and Visually Supported System for Distributed Annotations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 1–6, Sofia, Bulgaria, August. Association for Computational Linguistics.

Grammatical Error Correction: Machine Translation and Classifiers

Alla Rozovskaya
Department of Computer Science
Virginia Tech
Blacksburg, VA 24060
alla@vt.edu

Dan Roth
Department of Computer Science
University of Illinois
Urbana, IL 61820
danr@illinois.edu

Abstract

We focus on two leading state-of-the-art approaches to grammatical error correction – machine learning classification and machine translation. Based on the comparative study of the two learning frameworks and through error analysis of the output of the state-of-the-art systems, we identify key strengths and weaknesses of each of these approaches and demonstrate their complementarity. In particular, the machine translation method learns from parallel data without requiring further linguistic input and is better at correcting complex mistakes. The classification approach possesses other desirable characteristics, such as the ability to easily generalize beyond what was seen in training, the ability to train without human-annotated data, and the flexibility to adjust knowledge sources for individual error types.

Based on this analysis, we develop an algorithmic approach that combines the strengths of both methods. We present several systems based on resources used in previous work with a relative improvement of over 20% (and 7.4 F score points) over the previous state-of-the-art.

1 Introduction

For the majority of English speakers today, English is not the first language. These writers make a variety of grammar and usage mistakes that are not addressed by standard proofing tools. Recently, there has been a spike in research on grammatical error correction (GEC), correcting writing mistakes made by learners of English as a Second Language, including four shared tasks: HOO (Dale and Kilgarriff, 2011; Dale et al., 2012) and CoNLL (Ng et al., 2013; Ng et al., 2014). These shared tasks facilitated progress on the problem within the framework of two leading methods – machine learning classification and statistical machine translation (MT).

System	Method	Performance		
		P	R	F0.5
CoNLL-2014 top 3	MT	41.62	21.40	35.01
CoNLL-2014 top 2	Classif.	41.78	24.88	36.79
CoNLL-2014 top 1	MT, rules	39.71	30.10	37.33
Susanto et al. (2014)	MT, classif.	53.55	19.14	39.39
Miz. & Mats. (2016)	MT	45.80	26.60	40.00
This work	MT, classif.	60.17	25.64	**47.40**

Table 1: **(Lack of) progress in GEC over the last few years**.

The top CoNLL system combined a rule-based module with MT (Felice et al., 2014). The second system that scored almost as highly used machine learning classification (Rozovskaya et al., 2014), and the third system used MT (Junczys-Dowmunt and Grundkiewicz, 2014). Furthermore, Susanto et al. (2014) showed that a combination of the two methods is beneficial, but the advantages of each method have not been fully exploited.

Despite success of various methods and the growing interest in the task, the key differences between the leading approaches have not been identified or made explicit, which could explain the lack of progress on the task. Table 1 shows existing state-of-the-art since CoNLL-2014. The top results are close, suggesting that several groups have competitive systems. Two improvements (of $<$3 points) were published since then (Susanto et al., 2014; Mizumoto and Matsumoto, 2016).

The purpose of this work is to gain a better understanding of the values offered by each method and to facilitate progress on the task, building on the advantages of each approach. Through better understanding of the methods, we exploit the strengths of each technique and, building on existing architecture, develop superior systems within

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2205–2215,
Berlin, Germany, August 7-12, 2016.

each framework. Further combination of these systems yields even more significant improvements over existing state-of-the-art. We make the following contributions:

• We examine two state-of-the-art approaches to GEC and identify strengths and weaknesses of the respective learning frameworks.

• We perform an error analysis of the output of two state-of-the-art systems, and demonstrate how the methods differ with respect to the types of language misuse handled by each.

• We exploit the strengths of each framework: with classifiers, we explore the ability to learn from native data, i.e. without supervision, and the flexibility to adjust knowledge sources to specific error types; with MT, we leverage the ability to learn without further linguistic input and to better identify complex mistakes that cannot be easily defined in a classifier framework.

• As a result, we build several systems that combine the strengths of both frameworks and demonstrate substantial progress on the task. Specifically, the best system outperforms the previous best result by 7.4 F score points.

Section 2 describes related work. Section 3 presents error analysis. In Section 4, we develop classifier and MT systems that make use of the strengths of each framework. Section 5 shows how to combine the two approaches. Section 6 concludes.

2 Related Work

We first introduce the CoNLL-2014 shared task and briefly describe the state-of-the-art GEC systems in the competition and beyond. Next, an overview of the two leading methods is presented.

2.1 CoNLL-2014 shared task and approaches

CoNLL-2014 training data (henceforth CoNLL-train) is a corpus of learner essays (1.2M words) written by students at the National University of Singapore (Dahlmeier et al., 2013), corrected and error-tagged. The CoNLL-2013 test set was included in CoNLL-2014 and is used as development. Both the development and the test sets are also from the student population studying at the same University but annotated separately. We report results on the CoNLL-2014 test.

The annotation includes specifying the relevant correction as well as the information about each error type. The tagset consists of 28 categories. Table 2 illustrates the 11 most frequent errors in the development data; errors are marked with an asterisk, and ∅ denotes a missing word. The majority of these errors are related to grammar but also include *mechanical*, *collocation*, and other errors.

An F-based scorer, named M2, was used to score the systems (Dahlmeier and Ng, 2012). The metric in CoNLL-2014 was F0.5, i.e. weighing precision twice as much as recall. Two types of annotations were used: original and revised. We follow the recommendations of the organizers and use the original data (Ng et al., 2014).

The approaches varied widely: classifiers, MT, rules, hybrid systems. Table 3 summarizes the top five systems. The top team used a hybrid system that combined rules and MT. The second system developed classifiers for common grammatical errors. The third system used MT.

As for external resources, the top 1 and top 3 teams used additional learner data to train their MT systems, the Cambridge University Press Learners' Corpus and the Lang-8 corpus (Mizumoto et al., 2011), respectively. Many teams also used native English datasets. The most common ones are the Web1T corpus (Brants and Franz, 2006), the CommonCrawl dataset, which is similar to Web1T, and the English Wikipedia. Several teams used off-the-shelf spellcheckers.

In addition, Susanto et al. (2014) made an attempt at combining MT and classifiers. They used CoNLL-train and Lang-8 as non-native data and English Wikipedia as native data. We believe that the reason this study did not yield significant improvements (Table 1) is that individual strengths of each framework have not been fully exploited. Further, each system was applied separately and decisions were combined using a general MT combination technique (Heafield et al., 2009). Finally, Mizumoto and Matsumoto (2016) attempt to improve an MT system also trained on Lang-8 with discriminative re-ranking using part-of-speech (POS) and dependency features but only obtain a small improvement. These results suggest that standard combination and re-ranking techniques are not sufficient.

2.2 Overview of the State-of-the-Art

The statistical machine translation approach is based on the noisy-channel model. The best translation for a foreign sentence f is:

$$e^* = \arg\max_e p(e)p(f|e)$$

Error type	Rel. freq. (%)	Examples
Article (*ArtOrDet*)	19.93	*∅/*The* government should help encourage *the/∅ breakthroughs as well as *a/∅ complete medication system .
Wrong collocation (*Wci*)	12.51	Some people started to **think/wonder* if electronic products can replace human beings for better performances .
Noun number (*Nn*)	11.44	There are many reports around the internet and on newspaper stating that some users ' **iPhone/iPhones* exploded .
Preposition (*Prep*)	8.98	I do not agree **on/with* this argument...
Word form (*Wform*)	6.56	...the application of surveillance technology serves as a warning to the **murders/murderers* and they might not commit more murder .
Orthography/punctuation (*Mec*)	5.75	Even British Prime Minister , Gordon Brown *∅/, has urged that all cars in **britain/Britain* to be green by 2020 .
Verb tense (*Vt*)	4.56	Through the thousands of years , most Chinese scholars **are/{have been}* greatly affected by Confucianism .
Linking words/phrases (*Trans*)	4.10	**However/Although* , video surveillance may be a great help .
Local redundancy (*Rloc-*)	3.70	Some solutions *{*as examples*}/∅ would be to design plants/fertilizers that give higher yield ...
Subject-verb agreement (*SVA*)	3.58	However , tracking people **are/is* different from tracking goods .
Verb form (*Vform*)	3.52	Travelers survive in desert thanks to GPS **guide/guiding* them .

Table 2: **Example errors.** In the parentheses, the error codes used in the shared task are shown. Errors exemplifying the relevant phenomena are marked; the sentences may contain other mistakes.

Rank	System name	F0.5	Approach	External training data		External error modules
				Native data	Learner data	
1	CAMB	37.33	Rules and MT	Microsoft Web LM	Cambridge Corpus, Eng. Vocab Profile	Cambridge "Write and Improve"
2	CUUI	36.79	Classif.; patterns	Web1T		
3	AMU	35.01	MT	Wikipedia, CommonCrawl	Lang-8	
4	POST	30.88	LM and rules	Web1T		PyEnchant Spell
5	NTHU	29.92	Rules, MT, classif.	Web1T, Gigaword, BNC, Google Books		Spellcheckers: Aspell, GingerIt

Table 3: **The top 5 systems in CoNLL-2014.** The last column lists external proofing tools used. LM stands for *language models*.

The model consists of two components: a *language model* assigning a probability $p(e)$ for any target sentence *e*, and a translation model that assigns a conditional probability $p(f|e)$. The language model is learned using a monolingual corpus in the target language. The parameters of the *translation model* are estimated from a parallel corpus, i.e. the set of foreign sentences and their corresponding translations into the target language. In error correction, the task is cast as translating from erroneous learner writing into corrected well-formed English. The MT approach relies on the availability of a parallel corpus for learning the translation model. In case of error correction, a set of learner sentences and their corrections functions as a parallel corpus.

State-of-the-art MT systems are phrase-based, i.e. parallel data is used to derive a phrase-based lexicon (Koehn et al., 2003). The resulting lexicon consists of a list of pairs (seq_f, seq_e) where seq_f is a sequence of one or more foreign words, seq_e is a predicted translation. Each pair comes with an associated score. At decoding time, all phrases from sentence f are collected with their corresponding translations observed in training. These are scored together with the language modeling scores and may include other features. The phrase-based approach by Koehn et al. (2003) uses a log-linear model (Och and Ney, 2002), and the best correction maximizes the following:

$$\begin{aligned} e^* &= \arg\max_e P(e|f) \\ &= \arg\max_e \exp(\sum_{m=1}^{M} \lambda_m h_m(e,f)) \end{aligned} \quad (1)$$

where h_m is a feature function, such as language model score and translation scores, and λ_m corresponds to a feature weight.

The classifier approach is based on the context-sensitive spelling correction methodology (Golding and Roth, 1996; Golding and Roth, 1999; Banko and Brill, 2001; Carlson et al., 2001; Carlson and Fette, 2007) and goes back to earlier approaches to article and preposition error correction (Izumi et al., 2003; Han et al., 2006; Gamon et al., 2008; Felice and Pulman, 2008; Tetreault et al., 2010; Gamon, 2010; Dahlmeier and Ng, 2011; Dahlmeier and Ng, 2012). The classifier approach to error correction has been prominent for a long time before MT, since building a classifier does not require having annotated learner data.

Property	MT	Classifier
(1a) **Error coverage**: ability to address a wide variety of error phenomena	**+**All errors occurring in the training data are automatically covered	**-**Only errors covered by the classifiers; new errors need to be added explicitly
(1b) **Error complexity**: ability to handle complex and interacting mistakes that go beyond word boundaries	**+**Automatically through parallel data, via phrase-based lexicons	**-**Need to develop via specific approaches
(2) **Generalizability**: going beyond the error confusions observed in training	**-**Only confusions observed in training can be corrected	**+**Easily generalizable via confusion sets and features
(3) **Supervision/Annotation**: role of learner data in training the system	**-**Required	**+**Not required
(4) **System flexibility**: adapting knowledge sources per error phenomena	**-**Not easy to integrate error-specific knowledge resources	**+**Flexible; phenomenon-specific knowledge sources

Table 4: **Summary of the key properties of the MT and the classifier-based approaches.** We use $+$ and $-$ to indicate a positive or a negative value with respect to each factor.

Classifiers are trained individually for a specific error type. Because an error type needs to be defined, typically only well-defined mistakes can be addressed in a straightforward way. Given an error type, a *confusion set* is specified and includes a list of confusable words. For some errors, confusion sets are constructed using a closed list (e.g. prepositions). For other error types, NLP tools are required. To identify locations where an article was likely omitted incorrectly, for example, a phrase chunker is used. Each occurrence of a confusable word in text is represented as a vector of features derived from a *context window* around the target. The problem is cast as a multi-class classification task.

In the classifier paradigm, there are various algorithms – generative (Gamon, 2010; Park and Levy, 2011), discriminative (Han et al., 2006; Gamon et al., 2008; Felice and Pulman, 2008; Tetreault et al., 2010), and joint approaches (Dahlmeier and Ng, 2012; Rozovskaya and Roth, 2013). Earlier works trained on native data (due to lack of annotation). Later approaches incorporated learner data in training in various ways (Han et al., 2010; Gamon, 2010; Rozovskaya and Roth, 2010a; Dahlmeier and Ng, 2011).

3 Error Analysis of MT and Classifiers

This section presents error analysis of the MT and classifier approaches. We begin by identifying several key properties that distinguish between MT systems and classifier systems and that we use to characterize the learning frameworks and the outputs of the systems:

(1a) Error coverage denotes the ability of a system to identify and correct a variety of error types.

(1b) Error complexity indicates the capacity of a system to address complex mistakes such as those where multiple errors interact.

(2) Generalizibility refers to the ability of a system to identify mistakes in new unseen contexts and propose corrections beyond those observed in training data.

(3) The role of supervision or having annotated learner data for training.

(4) System flexibility is a property of the system that allows it to adapt resources specially to correct various phenomena. The two paradigms are summarized in Table 4. We use $+$ and $-$ to indicate whether a learning framework has desirable (+) or undesirable characteristic with regard to each factor.

The first three properties characterize system output, while (3) and (4) arise from the system frameworks. Below we analyze the output of several state-of-the-art CoNLL-2014 systems in more detail.[1] Section 4 explores (3) and (4) that relate to the learning frameworks.

3.1 Error Coverage and Complexity

Error coverage To understand how systems differ with respect to error coverage, we consider recall of each system per error type. Error-type *recall* can be easily computed using error tags and is reported in the CoNLL overview paper.

The recall numbers show substantial variations among the systems. If we consider error categories that have non-negligible recall numbers (higher than 10%), classifier-based approaches have a much lower proportion of error types for which 10% recall was achieved. Among the 28 error types, the top classifier systems – Columbia University-University of Illinois (CUUI, top-2) and National Tsing Hua University (NTHU, top-5) – have a recall higher than 10% for 8 and 9 error types, respectively. In contrast, the two MT-based systems – Cambridge University (CAMB,

[1] Outputs are available on the CoNLL-2014 website.

(1)	It is a concern that will be with us *{*during our whole life*}/{*for our entire life*} .
(2)	The decision to inform relatives of *{*such genetic disorder*}/{*such genetic disorders*} will be dependent . . .
(3)	.. we need to respect it and we have no right *{*in saying*}/{*to say*} that he must tell his relatives about it .
(4)	...and his family might be a *{*genetically risked*}/{*genetic risk*} family .
(5)	...he was **diagnosis*/{*diagnosed with*} a kind of genetic disease which is very serious .
(6)	The situation may become **worst*/*worse* if the child has diseases like cancer or heart disease . . .

Table 5: **Complex and interacting mistakes that MT successfully addresses.** Output of the MT-based AMU system.

top-1) and the Adam Mickiewicz University system (AMU, top-3) – have 15 and 17 error types, respectively, for which the recall is at least 10%.

These recall discrepancies indicate that the MT approach has a better overall coverage, which is intuitive given that all types of confusions are automatically added through phrase-based translation tables in MT, while classifiers must explicitly model each error type. Note, however, that these numbers do not necessarily indicate good type-based performance, since high recall may correspond to low precision.

Error complexity In the MT approach, error confusions are learned automatically via the phrase translation tables extracted from the parallel training data. Thus, an MT system can easily handle interacting and complex errors where replacements involve a sequence of words. Table 5 illustrates complex and interacting mistakes that the MT approach is able to handle. Example (1) contains a phrase-level correction that includes both a preposition replacement and an adjective change. (2) is an instance of an interacting mistake where there is a dependency between the article and the noun number, and a mistake can be corrected by changing one of the properties but not both. (3), (4) and (5) require multiple simultaneous corrections on various words in a phrase. (6) is an example of an incorrect adjectival form, an error that is typically not modeled with standard classifiers.

3.2 Generalizability

Because MT systems extract error/correction pairs from phrase-translation tables, they can only identify erroneous surface forms observed in training and propose corrections that occurred with the corresponding surface forms. Crucially, in a standard MT scenario, any resulting translation consists of "matches" mined from the translation tables, so a standard MT model lacks lexical abstractions that might help generalize, thus out-of-vocabulary words is a well-known problem in MT (Daume and Jagarlamudi, 2011). While more advanced MT models can abstract by adding higher-level features such as POS, previous attempt yielded only marginal improvements (Mizumoto and Matsumoto, 2016), since one typically needs different types of abstractions depending on the error type, as we show below.

Error type	AMU (MT)			CUUI (Classif.)		
	P	R	F0.5	P	R	F0.5
Orthog./punc. (*Mec*)	61.6	16.3	**39.6**	53.3	8.7	26.4
Article (*ArtOrDet*)	38.0	10.9	25.4	31.8	47.9	**34.0**
Preposition (*Prep*)	54.9	10.4	**29.5**	31.7	8.8	20.9
Noun number (*Nn*)	49.6	43.2	**48.2**	42.5	46.2	43.2
Verb tense (*Vt*)	30.2	9.3	20.8	61.1	5.4	19.9
Subj.-verb agr. (*SVA*)	48.3	14.9	33.3	57.7	57.7	**57.7**
Verb form (*Vform*)	40.5	16.8	31.8	69.2	15.1	**40.3**
Word form (*Wform*)	59.0	36.6	**52.6**	60.0	13.5	35.6

Table 6: **Performance of MT and classifier systems from CoNLL-2014 on common errors.**

With classifiers, it is easy to generalize using higher-level information that goes beyond surface form and to adjust the abstraction to the error type. Many grammatical errors may benefit from generalizations based on POS or parse information; we can thus expect that classifiers will do better on errors that require linguistic abstractions.

To validate this hypothesis, we evaluate type-based performance of two systems: a top-3 MT-based AMU system and a top-2 classifier-based CUUI; we do not include the top-1 system, since it is a hybrid system that also uses rules.

Unlike recall, estimating type-based *precision* requires knowing the type of the correction supplied by the system, which is not specified in the output. We thus manually analyze the output of the AMU and CUUI systems for seven common error categories and assign to each correction an appropriate type to estimate precision and F0.5 (Table 6). The CUUI system addresses all of these errors, with the exception of mechanical (Mec), of which it handles a small subset. The AMU system does better on mechanical, preposition, word form, and noun number. CUUI does better on articles, verb agreement, and verb form.

We now consider examples of errors that are corrected by the classifier-based CUUI system in these three categories but are missed by the MT-based AMU system (Table 7). Examples (1) and

(1)	**Long-distance dependencies: verb agreement** As a result , in the case that when one of the members **happen/happens* to feel uncomfortable or abnormal , he or she should be aware that . . .
(2)	A study of New York University in 2010 shown that patients with family members around generally **recovers/recover* 2-4 days faster than those taken care by professional nurses .
	Confusions not found in training: verb agreement and verb form
(3)	Hence , the social media sites **serves/serve* as a platform for the connection .
(4)	After **came/coming* back from the hospital , the man told his parents that the problem was that he carried . . .
(5)	social media is the only resource they can approach to know everything **happened/happening* in their country . . .
	Superfluous words: articles
(6)	For **an/∅* example , if exercising is helpful, we can always look for more chances for the family to go exercise .
(7)	. . . as soon a person is made aware of his or her genetic profile , he or she has **a/∅* knowledge about others .
	Omissions: articles
(8)	In this case , if one of the family members or close relatives is found to carry **∅/a* genetic risk . . .

Table 7: **Generalizing beyond surface form:** Examples of mistakes that classifiers successfully address. Output of the classifier-based CUUI system.

(2) illustrate verb errors with long-distance subjects ("one" and "patients"). This is handled in the classification approach via syntactic features. An MT system misses these errors because it is limited to edits within short spans. Examples (3), (4), and (5) illustrate verb mistakes for which the correct replacements were not observed in training but that are nonetheless corrected by generalizing beyond surface form. Finally, (6) and (7) illustrate omission and insertion errors, a majority of article mistakes. The MT system is especially bad at correcting such mistakes. Notably, the classifier-based CUUI system correctly identified *twice* as many omitted articles and more than *20 times* more superfluous articles than the MT-based AMU system. This happens because an MT system is restricted to suggesting deletions and insertions in those contexts that were observed in training, whereas a classifier uses shallow parse information, which allows it to insert or delete an article in front of every eligible noun phrase. These examples demonstrate that the ability of a system to generalize beyond the surface forms is indeed beneficial for *long-distance dependencies*, for abstracting away from surface forms when *formulating confusion sets*, and for mistakes involving *omitting or inserting a word*.

4 Developing New State-of-the-Art MT and Classifier Systems

In this section, we explore the advantages of each learning approach, as identified in the previous section, within each learning framework. To this end, drawing on the strengths of each framework, we develop new state-of-the-art MT and classifier systems.[2] In the next section, we will use these MT and classifier components and show how to exploit the strengths of each framework in combination. Table 8 summarizes the data used. Results are reported with respect to all errors in the test data. This is different from performance for individual errors in Table 6.

System	Learner		Native	
	CoNLL-train 1.2M	Lang-8 48M	Eng. Wiki. 2B	Web1T 1T
MT	✓	✓	✓	-
Classif.	✓	-	-	✓

Table 8: **Data used in the experiments.** Corpora sizes are in the number of words.

4.1 Machine Translation Systems

A key advantage of the MT framework is that, unlike with classifiers, error confusions are learned from parallel data automatically, without further (linguistic) input. We build two MT systems that differ only in the use of parallel data: the CoNLL-2014 training data and Lang-8. Our MT systems are trained using Moses (Koehn et al., 2007) and follow the standard approach (Junczys-Dowmunt and Grundkiewicz, 2014; Susanto et al., 2014). Both systems use two 5-gram language models – English Wikipedia and the corrected side of CoNLL-train – trained with KenLM (Heafield et al., 2013). Table 9 reports the performance of the systems. As shown, performance increases by more than 11 points when a larger parallel corpus is used. The best MT system outperforms the top CoNLL system by 2 points.

4.2 Classifiers

We now present several classifier systems, exploring the two important properties of the classification framework – the ability to train without super-

[2]Implementation details can be found at `cogcomp.cs.illinois.edu/page/publication_view/793`

Parallel data	Performance		
	P	R	F0.5
CoNLL-train	43.34	11.81	28.25
Lang-8	66.15	15.11	39.48
CoNLL-2014 top 1	39.71	30.10	37.33

Table 9: **MT systems trained in this work**.

vision and system flexibility (see Table 4).

4.2.1 Supervision

Supervision in the form of annotated learner data plays an important role in developing an error correction system but is expensive. Native data, in contrast, is cheap and available in large quantities. Therefore, the fact that, unlike with MT, it is possible to build a classifier system without any annotated data, is a clear advantage of classifiers.

Training without supervision is possible in the classification framework, as follows. For a given mistake type, e.g. preposition, a classifier is trained on native data that is assumed to be correct; the classifier uses context words around each preposition as features. The resulting model is then applied to learner prepositions and will predict the most likely preposition in a given context. If the preposition predicted by the classifier is different from what the author used in text, this preposition is flagged as a mistake. We refer the reader to Rozovskaya and Roth (2010b) and Rozovskaya and Roth (2011) for a description of training classifiers with and without supervision for error correction tasks. Below, we address two questions related to the use of supervision:

• **Training with supervision**: When training using learner data, how does a classifier-based system compare against an MT system?

• **Training without supervision**: How well can we do by building a classifier system with native data only, compared to MT and classifier-based systems that use supervision?

Our classifier system is based on the implementation framework of the second CoNLL-2014 system (Rozovskaya et al., 2014) and consists of classifiers for 7 most common grammatical errors in CoNLL-train: article; preposition; noun number; verb agreement; verb form; verb tense; word form. All modules take as input the corpus documents pre-processed with a POS tagger[3] (Even-Zohar and Roth, 2001), a shallow parser[4] (Punyakanok and Roth, 2001), a syntactic parser (Klein and Manning, 2003) and a dependency converter (Marneffe et al., 2006).

[3] http://cogcomp.cs.illinois.edu/page/software_view/POS

[4] http://cogcomp.cs.illinois.edu/page/software_view/Chunker

System	Performance		
	P	R	F0.5
Classifiers (learner)	32.15	17.96	27.76
Classifiers (native)	38.41	23.05	33.89
MT	43.34	11.81	28.25
CoNLL-2014 top 1	39.71	30.10	37.33
CoNLL-2014 top 2	41.78	24.88	36.79
CoNLL-2014 top 3	41.62	21.40	35.01

Table 10: **Classifier systems trained with and without supervision.** *Learner* data refers to CoNLL-train. *Native* data refers to Web1T. The MT system uses CoNLL-train for parallel data.

Classifiers are trained either on learner data (CoNLL-train) or native data (Web1T). Classifiers built on CoNLL-train are trained discriminatively with the Averaged Perceptron algorithm (Rizzolo and Roth, 2010) and use rich POS and syntactic features tailored to specific error types that are standard for these tasks (Lee and Seneff, 2008; Han et al., 2006; Tetreault et al., 2010; Rozovskaya et al., 2011); Naïve Bayes classifiers are trained on Web1T with word n-gram features. A detailed description of the classifiers and the features used can be found in Rozovskaya and Roth (2014). We also add several novel ideas that are described below.

Table 10 shows the performance of two classifier systems, trained with supervision (on CoNLL-train) and without supervision on native data (Web1T), and compares these to an MT approach trained on CoNLL-train. The first classifier system performs comparably to the MT system (27.76 vs. 28.25), however, the native-trained classifier system outperforms both, and does not use any annotated data. The native-trained classifier system would place fourth in CoNLL-2014.

4.2.2 Flexibility

We now explore another advantage of the classifier-based approach, that of allowing for a flexible architecture where we can tailor knowledge sources for individual phenomena. In Section 4.2.1, we already took advantage of the fact that in the classifier framework it is easy to incorporate features suited to individual error types. We now show that by *adding supervision in a way tailored toward specific errors* we can further improve the classifier-based approach.

Adding Supervision in a Tailored Way There is a trade-off between training on native and learner

	Training data	Performance P	R	F0.5
(1)	Learner	32.15	17.96	27.76
(2)	Native	38.41	23.05	33.89
(3)	Tailored	57.07	14.74	**36.26**

Table 11: **Classifiers: supervision in a tailored way**. Trained on (1) learner data (CoNLL-train); (2) native data (Web1T); (3) data sources tailored per error type.

data. The advantage of training on native data is clearly the size, which is important for estimating context parameters. Learner data provides additional information, such as learner error patterns and the manner of non-native writing.

Instead of choosing to train on one data type, the classifier framework allows one to combine the two data sources in various ways: voting (Rozovskaya et al., 2014), alternating structure optimization (Dahlmeier and Ng, 2011), training a meta-classifier (Gamon, 2010), and extracting error patterns (Rozovskaya and Roth, 2011). We compare two approaches of adding supervision:
(1) **Learner error patterns:** Error patterns are extracted from learner data and "injected" into models trained on native data (Rozovskaya and Roth, 2011). Learner data is used to estimate mistake parameters; contextual cues are based on native data.
(2) **Learner error patterns+native predictions:** Classifiers are trained on native data. Classifier predictions are used as features in models trained on learner data. Learner data thus contributes both the specific manner of learner writing and the mistake parameters. The native data contributes contextual information.

We found that (2) is superior to (1) for article, agreement, and preposition errors; (1) works better on verb form and word form errors; and noun number errors perform best when a classifier is trained on native data. (Learner error patterns were found not to be beneficial for correcting noun number errors (Rozovskaya and Roth, 2014)). Tailored supervision yields an improvement of almost 3 points over the system trained on native data and almost 9 points over the system trained on learner data (Table 11).

Adding Mechanical Errors Finally, we add components for *mechanical* errors: punctuation, spelling, and capitalization. These are distinguished from the grammatical mistakes, as they are not specific to GEC and can be handled with existing resources or simple methods.

For capitalization and missing commas, we compile a list of patterns using CoNLL training data. We also use an off-the-shelf speller (Flor, 2012; Flor and Futagi, 2012). Results are shown in Table 12. Performance improves by almost 5 and 7 points for the native-trained system and for the best configuration of classifiers with supervision. Both systems also outperform the top CoNLL system, by 1 and 6 points, respectively. The result of 43.11 by the best classifier configuration substantially outperforms the existing state-of-the-art: a combination of two MT systems and two classifier systems, and MT with re-ranking (Susanto et al., 2014; Mizumoto and Matsumoto, 2016).

System/training data	Performance P	R	F0.5
Native	38.41	23.05	33.89
Native+mechanical	42.72	27.69	**38.54**
Tailored	57.07	14.74	36.26
Best (tailored+mechanical)	60.79	19.93	**43.11**
CoNLL-2014 top system	39.71	30.10	37.33
Susanto et al. (2014)	53.55	19.14	39.39
Miz. & Mats. (2016)	45.80	26.60	40.00

Table 12: **Classifier systems in this work.** Comparison to existing state-of-the-art.

System	Performance P	R	F0.5
MT is trained on CoNLL-train			
MT	43.34	11.81	28.25
Spelling+MT	49.86	16.36	35.37
Article+MT	45.11	13.99	31.22
Verb agr.+MT	46.36	14.63	32.33
Art.+Verb agr.+Spell+MT	52.07	20.89	**40.10**
MT is trained on Lang-8			
MT	66.15	15.11	39.48
Spelling+MT	65.87	16.94	41.75
Article+MT	63.81	17.70	41.95
Verb. agr.+MT	66.09	18.01	43.08
Art.+Verb agr.+Spell+MT	64.13	22.15	**46.51**

Table 13: **Pipelines: select classifiers and MT.**

5 Combining MT and Classifier Systems

Since MT and classifiers differ with respect to the types of errors they can better handle, we combine these systems in a pipeline architecture where the MT is applied to the output of classifiers. Classifiers are applied first, since MT is better at handling complex phenomena. First, we add the speller and those classifier components that perform substantially better than MT (articles and verb agreement), due to the ability of classifiers to generalize beyond lexical information. The added classifiers are part of the best system in Table 12.

Results are shown in Table 13. Adding classifiers improves the performance, thereby demon-

System	Performance		
	P	R	F0.5
MT (CoNLL-train)	43.34	11.81	28.25
MT (Lang-8)	66.15	15.11	39.48
Best classifier (Table 12)	60.79	19.93	43.11
Best class.+MT (CoNLL-train)	51.92	25.08	42.77
Best class.+MT (Lang-8)	60.17	25.64	**47.40**

Table 14: **Pipelines: the best classifier system and MT systems.**

System	Performance		
	P	R	F0.5
Best classifier (Table 12)	60.79	19.93	**43.11**
Art.+Verb agr.+Spell+MT	64.13	22.15	**46.51**
Best classifier+MT	60.17	25.64	**47.40**
CoNLL-2014 top system	39.71	30.10	37.33
Susanto et al. (2014)	53.55	19.14	39.39
Miz. & Mats. (2016)	45.80	26.60	40.00

Table 15: **Best systems in this work.** Comparison to existing state-of-the-art.

strating that the classifiers address a complementary set of mistakes. Adding all three modules improves the results from 28.25 to 40.10 and from 39.48 to 46.51 for the MT systems trained on CoNLL-train and Lang-8, respectively. Notably, the CoNLL-train MT system especially benefits, which shows that when the parallel data is small, it is particularly worthwhile to add classifiers.

It should be stressed that even with a smaller parallel corpus, when the three modules are added, the resulting system is very competitive with previous state-of-the-art that uses a lot more supervision: Susanto et al. (2014) and Mizumoto and Matsumoto (2016) use Lang-8. These results show that *when one has an MT system, it is possible to improve by investing effort into building select classifiers for phenomena that are most challenging for MT*.

Finally, Table 14 demonstrates that combining MT with the best classifier system improves the result further when the MT system is trained on Lang-8, but not when the MT system is trained on CoNLL-train. We also note that the CoNLL-train MT system also has a much lower precision than the other systems. We conclude that when only a limited amount of data is available, the classifier approach on its own performs better.

As a summary, Table 15 lists the best systems developed in this work – a classifier system, a pipeline of select classifiers and MT, and a pipeline consisting of the best classifier and the MT systems – and compares to existing state-of-the-art. Our classifier system is a 3-point improvement over the existing state-of-the-art, while the best pipeline is a 7.4-point improvement (20% relative improvement).

6 Discussion and Conclusions

A recent surge in GEC research has produced two leading state-of-the-art approaches – machine learning classification and machine translation. Based on the analysis of the methods and an error analysis on the outputs of state-of-the-art systems that adopt these approaches, we explained the differences and the key advantages of each. With respect to error phenomena, we showed that while MT is better at handling complex mistakes, classifiers are better at correcting mistakes that require abstracting beyond lexical context. We further showed that the key strengths of the classification framework are its flexibility and the ability to train without supervision.

We built several systems that draw on the strengths of each approach individually and in a pipeline. The best classifier system and the pipelines outperform reported best results on the task, often by a large margin.

The purpose of this work is to gain a better understanding of the advantages offered by each learning method in order to make further progress on the GEC task. We showed that the values provided by each method can be exploited within each approach and in combination, depending on the resources available, such as annotated learner data (MT), and additional linguistic resources (classifiers). As a result, we built robust systems and showed substantial improvement over existing state-of-the-art.

For future work, we intend to study the problem in the context of other languages. However, it is important to realize that the problem is far from being solved even in English, and the current work makes very significant progress on it.

Acknowledgments

The authors thank Michael Flor for his help with running the spelling system on the data and John Wieting for sharing the English Wikipedia corpus. The authors are also grateful to Mark Sammons, Peter Chew, and the anonymous reviewers for the insightful comments on the paper. The work of Dan Roth on this project was supported by DARPA under agreement number FA8750-13-2-0008. Any opinions, findings, conclusions or recommendations are those of the authors and do not necessarily reflect the view of the agencies.

References

M. Banko and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *Proceedings*

of 39th Annual Meeting of the Association for Computational Linguistics, pages 26–33, Toulouse, France, July.

T. Brants and A. Franz. 2006. *Web 1T 5-gram Version 1*. Linguistic Data Consortium.

A. Carlson and I. Fette. 2007. Memory-based context-sensitive spelling correction at web scale. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*.

A. J. Carlson, J. Rosen, and D. Roth. 2001. Scaling up context sensitive text correction. In *IAAI*.

D. Dahlmeier and H. T. Ng. 2011. Grammatical error correction with alternating structure optimization. In *Proceedings of ACL*.

D. Dahlmeier and H.T Ng. 2012. A beam-search decoder for grammatical error correction. In *Proceedings of EMNLP-CoNLL*.

D. Dahlmeier, H.T. Ng, and S.M. Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the NAACL Workshop on Innovative Use of NLP for Building Educational Applications*.

R. Dale and A. Kilgarriff. 2011. Helping Our Own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*.

R. Dale, I. Anisimoff, and G. Narroway. 2012. A report on the preposition and determiner error correction shared task. In *Proceedings of the NAACL Workshop on Innovative Use of NLP for Building Educational Applications*.

H. Daume and J. Jagarlamudi. 2011. Domain adaptation for machine translation by mining unseen words. In *ACL*.

Y. Even-Zohar and D. Roth. 2001. A sequential model for multi class classification. In *Proceedings of EMNLP*.

R. De Felice and S. Pulman. 2008. A classifier-based approach to preposition and determiner error correction in L2 English. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 169–176, Manchester, UK, August.

M. Felice, Z. Yuan, Ø. Andersen, H. Yannakoudakis, and E. Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

M. Flor and Y. Futagi. 2012. On using context for automatic correction of non-word misspellings in student essays. In *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics.

M. Flor. 2012. Four types of context for automatic spelling correction. *Traitement Automatique des Langues (TAL). (Special Issue: Managing noise in the signal: error handling in natural language processing)*, 3(53):61–99.

M. Gamon, J. Gao, C. Brockett, A. Klementiev, W. Dolan, D. Belenko, and L. Vanderwende. 2008. Using contextual speller techniques and language modeling for ESL error correction. In *Proceedings of IJCNLP*.

M. Gamon. 2010. Using mostly native data to correct errors in learners' writing. In *Proceedings of NAACL*.

A. R. Golding and D. Roth. 1996. Applying Winnow to context-sensitive spelling correction. In *ICML*.

A. R. Golding and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*.

N. Han, M. Chodorow, and C. Leacock. 2006. Detecting errors in English article usage by non-native speakers. *Journal of Natural Language Engineering*, 12(2):115–129.

N. Han, J. Tetreault, S. Lee, and J. Ha. 2010. Using an error-annotated learner corpus to develop and ESL/EFL error correction system. In *Proceedings of LREC*.

K. Heafield, G. Hanneman, and A. Lavie. 2009. Machine translation system combination with flexible word ordering. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*. Association for Computational Linguistics.

K. Heafield, I. Pouzyrevsky, J. H. Clark, and P. Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *ACL*.

E. Izumi, K. Uchimoto, T. Saiga, T. Supnithi, and H. Isahara. 2003. Automatic error detection in the Japanese learners' English spoken data. In *Proceedings of ACL*.

M. Junczys-Dowmunt and R. Grundkiewicz. 2014. The AMU system in the CoNLL-2014 shared task: Grammatical error correction by data-intensive and feature-rich statistical machine translation. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*.

D. Klein and C. D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Proceedings of NIPS*.

P. Koehn, F. J. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *ACL*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *ACL*.

J. Lee and S. Seneff. 2008. An analysis of grammatical errors in non-native speech in English. In *Proceedings of the 2008 Spoken Language Technology Workshop*.

M. Marneffe, B. MacCartney, and Ch. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*.

T. Mizumoto and Y. Matsumoto. 2016. Discriminative reranking for grammatical error correction with statistical machine translation. In *NAACL. To appear*.

T. Mizumoto, M. Komachi, M. Nagata, and Y. Matsumoto. 2011. Mining revision log of language learning SNS for automated japanese error correction of second language learners. In *IJCNLP*.

H. T. Ng, S. M. Wu, Y. Wu, Ch. Hadiwinoto, and J. Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of CoNLL: Shared Task*.

H. T. Ng, S. M. Wu, T. Briscoe, C. Hadiwinoto, R. H. Susanto, and C. Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of CoNLL: Shared Task*.

F. J. Och and H. Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *ACL*.

A. Park and R. Levy. 2011. Automated whole sentence grammar correction using a noisy channel model. In *ACL*, Portland, Oregon, USA, June. Association for Computational Linguistics.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *Proceedings of NIPS*.

N. Rizzolo and D. Roth. 2010. Learning Based Java for Rapid Development of NLP Systems. In *Proceedings of LREC*.

A. Rozovskaya and D. Roth. 2010a. Generating confusion sets for context-sensitive error correction. In *Proceedings of EMNLP*.

A. Rozovskaya and D. Roth. 2010b. Training paradigms for correcting errors in grammar and usage. In *Proceedings of NAACL*.

A. Rozovskaya and D. Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In *Proceedings of ACL*.

A. Rozovskaya and D. Roth. 2013. Joint learning and inference for grammatical error correction. In *Proceedings of EMNLP*.

A. Rozovskaya and D. Roth. 2014. Building a State-of-the-Art Grammatical Error Correction System. In *Transactions of ACL*.

A. Rozovskaya, M. Sammons, J. Gioja, and D. Roth. 2011. University of Illinois system in HOO text correction shared task. In *Proceedings of the European Workshop on Natural Language Generation (ENLG)*.

A. Rozovskaya, K.-W. Chang, M. Sammons, D. Roth, and N. Habash. 2014. The University of Illinois and Columbia system in the CoNLL-2014 shared task. In *Proceedings of CoNLL Shared Task*.

R. H. Susanto, P. Phandi, and H. T. Ng. 2014. System combination for grammatical error correction. In *EMNLP*.

J. Tetreault, J. Foster, and M. Chodorow. 2010. Using parse features for preposition selection and error detection. In *Proceedings of ACL*.

Recurrent neural network models for disease name recognition using domain invariant features

Sunil Kumar Sahu and **Ashish Anand**
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Assam, India - 781039
{sunil.sahu, anand.ashish}@iitg.ernet.in

Abstract

Hand-crafted features based on linguistic and domain-knowledge play crucial role in determining the performance of disease name recognition systems. Such methods are further limited by the scope of these features or in other words, their ability to cover the contexts or word dependencies within a sentence. In this work, we focus on reducing such dependencies and propose a domain-invariant framework for the disease name recognition task. In particular, we propose various end-to-end recurrent neural network (RNN) models for the tasks of disease name recognition and their classification into four pre-defined categories. We also utilize convolution neural network (CNN) in cascade of RNN to get character-based embedded features and employ it with word-embedded features in our model. We compare our models with the state-of-the-art results for the two tasks on NCBI disease dataset. Our results for the disease mention recognition task indicate that state-of-the-art performance can be obtained without relying on feature engineering. Further the proposed models obtained improved performance on the classification task of disease names.

1 Introduction

Automatic recognition of disease names in biomedical and clinical texts is of utmost importance for development of more sophisticated NLP systems such as information extraction, question answering, text summarization and so on (Rosario and Hearst, 2004). Complicate and inconsistent terminologies, ambiguities caused by use of abbreviations and acronyms, new disease names, multiple names (possibly of varying number of words) for the same disease, complicated syntactic structure referring to multiple related names or entities are some of the major reasons for making automatic identification of the task difficult and challenging (Leaman et al., 2009). State-of-the-art disease name recognition systems (Mahbub Chowdhury and Lavelli, 2010; Doğan and Lu, 2012; Dogan et al., 2014) depends on user defined features which in turn try to capture context keeping in mind above mentioned challenges. Feature engineering not only requires linguistic as well as domain insight but also is time consuming and is corpus dependent.

Recently window based neural network approach of (Collobert and Weston, 2008; Collobert et al., 2011) got lot of attention in different sequence tagging tasks in NLP. It gave state-of-art results in many sequence labeling problems without using many hand designed or manually engineered features. One major drawback of this approach is its inability to capture features from outside window. Consider a sentence *"Given that the skin of these adult mice also exhibits signs of de novo hair-follicle morphogenesis, we wondered whether human pilomatricomas might originate from hair matrix cells and whether they might possess beta-catenin-stabilizing mutations"* (taken *verbatim* from PMID: 10192393), words such as *signs* and *originate* appearing both sides of the word *"pilomatricomas"*, play important role in deciding it is a disease. Any model relying on features defined based on words occurring within a fixed window of neighboring words will fail to capture information of influential words occurring outside this window.

Our motivation can be summarized in the following question: *can we identify disease name and categorize them without relying on feature en-*

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2216–2225,
Berlin, Germany, August 7-12, 2016.

gineering, domain-knowledge or task specific resources? In other words, we can say this work is motivated towards mitigating the two issues: first, feature engineering relying on linguistic and domain-specific knowledge; and second, bring flexibility in capturing influential words affecting model decisions irrespective of their occurrence anywhere within the sentence. For the first, we used character-based embedding (likely to capture orthographic and morphological features) as well as word embedding (likely to capture lexico-semantic features) as features of the neural network models.

For the second issue, we explore various recurrent neural network (RNN) architectures for their ability to capture long distance contexts. We experiment with bidirectional RNN (Bi-RNN), bidirectional long short term memory network (Bi-LSTM) and bidirectional gated recurrent unit (Bi-GRU). In each of these models we used sentence level log likelihood approach at the top layer of the neural architecture. The main contributions of the work can be summarized as follows

- Domain invariant features with various RNN architectures for the disease name recognition and classification tasks,
- Comparative study on the use of character based embedded features, word embedding features and combined features in the RNN models.
- Failure analysis to check where exactly our models are failed in the considered tasks.

Although there are some related works (discussed in sec 6), this is the first study, to the best of our knowledge, which comprehensively uses various RNN architectures without resorting to feature engineering for disease name recognition and classification tasks.

Our results show near state-of-the-art performance can be achieved on the disease name recognition task. More significantly, the proposed models obtain significantly improved performance on the disease name classification task.

2 Methods

We first give overview of the complete model used for the two tasks. Next we explained embedded features used in different neural network models. We provide short description of different RNN models in the section 2.3. Training and inference strategies are explained in the section 2.4.

2.1 Model Architectures

Similar to any named entity recognition task, we formulate the disease mention recognition task as a token level sequence tagging problem. Each word has to be labeled with one of the defined tags. We choose *BIO model* of tagging, where *B* stands for beginning, *I* for intermediate and *O* for outsider or other. This way we have two possible tags for all entities of interest, i.e., for all disease mentions, and one tag for other entities.

Generic neural architecture is shown in the figure 1. In the very first step, each word is mapped to its embedded features.

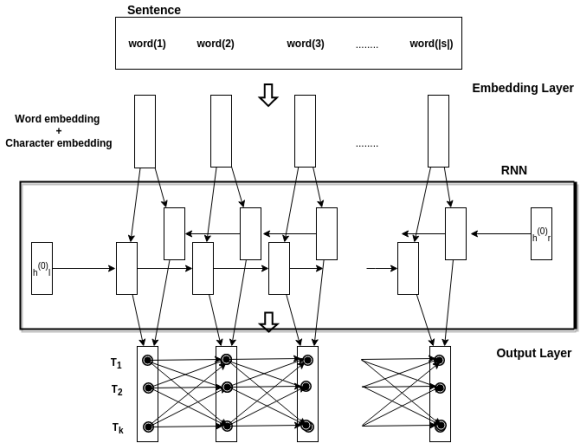


Figure 1: Generic bidirectional recurrent neural network with sentence level log likelihood at the top-layer for sequence tagging task

We call this layer as *embedding layer*. This layer acts as input to hidden layers of RNN model. We study the three different RNN models, and have described them briefly in the section 2.3. Output of the hidden layers is then fed to the output layer to compute the scores for all tags of interest (Collobert et al., 2011; Huang et al., 2015). In output layer we are using sentence level log likelihood, to make inference. Table 1 briefly describes all notations used in the paper.

2.2 Features

Distributed Word Representation (WE)

Distributed word representation or word embedding or simply word vector (Bengio et al., 2003; Collobert and Weston, 2008) is the technique of learning vector representation of a word in a given

Symbols	Explanation
V	Vocabulary of words $(v_1, v_2 ... v_{\|V\|})$
C	Vocabulary of characters $(c_1, c_2 .. c_{\|C\|})$
T	Tag set $(t_1, t_2 ... t_{\|T\|})$
d^{we}	Dimension of word embedding
d^{chr}	Dimension of character embedding
d^{ce}	Dimension of character level word embedding
$M^{we} \in \mathbb{R}^{d^{we} X \|V\|}$	word embedding matrix, where every column M_i^{we} is a vector representation of corresponding word v_i in *V*
$M^{cw} \in \mathbb{R}^{d^{chr} X \|C\|}$	character embedding matrix, where every column M_i^{cw} is a vector representation of corresponding character c_i in *C*.
$w^{(i)} \in \mathbb{R}^{d^{we}}$	word embedding of v_i
$y^{(i)} \in \mathbb{R}^{d^{ce}}$	character level word embedding of v_i
$x^{(i)} \in \mathbb{R}^{d^{we}+d^{ce}}$	feature vector of word $w^{(i)}$. We get this after concatenating $w^{(i)}$ and $y^{(i)}$
$z^{(i)} \in \mathbb{R}^{\|T\|}$	score for i^{th} word in sentence at output layer of neural network. Here j^{th} element will indicate the score for t_j^{th} tag.
W_*^*, U_*^*, V_*^*	Parameters of different neural networks

Table 1: Notation

corpus. Word vectors are present in columns of matrix M^{we}. We can get this vector by taking product of matrix M^{we} and *one hot vector* of v_i.

$$w^{(i)} = M^{we} h^{(i)} \quad (1)$$

Here $h^{(i)}$ is the one hot vector representation of i^{th} word in *V*. We use pre-trained 50 dimensional word vectors learned using skipgram method on a biomedical corpus (Mikolov et al., 2013b; Mikolov et al., 2013a; TH et al., 2015).

Character Level Word Embedding (CE)

Word embedding preserve syntactic and semantic information well but fails to seize morphological and shape information. However, for the disease entity recognition task, such information can play an important role. For instance, letter *-o-* in the word *gastroenteritis* is used to combine various body parts *gastro* for *stomach*, *enter* for *intestines*, and *itis* indicates *inflammation*. Hence taken together it implies *inflammation of stomach and intestines*, where *-itis* play significant role in determining it is actually a disease name.

Character level word embedding was first introduced by (dos Santos and Zadrozny, 2014) with the motivation to capture word shape and morphological features in word embedding. Character level word embedding also automatically mitigate the problem of out of vocabulary words as we can embed any word by its characters through character level embedding. In this case, a vector is initialized for every character in the corpus. Then we learn vector representation for any word by applying CNN on each vector of character sequence of that word as shown in figure 2. These character vectors will get update while training RNN in supervised manner only. Since number of characters in the dataset is not high we assume that every character vectors will get sufficient updation while training RNN itself.

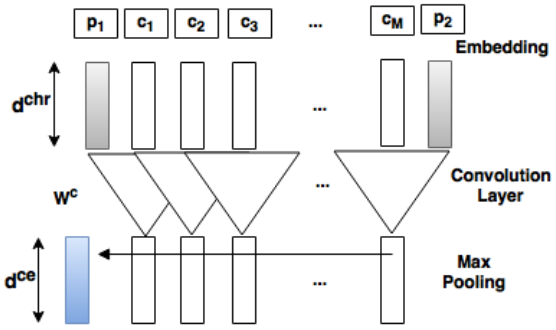


Figure 2: CNN with Max Pooling for Character Level Embedding (p_1 and p_2 are padding). Here, filter length is 3.

Let $\{p_1, c_1, c_2 ... c_M, p_2\}$ is sequence of characters for a word with padding at beginning and ending of word and let $\{a_l, a_1, a_2 ... a_M, a_r\}$ is its sequence of character vector, which we obtain by multiplying M^{cw} with one hot vector of corresponding character. To obtain character level word embedding we need to feed this in convolution neural network (CNN) with max pooling layer (dos Santos and Zadrozny, 2014). Let $W^c \in$

$\mathbb{R}^{d^{ce}X(d^{chr}Xk)}$ is a filter and b^c bias of CNN, then

$$[y^{(i)}]_j = \max_{1<m<M}[W^c q^{(m)} + b^c]_j \quad (2)$$

Here k is window size, $q^{(m)}$ is obtained by concatenating the vector of $(k-1)/2$ character left to $(k-1)/2$ character right of c_m. Same filter will be used for all window of characters and max pooling operation is performed on results of all. We learn 100 dimensional character embedding for all characters in a given dataset (avoiding case sensitivity) and 25 dimensional character level word embedding from character sequence of words.

2.3 Recurrent Neural Network Models

Recurrent Neural Network (RNN) is a class of artificial neural networks which utilizes sequential information and maintains history through its intermediate layers (Graves et al., 2009; Graves, 2013). We experiment with three different variants of RNN, which are briefly described in subsequent subsections.

Bi-directional Recurrent Neural Network

In Bi-RNN, context of the word is captured through past and future words. This is achieved by having two hidden components in the intermediate layer, as schematically shown in the fig 1. One component process the information in forward direction (left to right) and other in reverse direction. Subsequently outputs of these components then concatenated and fed to the output layer to get score for all tags of the considered word. Let $x^{(t)}$ is a feature vector of t^{th} word in sentence (concatenation of corresponding embedding features w^{t_i} and y^{t_i}) and $h_l^{(t-1)}$ is the computation of last hidden state at $(t-1)^{th}$ word, then computation of hidden and output layer values would be:

$$h_l^{(t)} = tanh(U^l x^{(t)} + W^l h_l^{(t-1)})$$

$$z^{(t)} = V(h_l^{(t)} : h_r^{(t)}) \quad (3)$$

Here $U^l \in \mathbb{R}^{n_H \times n_I}$ and $W^l \in \mathbb{R}^{n_H \times n_H}$, where n_I is input vector of length $d^{we} + d^{ce}$, n_H is hidden layer size and $V \in \mathbb{R}^{n_O \times (n_H + n_H)}$ is the output layer parameter. $h_l^{(t)}$ and $h_r^{(t)}$ correspond to left and right hidden layer components respectively and $h_r^{(t)}$ is calculated similarly to $h_l^{(t)}$ by reversing the words in the sentence. At the beginning $h_l^{(0)}$ and $h_r^{(0)}$ are initialized randomly.

Bi-directional Long Short Term Memory Network

Traditional RNN models suffer from both vanishing and exploding gradient (Pascanu et al., 2012; Bengio et al., 2013). Such models are likely to fail where we need longer contexts to do the job. These issues were the main motivation behind the LSTM model (Hochreiter and Schmidhuber, 1997). LSTM layer is just another way to compute a hidden state which introduces a new structure called a memory cell (c_t) and three gates called as input (i_t), output (o_t) and forget (f_t) gates.

These gates are composed of sigmoid activation function and responsible for regulating information in memory cell. The input gate by allowing incoming signal to alter the state of the memory cell, regulates proportion of history information memory cell will keep. On the other hand, the output gate regulates what proportion of stored information in the memory cell will influence other neurons. Finally, the forget gate can modulate the memory cells and allowing the cell to remember or forget its previous state. Computation of memory cell ($c^{(t)}$) is done through previous memory cell and candidate hidden state ($g^{(t)}$) which we compute through current input and the previous hidden state. The final output of hidden state would be calculated based on memory cell and forget gate.

In our experiment we used model discussed in (Graves, 2013; Huang et al., 2015). Let $x^{(t)}$ is feature vector for t^{th} word in a sentence and $h_l^{(t-1)}$ is previous hidden state then computation of hidden ($h_l^{(t)}$) and output layer ($z^{(t)}$) of LSTM would be.

$$\begin{aligned}
i_l^{(t)} &= \sigma(U_l^{(i)} x^{(t)} + W_l^{(i)} h_l^{(t-1)} + b_l^i) \\
f_l^{(t)} &= \sigma(U_l^{(f)} x^{(t)} + W_l^{(f)} h_l^{(t-1)} + b_l^f) \\
o_l^{(t)} &= \sigma(U_l^{(o)} x^{(t)} + W_l^{(o)} h_l^{(t-1)} + b_l^o) \\
g_l^{(t)} &= tanh(U_l^{(g)} x^{(t)} + W_l^{(g)} h_l^{(t-1)} + b_l^g) \\
c_l^{(t)} &= c_l^{(t-1)} * f_l + g_l * i_l \\
h_l^{(t)} &= tanh(c_l^{(t)}) * o_l
\end{aligned}$$

Where σ is sigmoid activation function, $*$ is a element wise product, $U_l^{(i)}, U_l^{(f)}, U_l^{(o)}, U_l^{(g)} \in \mathbb{R}^{n_H \times n_I}$ and $W_l^{(i)}, W_l^{(o)}, W_l^{(f)}, W_l^{(g)} \in \mathbb{R}^{n_H \times n_H}$, where n_I is input size ($d^{we} + d^{ce}$) and n_H is hidden layer size. We compute $h_r^{(t)}$ in similar manner as $h_l^{(t)}$ by reversing the all words of sentence. Let $V \in \mathbb{R}^{n_O \times (n_H + n_H)}$ (n_O size of output layer) is

the parameter of output layer of LSTM then computation of output layer will be:

$$z^{(t)} = V(h_l^{(t)} : h_r^{(t)}) \quad (4)$$

Bi-directional Gated Recurrent Unit Network

A gated recurrent unit (GRU) was proposed by (Cho et al., 2014) to make each recurrent unit to adaptively capture dependencies of different time scales. Similar to the LSTM unit, the GRU has gating units reset *r* and update z gates that modulate the flow of information inside the unit, however, without having a separate memory cells. The resulting model is simpler than standard LSTM models.

We follow (Chung et al., 2014) model of GRU to transform the extracted word embedding and character embedding features to score for all tags. Let $x^{(t)}$ embedding feature for *t*th word in sentence and $h_l^{(t-1)}$ is computation of hidden state for $(t-1)$th word then computation of GRU would be:

$$z_l^{(t)} = \sigma(U_l^{(z)} x^{(t)} + W_l^{(z)} h_l^{(t-1)} + b_l^{(z)})$$
$$r_l^{(t)} = \sigma(U_l^{(r)} x^{(t)} + W_l^{(r)} h_l^{(t-1)} + b_l^{(r)})$$
$$\tilde{h}_l^{(t)} = \tanh(U_l^{(h)} x^{(t)} + W_l^{(h)} h_l^{(t-1)} * r_l + b_l^{(h)})$$
$$h_l^{(t)} = z_l^{(t)} * \tilde{h}_l + (1 - z_l^{(t)}) * h_l^{(t-1)}$$

$$z^{(t)} = V(h_l^{(t)} : h_r^{(t)}) \quad (5)$$

Where $*$ is pair wise multiplication, $U_l^{(z)}, U_l^{(r)}$, $U_l^{(h)}, U_l^{(h)} \in \mathbb{R}^{n_H \times n_I}$ and $W_l^{(z)}, W_l^{(r)} W_l^{(h)} \in \mathbb{R}^{n_H \times n_H}$ are parameters of GRU. $V \in \mathbb{R}^{n_O \times (n_H + n_H)}$ is output layer parameter. Computation of $h_r^{(t)}$ is done in similar manner as $h_l^{(t)}$ by reversing the words of sentence.

2.4 Training and Inference

Equations 3, 4 and 5 are the scores of all possible tags for t^{th} word sentence. We follow *sentence-level log-likelihood (SLL)* (Collobert et al., 2011) approach equivalent to linear-chain CRF to infer the scores of a particular tag sequence for the given word sequence. Let $[w]_1^{|s|}$ is sentence and $[t]_1^{|s|}$ is the tag sequence for which we want to find the joint score, then score for the whole sentence with the particular tag sequence would be:

$$s([w]_1^{|s|}, [t]_1^{|s|}) = \sum_{1 \le i \le |s|} (W_{t_{i-1}, t_i}^{trans} + z_{t_i}^{(i)}), \quad (6)$$

where W^{trans} is transition score matrix and $W_{i,j}^{trans}$ is indicating the transition score moving from tag t_i to t_j; t_j is tag for the j^{th} word; $z_{t_i}^{(i)}$ is the output score from the neural network model for the tag t_i of i^{th} word. To train our model we used cross entropy loss function and adagrad (Duchi et al., 2010) approach to optimize the loss function. Entire neural network parameters, word embedding, character embedding and W^{trans} (transition score matrix used in the SLL) was updated during training. Entire code has been implemented using theano (Bastien et al., 2012) library in python language.

3 Experiments

3.1 Dataset

We used NCBI dataset (Doğan and Lu, 2012), the most comprehensive publicly available dataset annotated with disease mentions, in this work. NCBI dataset has been manually annotated by a group of medical practitioners for identifying diseases and their types in biomedical articles. All disease mentions were categorized into four different categories, namely, *specific disease*, *disease class*, *composite disease* and *modifier*. A word is annotated as *specific disease*, if it indicates a particular disease. *Disease class* category indicates a word describing a family of many specific diseases, such as autoimmune disorder. A string signifying two or more different disease mentions is annotated with *composite mention*. *Modifier* category indicates disease mention has been used as modifiers for other concepts. This dataset is a extension of the AZDC dataset (Leaman et al., 2009) which was annotated with disease mentions only and not with their categories. Statistics of the dataset is mentioned in the Table 2.

Corpus	***Train set***	***Dev set***	***Test set***
sentences	5661	939	961
disease	5148	791	961
spe. dis.	2959	409	556
disease class	781	127	121
modifier	1292	218	264
comp. men.	116	37	20

Table 2: Dataset statistics. spe. dis. : specific disease and comp. men.: composite mention

In our evaluation we used this dataset in two settings, **A**: *disease mention recognition*, where all

Task	Model	Validation Set			Test Set		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score
A	NN+CE	76.98	75.80	76.39	78.51	72.75	75.52
	Bi-RNN+CE	71.96	74.90	73.40	74.14	72.12	73.11
	Bi-GRU+CE	76.28	74.14	75.19	76.03	69.81	72.79
	Bi-LSTM+CE	81.52	72.86	**76.94**	76.98	75.80	**76.39**
B	NN+CE	67.27	53.45	59.57	67.90	49.95	57.56
	Bi-RNN+CE	61.34	56.32	58.72	60.32	57.28	58.76
	Bi-GRU+CE	61.94	59.11	**60.49**	62.56	56.50	59.38
	Bi-LSTM+CE	61.82	57.03	59.33	64.74	55.53	**59.78**

Table 3: Performance of various models using 25 dimensional CE features, A:Disease name recognition, B: Disease classification task

disease types are flattened into a single category and, the **B**: *disease class recognition*, where we need to decide exact categories of disease mentions. It is noteworthy to mention that the Task B is more challenging as it requires model to capture semantic contexts to put disease mentions into appropriate categories.

4 Results and Discussion

Evaluation of different models using CE

We first evaluate the performance of different RNNs using only character embedding features. We compare the results of RNN models with window based neural network (Collobert et al., 2011) using sentence level log likelihood approach (NN + CE). For the window based neural network, we considered window size 5 (two words from both left and right, and one central word) and same settings of character embedding were used as features. The same set of parameters are used in all experiments unless we mention specifically otherwise. We used exact matching scheme to evaluate performance of all models.

Table 3 shows the results obtained by different RNN models with only character level word embedding features. For the task A (Disease name recognition) Bi-LSTM and NN models gave competitive performance on the test set, while Bi-RNN and Bi-GRU did not perform so well. On the other hand for the task B, there is $2.08\% - 3.8\%$ improved performance (F1-score) shown by RNN models over the NN model again on the test set. Bi-LSTM model obtained F1-score of 59.78% while NN model gave 57.56%. As discussed earlier, task B is difficult than task A as disease category is more likely to be influenced by the words falling outside the context window considered in window based methods. This could be reason for RNN models to perform well over the NN model. This hypothesis will be stronger if we observe similar pattern in our other experiments.

Evaluation of different models with WE and WE+CE

Next we investigated the results obtained by the various models using only 50 dim word embedding features. The first part of table 4 shows the results obtained by different RNNs and the window based neural network (NN). In this case RNN models are giving better results than the NN model for both the tasks. In particular performance of Bi-LSTM models are best than others in both the tasks. We observe that for the task A, RNN models obtained 1.2% to 3% improvement in F1-score than the baseline NN performance. Similarly 2.55% to 4% improvement in F1-score are observed for the task B, with Bi-LSTM model obtaining more than 4% improvement.

In second part of this table we compare the results obtained by various models using the features set obtained by combining the two feature sets. If we look at performance of individual model using three different set of features, model using only word embedding features seems to give consistently best performance. Among all models, Bi-LSTM using word embedding features obtained best F1-scores of 79.13% and 63.16% for the tasks A and B respectively.

Importance of tuning pre-trained word vectors

We further empirically evaluate the importance of updating of word vectors while training. For this, we performed another set of experiments, where pre-trained word vectors are not updated while

Task	Model	Validation Set			Test Set		
		Precision	Recall	F1 Score	Precision	Recall	F1 Score
A	NN+WE	81.86	76.82	79.26	80.32	73.58	76.81
	Bi-RNN+WE	84.14	77.46	80.67	82.49	73.58	77.78
	Bi-GRU+WE	84.51	78.23	81.25	82.32	75.16	78.58
	Bi-LSTM+WE	85.13	77.72	**81.26**	84.87	74.11	**79.13**
B	NN+WE	65.33	56.43	60.55	64.23	57.14	60.48
	Bi-RNN+WE	63.62	56.84	60.04	67.47	57.50	62.09
	Bi-GRU+WE	66.42	57.41	61.59	68.25	58.58	63.05
	Bi-LSTM+WE	67.48	58.01	**62.39**	68.97	58.25	**63.16**
A	NN+WE+CE	76.37	78.62	**77.48**	74.92	75.16	75.04
	Bi-RNN+WE+CE	76.10	75.03	75.56	77.01	72.33	74.59
	Bi-GRU+WE+CE	77.73	76.44	77.08	78.04	73.38	**75.63**
	Bi-LSTM+WE+CE	76.94	77.34	77.14	76.10	74.11	75.09
B	NN+WE+CE	67.60	56.70	61.67	67.60	56.70	61.67
	Bi-RNN+WE+CE	60.94	61.34	61.14	64.36	60.90	62.58
	Bi-GRU+WE+CE	61.58	61.99	**61.78**	61.92	63.85	**62.87**
	Bi-LSTM+WE+CE	64.92	58.61	61.60	61.14	60.54	60.84

Table 4: Performance of various models using 50 dimensional WE features. A:Disease name recognition, B: Disease classification task

training. Results obtained on the validation dataset of the Task A are shown in the Table 5. One can observe that performance of all models have deteriorated. Next, instead of using pre-trained word vectors, we initialized each word with zero vector but kept updating them while training. Although performance (Table 6) deteriorated (compare to Table 4) but not as much as in table 5. This observation highlights the importance of tuning word vectors for a specific task during training.

Model	*P*	*R*	*F*
NN+WE	74.02	67.86	70.81
Bi-RNN+WE	72.17	64.40	68.06
Bi-GRU+WE	77.06	70.55	73.66
Bi-LSTM+WE	77.32	73.75	75.49

Table 5: Performance of different models with *50 dim embedded vectors* in **Task A** *validation set* **when word vectors are not getting updated while training**

Comparison with State-of-art

At the end we are comparing our results with state-of-the art results reported in (Doğan and Lu, 2012) on this dataset using BANNER (Leaman and Gonzalez, 2008) in table 7. BANNER is a CRF based bio entity recognition model, which uses general linguistic, orthographic, syntactic dependency features. Although the result reported in (Doğan and Lu, 2012) (F1-score = 81.8) is better than that of our RNN models but it should be noted that competitive result (F1-score = 79.13%) is obtained by the proposed Bi-LSTM model which does not depend on any feature engineering or domain-specific resources and is using only word embedding features trained in unsupervised manner on a huge corpus.

Model	*P*	*R*	*F*
NN+RV	81.64	74.01	77.64
Bi-RNN+RV	82.32	72.73	77.2
Bi-GRU+RV	82.48	74.14	78.08
Bi-LSTM+RV	83.41	72.73	77.70

Table 6: Results of different models with *50 dim random vectors* in **Task A** *validation set*

For the task B, we did not find any paper except (Li, 2012). Li (2012) used linear soft margin support vector (SVM) machine with a number of hand designed features including dictionary based features. The best performing proposed model shows more than 37% improvement in F1-score (benchmark: 46% vs Bi-LSTM+WE: 63.16%).

5 Failure Analysis

To see where exactly our models failed to recognize diseases, we analyzed the results carefully.

Task	Model	Validation Set			Test Set		
		P	R	F	P	R	F
A	Bi-LSTM+WE	85.13	77.72	81.26	84.87	74.11	79.13
	BANNER (Doğan and Lu, 2012)	-	-	**81.9**	-	-	**81.8**
B	Bi-LSTM+WE	67.48	58.01	**62.39**	68.97	58.25	**63.16**
	SM-SVM(Li, 2012)	-	-	-	66.1	35.2	46.0

Table 7: Comparisons of our best model results and state-of-art results. SM-SVM :Soft Margin Support Vector Machine

We found that significant proportion of errors are coming due to use of acronyms of diseases and use of disease form which is rarely appearing in our corpus. Examples of few such cases are *"CD", "HNPCC","SCA1"*. We observe that this error is occurring because we do not have exact word embedding for these words. Most of the acronyms in the disease corpus were mapped to rare-word embedding[1]. Another major proportion of errors in our results were due to difficulty in recognizing nested forms of disease names. For example, in all of the following cases: *"hereditary forms of 'ovarian cancer'" , "inherited 'breast cancer'", "male and female 'breast cancer'"*, part of phrase such as *ovarian cancer* in *hereditary forms of ovarian cancer*, *breast cancer* in *inherited breast cancer* and *male and female breast cancer* are disease names and our models are detecting this very well. However, according to annotation scheme if any disease is part of nested disease name, annotators considered whole phrase as a single disease. So even our model is able to detect part of the disease accurately but due to the exact matching scheme, this will be false positive for us.

6 Related Research

In biomedical domain, named entity recognition has attracted much attention for identification of entities such as genes and proteins (Settles, 2005; Leaman and Gonzalez, 2008; Leaman et al., 2009) but not as much for disease name recognition. Notable works, such as of Chowdhury and Lavelli (2010), are mainly conditional random field (CRF) based models using lots of manually designed template features. These include linguistic, orthographic, contextual and dictionary based features. However, they have evaluated their model on the AZDC dataset which is small compared to the NCBI dataset, which we have considered in this study. Nikfarjam et al. (2015) have proposed a CRF based sequence tagging model, where cluster id of embedded word as an extra feature with manually engineered features is used for adverse drug reaction recognition in tweets.

Recently deep neural network models with minimal dependency on feature engineering have been used in few studies in NLP including NER tasks (Collobert et al., 2011; Collobert and Weston, 2008). dos Santos et al. (2015) used deep neural network based model such as window based network to recognize named entity in Portuguese and Spanish texts. In this work, they exploit the power of CNN to get morphological and shape features of words in character level word embedding, and used it as feature with concatenation of word embedding. Their results indicate that CNN are able to preserve morphological and shape features through character level word embedding. Our models are quite similar to this model but we used different variety of RNN in place of window based neural network.

Labeau et al. (2015) used Bi-RNN with character level word embedding only as a feature for PoS tagging in German text. Their results also show that with only character level word embedding we can get state-of-art results in PoS tagging in German text. Our model used word embedding as well as character level word embedding together as features and also we have tried more sophisticated RNN models such as LSTM and GRU in bi-directional structure. More recent work of Huang et al. (2015) used LSTM and CRF in variety of combination such as only LSTM, LSTM with CRF and Bi-LSTM with CRF for PoS tagging, chunking and NER tasks in general texts. Their results shows that Bi-LSTM with CRF gave best results in all these tasks. These two works have used either Bi-RNN with character embedding features or Bi-LSTM with word embedding

[1]we obtained pre-trained word-embedding features from (TH et al., 2015) and in their pre-processing strategy, all words of frequency less than 50 were mapped to rare-word.

features in general or news wire texts, whereas in this work we compare the performance of three different types of RNNs: Bi-RNN, Bi-GRU and Bi-LSTM with both word embedding and character embedding features in biomedical text for disease name recognition.

7 Conclusions

In this work, we used three different variants of bidirectional RNN models with word embedding features for the first time for disease name and class recognition tasks. Bidirectional RNN models are used to capture both forward and backward long term dependencies among words within a sentence. We have shown that these models are able to obtain quite competitive results compared to the benchmark result on the disease name recognition task. Further our results have shown a significantly improved results on the relatively harder task of disease classification which has not been studied much. All our results were obtained without putting any effort on feature engineering or requiring domain-specific knowledge. Our results also indicate that RNN based models perform better than window based neural network model for the two tasks. This could be due to the implicit ability of RNN models to capture variable range dependencies of words compared to explicit dependency on context window size of window based neural network models.

Acknowledgments

We acknowledge the use of computing resources made available from the Board of Research in Nuclear Science (BRNS), Dept of Atomic Energy (DAE) Govt. of India sponsered project (No.2013/13/8-BRNS/10026) by Dr Aryabartta Sahu at Department of Computer Science and Engineering, IIT Guwahati.

References

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March.

Yoshua Bengio, Nicolas Boulanger-Lewandowski, and Razvan Pascanu. 2013. Advances in optimizing recurrent networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8624–8628. IEEE.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November.

Rezarta Islamaj Dogan, Robert Leaman, and Zhiyong Lu. 2014. NCBI disease corpus: A resource for disease name recognition and concept normalization. *Journal of Biomedical Informatics*, 47:1–10.

Cicero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *International Conference on Machine Learning (ICML)*, volume 32, pages 1818–1826. JMLR W&CP.

Cicero dos Santos, Victor Guimaraes, RJ Niterói, and Rio de Janeiro. 2015. Boosting named entity recognition with neural character embeddings. *Proceedings of NEWS 2015 The Fifth Named Entities Workshop*, page 9.

Rezarta Islamaj Doğan and Zhiyong Lu. 2012. An improved corpus of disease mentions in pubmed citations. In *Proceedings of the 2012 Workshop on Biomedical Natural Language Processing*, BioNLP '12, pages 91–99, Stroudsburg, PA, USA. Association for Computational Linguistics.

John Duchi, Elad Hazan, and Yoram Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. 2009. A novel connectionist system for unconstrained handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(5):855–868, May.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

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

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991.

Matthieu Labeau, Kevin Lser, and Alexandre Allauzen. 2015. Non-lexical neural architecture for fine-grained pos tagging. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 232–237. The Association for Computational Linguistics.

Robert Leaman and Graciela Gonzalez. 2008. Banner: An executable survey of advances in biomedical named entity recognition. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany Murray, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing*, pages 652–663. World Scientific.

Robert Leaman, Christopher Miller, and G Gonzalez. 2009. Enabling recognition of diseases in biomedical text with machine learning: corpus and benchmark. *Proceedings of the 2009 Symposium on Languages in Biology and Medicine*, 82(9).

Gang Li. 2012. Disease mention recognition using soft-margin svm. *Training*, 593:5–148.

Md. Faisal Mahbub Chowdhury and Alberto Lavelli. 2010. Disease mention recognition with specific features. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, BioNLP '10, pages 83–90, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

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

Azadeh Nikfarjam, Abeed Sarker, Karen OConnor, Rachel Ginn, and Graciela Gonzalez. 2015. Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *Journal of the American Medical Informatics Association*, page ocu041.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2012. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063.

Barbara Rosario and Marti A. Hearst. 2004. Classifying semantic relations in bioscience texts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

B. Settles. 2005. ABNER: An open source tool for automatically tagging genes, proteins, and other entity names in text. *Bioinformatics*, 21(14):3191–3192.

MUNEEB TH, Sunil Sahu, and Ashish Anand. 2015. Evaluating distributed word representations for capturing semantics of biomedical concepts. In *Proceedings of BioNLP 15*, pages 158–163, Beijing, China, July. Association for Computational Linguistics.

Domain Adaptation for Authorship Attribution: Improved Structural Correspondence Learning

Upendra Sapkota
University of Alabama at Birmingham
upendra@uab.edu

Thamar Solorio
University of Houston
solorio@cs.uh.edu

Manuel Montes-y-Gómez
Instituto Nacional de
Astrofísica, Optica y Electrónica
mmontesg@inaoep.mx

Steven Bethard
University of Alabama at Birmingham
bethard@uab.edu

Abstract

We present the first domain adaptation model for authorship attribution to leverage unlabeled data. The model includes extensions to structural correspondence learning needed to make it appropriate for the task. For example, we propose a median-based classification instead of the standard binary classification used in previous work. Our results show that punctuation-based character n-grams form excellent pivot features. We also show how singular value decomposition plays a critical role in achieving domain adaptation, and that replacing (instead of concatenating) non-pivot features with correspondence features yields better performance.

1 Introduction

Authorship Attribution (AA) can be used for historical purposes, such as disentangling the different authors contributing to a literary work. It can also help in understanding language evolution and change at the individual level, revealing a writer's changes in linguistic patterns over time (Hirst and Feng, 2012). Authorship attribution can also help to settle disputes over the original creators of a given piece of text. Or it can help build a prosecution case against an online abuser, an important application especially considering the rising trends in cyber-bullying and other electronic forms of teen violence[1]. The absorbing social media networks, together with the ever increasing use of electronic communications will require robust approaches to authorship attribution that can help to determine with certainty the author of a text, determine the provenance of a written sample, and in sum, help us determine the trustworthiness of electronic data.

[1] http://cyberbullying.org/

One of the scenarios that has received limited attention is cross-domain authorship attribution, when we need to identify the author of a text but all the text with known authors is from a different topic, genre, or modality. Here we propose to solve the problem of cross-domain authorship attribution by adapting the Structural Correspondence Learning (SCL) algorithm proposed by Blitzer et al. (2006). We make the following contributions:

- We introduce the first domain adaptation model for authorship attribution that combines labeled data in a source domain with unlabeled data from a target domain to improve performance on the target domain.
- We examine two sets of features that have previously been successful in cross-domain authorship attribution, explain how these can be used to select the "pivot" features required by SCL, and show that typed n-gram features (which differentiate between the *the* in ***the**ir* and the *the* in *brea**the***) produce simpler models that are just as accurate.
- We propose a new approach for defining SCL's pivot feature classification task so that it is able to handle count-based features, and show that this median-based approach outperforms the standard SCL approach.
- We examine the importance of the dimensionality reduction step in SCL, and show that the singular value decomposition increases robustness even beyond the robustness achieved by SCL's learned feature transformations.
- We propose an alternative approach to combining features within SCL, and show that excluding the non-pivot features from the final classifier generally improves performance.

Our experimental results show that using standard SCL for this domain adaptation authorship attribution task improves prediction accuracy by

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2226–2235,
Berlin, Germany, August 7-12, 2016.

only 1% over a model without any domain adaptation. In contrast, our proposed improvements to SCL reach an accuracy boost of more than 15% over the no domain adaptation model and of 14% over the standard SCL formulation. The extensions to SCL that we propose in this work are likely to yield performance improvements in other tasks where SCL has been successfully applied, such as part-of-speech tagging and sentiment analysis. We plan to investigate this further in the future.

2 Related Work

Cross-Domain Authorship Attribution Almost all previous authorship attribution studies have tackled traditional (single-domain) authorship problems where the distribution of the test data is the same as that of the training data (Madigan et al., 2005; Stamatatos, 2006; Luyckx and Daelemans, 2008; Escalante et al., 2011). However, there are a handful of authorship attribution studies that explore cross-domain authorship attribution scenarios (Mikros and Argiri, 2007; Goldstein-Stewart et al., 2009; Schein et al., 2010; Stamatatos, 2013; Sapkota et al., 2014). Here, following prior work, cross-domain is a cover term for cross-topic, cross-genre, cross-modality, etc., though most work focuses on the cross-topic scenario.

Mikros and Argiri (2007) illustrated that many stylometric variables are actually discriminating topic rather than author. Therefore, the authors suggest their use in authorship attribution should be done with care. However, the study did not attempt to construct authorship attribution models where the source and target domains differ.

Goldstein-Stewart et al. (2009) performed a study on cross-topic authorship attribution by concatenating the texts of an author from different genres on the same topics. Such concatenation allows some cross-topic analysis, but as each test document contains a mix of genres it is not representative of real world authorship attribution problems.

Stamatatos (2013) and Sapkota et al. (2014) explored a wide variety of features, including lexical, stopword, stylistic, and character n-gram, and demonstrated that character n-grams are the most effective features in cross-topic authorship attribution. Stamatatos (2013) concluded that avoiding rare features is effective in both intra-topic and cross-topic authorship attribution by training a SVM classifier on one fixed topic and testing on each of the remaining topics. Sapkota et al. (2014), rather than fixing a single training topic in advance, considered all possible training/testing topic combinations to investigate cross-topic authorship attribution. This showed that training on documents from multiple topics (thematic areas) improves performance in cross-topic authorship attribution (Sapkota et al., 2014), even when controlling the amount of training data.

However, none of these studies exploited domain adaptation methods that combine labeled data in a source domain with unlabeled data from a target domain to improve performance on the target domain. Instead, they focused on identifying relevant features and simply evaluating them when trained on source-domain data and tested on target-domain data. To our knowledge, we are the first to leverage unlabeled data from the target domain to improve authorship attribution.

Domain Adaptation Domain adaptation is the problem of modifying a model trained on data from a source domain to a different, possibly related, target domain. Given the effort and the cost involved in labeling data for a new target domain, there is a lot of interest in the design of domain adaptation techniques. In NLP related tasks, researchers have explored domain adaptation for part-of-speech tagging, parsing, semantic role labeling, word-sense disambiguation, and sentiment analysis (Li, 2012).

Daumé (2007) proposed a feature space transformation method for domain adaptation based on a simple idea of feature augmentation. The basic idea is to create three versions of each feature from the original problem: the general (domain-independent) version, the source specific version, and the target specific version. While generally successful, there are some limitations of this method. First, it requires labeled instances in the target domain. Second, since this method simply duplicates each feature in the source domain as domain-independent and domain-specific versions, it is unable to extract the potential correlations when the features in the two domains are different, but have some hidden correspondences.

In contrast, structural correspondence learning (SCL) is a feature space transformation method that requires no labeled instances from the target domain, and can capture the hidden correlations among different domain-independent features. SCL's basic idea is to use unlabeled data from both the source and target domains to obtain a common feature representation that is meaningful across

domains (Blitzer et al., 2006). Although the distributions of source and target domain differ, the assumption is that there will still be some general features that share similar characteristics in both domains. SCL has been applied to tasks such as sentiment analysis, dependency parsing, and part-of-speech tagging, but has not yet been explored for the problem of authorship attribution.

The common feature representation in SCL is created by learning a projection to "pivot" features from all other features. These pivot features are a critical component of the successful use of SCL, and their selection is something that has to be done carefully and specifically to the task at the hand. Tan and Cheng (2009) studied sentiment analysis, using frequently occurring sentiment words as pivot features. Similarly, Zhang et al. (2010) proposed a simple and efficient method for selecting pivot features in domain adaptive sentiment analysis: choose the frequently occurring words or word-bigrams among domains computed after applying some selection criterion. In dependency parsing, Shimizu and Nakagawa (2007) chose the presence of a preposition, a determiner, or a helping verb between two tokens as the pivot features. For part-of-speech tagging, Blitzer et al. (2006) used words that occur more than 50 times in both domains as the pivot features, resulting in mostly function words. In cross-lingual adaptation using SCL, semantically related pairs of words from source and target domains were used as pivot features (Prettenhofer and Stein, 2011). For authorship attribution, we propose two ways of selecting pivot and non-pivot features based on character n-grams.

Another important aspect of the SCL algorithm is associating a binary classification problem with each pivot feature. The original SCL algorithm assumes that pivot features are binary-valued, so creating a binary classification problem for each pivot feature is trivial: is the value 0 or 1? Most previous work on part-of-speech tagging, sentiment analysis, and dependency parsing also had only binary-valued pivot features. However, for authorship attribution, all features are count-based, so translation from a pivot feature value to a binary classification problem is not trivial. We propose a median-based solution to this problem.

3 Methodology

Structural Correspondence Learning (Blitzer et al., 2006) uses only unlabeled data to find a common feature representation for a source and a target domain. The idea is to first manually identify "pivot" features that are likely to have similar behavior across both domains. SCL then learns a transformation from the remaining non-pivot features into the pivot feature space. The result is a new set of features that are derived from all the non-pivot features, but should be domain independent like the pivot features. A classifier is then trained on the combination of the original and the new features.

Table 1 gives the details of the SCL algorithm. First, for each pivot feature, we train a linear classifier to predict the value of that pivot feature using only the non-pivot features. The weight vectors learned for these linear classifiers, $\hat{w}_i$, are then concatenated into a matrix, W, which represents a projection from non-pivot features to pivot features. Singular value decomposition is used to reduce the dimensionality of the projection matrix, yielding a reduced-dimensionality projection matrix θ. Finally, a classifier is trained on the combination of the original features and the features generated by applying the reduced-dimensionality projection matrix θ to the non-pivot features $\mathbf{x}_{[p:m]}$.

3.1 Standard SCL parameter definitions

Standard SCL does not define how pivot features are selected; this must be done manually for each new task. However, SCL does provide standard definitions for the loss function (L), the conversion to binary values (B_i), the dimensionality of the new correspondence space (d), and the feature combination function (C).

L is defined as Huber's robust loss:

$$L(a, b) = \begin{cases} \max(0, 1 - ab)^2 & \text{if } ab \geq -1 \\ -4ab & \text{otherwise} \end{cases}$$

The conversion from pivot feature values to binary classification is defined as:

$$B_i(y) = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases}$$

A few different dimensionalities for the reduced feature space have been explored (Prettenhofer and Stein, 2011), but most implementations have followed the standard SCL description (Blitzer et al., 2006) with d defined as:

$$d = 25$$

The feature combination function, C, is defined as simple concatenation, i.e., use all of the old pivot

Input:

- $S = \{\mathbf{x} : \mathbf{x} \in \mathbb{R}^m\}$, the labeled instances from source domain
- $U = \{\mathbf{x} : \mathbf{x} \in \mathbb{R}^m\}$, the unlabeled instances from both domains
- p and n such that $\mathbf{x}_{[0:p]}$ are the p pivot features and $\mathbf{x}_{[p:m]}$ are $n = m - p$ non-pivot features
- $f : S \rightarrow A$, the source domain labels, where A is the set of authors
- $L : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$, a loss function
- $B_i : \mathbb{R} \rightarrow \{0, 1\}$ for $0 \leq i < p$, a conversion from a real-valued pivot feature i to binary classification
- d, the size of the reduced-dimensionality correspondence space to learn
- $C : \mathbb{R}^m \times \mathbb{R}^d \rightarrow \mathbb{R}^k$, a function for combining the original and new features

Output:

- $\theta \in \mathbb{R}^{n \times d}$, a projection from non-pivot features to the correspondence space
- $h : \mathbb{R}^{m+d} \rightarrow A$, the trained predictor

Algorithm:

1. For each pivot feature $i : 0 \leq i < p$, learn prediction weights $\hat{w}_i = \min_{\mathbf{w} \in \mathbb{R}^n} \sum_{\mathbf{x} \in U} L(\mathbf{w}^\top \mathbf{x}_{[p:m]}, B(x_i))$
2. Construct a matrix $W \in \mathbb{R}^{n \times p}$ using each $\hat{w}_i$ as a column
3. Apply singular value decomposition $W = U \Sigma V^\top$ where $U \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{n \times p}$, $V^\top \in \mathbb{R}^{p \times p}$
4. Select the reduced-dimensionality projection, $\theta = {U_{[0:d,:]}}^\top$
5. Train a classifier h from $\left\{ \left([C(\mathbf{x}, \mathbf{x}_{[p:m]}\theta), f(\mathbf{x}) \right) : \mathbf{x} \in S \right\}$

Table 1: The structural correspondence learning (SCL) algorithm

features, all the old non-pivot features, and all the new correspondence features:

$$C(\mathbf{x}, \mathbf{z}) = [\mathbf{x}; \mathbf{z}]$$

We call this the *pivot+nonpivot+new* setting of C.

The following sections discuss alternative parameter choices for pivot features, B_i, d, and C.

3.2 Pivot Features for Authorship Attribution

The SCL algorithm depends heavily on the pivot features being domain-independent features, and as discussed in Section 2, which features make sense as pivot features varies widely by task. No previous studies have explored structural correspondence learning for authorship attribution, so one of the outstanding questions we tackle here is how to identify pivot features. Research has shown that the most discriminative features in attribution and the most robust features across domains are character n-grams (Stamatatos, 2013; Sapkota et al., 2014). We thus consider two types of character n-grams used in authorship attribution that might make good pivot features.

3.2.1 Untyped Character N-grams

Classical character n-grams are simply the sequences of characters in the text. For example, given the text:

> The structural correspondence

character 3-gram features would look like:

```
"The", "he ", "e s", " st",
"str",  "tru", "ruc", "uct", ...
```

We propose to use as pivot features the p most frequent character n-grams. For non-pivot features, we use the remaining features from prior work (Sapkota et al., 2014). These include both the remaining (lower frequency) character n-grams, as well as stop-words and bag-of-words lexical features. We call this the *untyped* formulation of pivot features.

3.2.2 Typed Character N-grams

Sapkota et al. (2015) showed that classical character n-grams lose some information in merging together instances of n-grams like *the* which could be a prefix (***the**sis*), a suffix (*brea**the***), or a standalone word (***the***). Therefore, untyped character n-grams were separated into ten distinct categories. Four of the ten categories are related to affixes: prefix, suffix, space-prefix, and space-suffix. Three are word-related: whole-word, mid-word, and multi-word. The final three are related to the use of punctuation: beg-punct, mid-punct, and end-punct. For example, the character n-grams from the last section would instead be replaced with:

```
"whole-word:The", "space-suffix:he ",
"multi-word:e s", "space-prefix: st",
"prefix:str",  "mid-word:tru",
"mid-word:ruc", "mid-word:uct", ...
```

Sapkota et al. (2015) demonstrated that n-grams starting with a punctuation character (the *beg-punct* category) and with a punctuation character in the middle (the *mid-punct* category) were the most effective character n-grams for cross-domain authorship attribution. We therefore propose to use as pivot features the $p/2$ most frequent character n-grams from each of the *beg-punct* and *mid-punct* categories, yielding in total p pivot features. For non-pivot features, we use all of the remaining features of Sapkota et al. (2015). These include both the remaining (lower frequency) *beg-punct* and *mid-punct* character n-grams, as well as all of the character n-grams from the remaining eight categories. We call this the *typed* formulation of pivot features.[2]

3.3 Pivot feature binarization parameters

Authorship attribution typically relies on count-based features. However, the classic SCL algorithm assumes that all pivot features are binary, so that it can train binary classifiers to predict pivot feature values from non-pivot features. We propose a binarization function to produce a binary classification problem from a count-based pivot feature by testing whether the feature value is above or below the feature's median value in the training data:

$$B_i(y) = \begin{cases} 1 & \text{if } y > median(\{\mathbf{x}_i : \mathbf{x} \in S \cup U\}) \\ 0 & \text{otherwise} \end{cases}$$

The intuition is that for count-based features, "did this pivot feature appear at least once in the text" is not a very informative distinction, especially since the average document has hundreds of words, and pivot features are common. A more informative distinction is "was this pivot feature used more or less often than usual?" and that corresponds to the below-median vs. above-median classification.

3.4 Dimensionality reduction parameters

The reduced dimensionality (d) of the low-rank representation varies depending on the task at hand, though lower dimensionality may be preferred as it will result in faster run times. We empirically compare different choices for d: 25, 50, and 100.

We also consider the question, how critical is dimensionality reduction? For example, if there are only $p = 100$ pivot features, is there any need to run singular-value decomposition? The goal here is to determine if SCL is increasing the robustness across domains primarily through transforming non-pivot features into pivot-like features, or if the reduced dimensionality from the singular-value decomposition contributes something beyond that.

[2]Because the **untyped** and **typed** feature sets are designed to directly replicate Sapkota et al. (2014) and Sapkota et al. (2015), respectively, both include character n-grams, but only **untyped** includes stop-words and lexical features.

Topics	4
Authors	13
Documents/author/topic	10
Average sentences/document	53
Average words/document	1034

Table 2: Statistics of the Guardian dataset.

3.5 Feature combination parameters

It's not really clear why the standard formulation of SCL uses the non-pivot features when training the final classifier. All of the non-pivot features are projected into the pivot feature space in the form of the new correspondence features, and the pivot feature space is, by design, the most domain independent part of the feature space. Thus, it seems reasonable to completely replace the non-pivot features with the new pivot-like features. We therefore consider a *pivot+new* setting of C:

$$\textit{pivot+new: } C(\mathbf{x}, \mathbf{z}) = [\mathbf{x}_{[0:p]}; \mathbf{z}]$$

We also consider other settings of C, primarily for understanding how the different pieces of the SCL feature space contribute to the overall model.

$$\begin{aligned} \textit{pivot: } C(\mathbf{x}, \mathbf{z}) &= \mathbf{x}_{[0:p]} \\ \textit{nonpivot: } C(\mathbf{x}, \mathbf{z}) &= \mathbf{x}_{[p:m]} \\ \textit{new: } C(\mathbf{x}, \mathbf{z}) &= \mathbf{z} \\ \textit{pivot+nonpivot: } C(\mathbf{x}, \mathbf{z}) &= \mathbf{x} \end{aligned}$$

Note that the *pivot+nonpivot* setting corresponds to a model that does not apply SCL at all.

4 Dataset

To explore cross-domain settings of authorship attribution, we need datasets containing documents from a number of authors from different domains (different topics, different genres). We use a corpus that consists of texts published in The Guardian daily newspaper that is actively used by the authorship attribution community in cross-domain studies (Stamatatos, 2013; Sapkota et al., 2014;

Sapkota et al., 2015). The Guardian corpus contains opinion articles written by 13 authors in four different topics: World, U.K., Society, and Politics. Following prior work, to make the collection balanced across authors, we choose at most ten documents per author for each of the four topics. Table 2 presents some statistics about the datasets.

5 Experimental Settings

We trained support vector machine (SVM) classifiers using the Weka implementation (Witten and Frank, 2005) with default parameters. For the *untyped* features, we used character 3-grams appearing at least 5 times in the training data, a list of 643 predefined stop-words, and the 3,500 most frequent non-stopword words as the lexical features. For the *typed* features, we used the top 3,500 most frequent 3-grams occurring at least five times in the training data for each of the 10 character n-gram categories. In both cases, we selected $p = 100$ pivot features as described in Section 3.2.

We measured performance in terms of accuracy across all possible topic pairings. That is, we paired each of the 4 topics in the Guardian corpus with each of the 3 remaining topics: train on Politics, test on Society; train on Politics, test on UK; train on Politics, test on World; etc. For each such model, we allowed SCL to learn feature correspondences from the labeled data of the 1 training topic and the unlabeled data of the 1 test topic. This resulted in 12 pairings of training/testing topics. We report both accuracy on the individual pairings and an overall average of the 12 accuracies.

We compare performance against two state-of-the-art baselines: Sapkota et al. (2014) and Sapkota et al. (2015), as described in Section 3.2, and whose features are denoted as **untyped** and **typed**, respectively. We replicate these models by using the *pivot+nonpivot* setting of C, i.e., not including any of the new SCL-based features.

6 Results

The following sections explore the results of our innovations in different areas: pivot features, feature binarizations, dimensionality reduction, and feature combination. For each section, we hold the other parameters constant and vary only the one parameter of interest. Thus, where not otherwise specified, we set parameters to the best values we observed in our experiments: we set the feature set to *typed*, the binarization $B_i(y)$ to the median, the reduced dimensionality d to 50, and the feature combination $C(\mathbf{x}, \mathbf{z})$ to *pivot+new* (i.e., we use the old pivot features alongside the new correspondence features). All reports of statistical significance are based on paired, two-tailed t-tests over the 12 different topic pairings.

Dataset	**untyped**	**typed**
Politics-Society	61.29	**67.74**
Politics-UK	**66.67**	63.33
Politics-World	58.97	**64.10**
Society-Politics	**62.96**	**62.96**
Society-UK	**72.50**	**72.50**
Society-World	**56.62**	48.08
UK-Politics	**68.75**	60.71
UK-Society	66.13	**67.74**
UK-World	57.27	**58.97**
World-Politics	**62.50**	59.82
World-Society	61.29	**62.90**
World-UK	46.67	**54.44**
Average	61.80	**61.94**

Table 3: Accuracy of **untyped** and **typed** feature sets. The difference between the averages is not statistically significant (p=0.927).

6.1 Untyped vs. Typed features

Table 3 compares the **untyped** feature set to the **typed** feature set. Both feature sets perform reasonably well, and substantially better than a model without SCL, where the performance of **untyped** is 56.43 and **typed** is 53.62 (see the *pivot+nonpivot* columns of Table 6 and Table 7, discussed in Section 6.4). Recall that the **typed** formulation includes only character n-gram features, while the **untyped** formulation includes stopwords and lexical features as well. Thus, given their very similar performance in Table 3, **typed** being slightly better, we select the simpler **typed** feature formulation for the remaining experiments.

6.2 Greater-than-zero vs. Median Binarization

Table 4 compares choices for $B_i(y)$, the function for converting a pivot feature value into a binary classification problem. In every single train/test scenario, and for both **untyped** and **typed** feature sets, our proposed median-based binarization function yielded performance greater than or equal to that of the traditional SCL greater-than-zero binarization function. This confirms our hypothesis that count-based features were inadequately modeled

Dataset	**untyped**		**typed**	
	>0	>med	>0	>med
Politics-Society	58.06	**61.29**	61.29	**67.74**
Politics-UK	**66.67**	**66.67**	**63.33**	**63.33**
Politics-World	55.56	**58.97**	63.81	**64.10**
Society-Politics	61.81	**62.96**	62.67	**62.96**
Society-UK	72.50	**72.50**	71.00	**72.50**
Society-World	51.92	**56.62**	46.00	**48.08**
UK-Politics	59.82	**68.75**	60.00	**60.71**
UK-Society	59.68	**66.13**	64.52	**67.74**
UK-World	47.86	**57.27**	57.27	**58.97**
World-Politics	56.25	**62.50**	56.50	**59.82**
World-Society	50.00	**61.29**	61.52	**62.90**
World-UK	42.22	**46.67**	50.00	**54.44**
Average	56.11	**61.80**	59.83	**61.94**

Table 4: Accuracy of greater-than-zero and median formulations of the $B_i(y)$ binarization function. Median is significantly better than greater-than-zero in both **untyped** (p=0.0007) and **typed** (p=0.003).

Dataset	d=25	d=50	d=100	no SVD
Politics-Society	66.13	67.74	**72.58**	50.00
Politics-UK	62.22	63.33	**66.67**	48.89
Politics-World	63.25	**64.10**	**64.10**	47.01
Society-Politics	**64.81**	62.96	55.56	57.41
Society-UK	67.50	**72.5**	67.5	70.00
Society-World	**48.08**	**48.08**	44.23	46.15
UK-Politics	**60.71**	**60.71**	58.93	51.79
UK-Society	64.52	**67.74**	56.45	59.68
UK-World	**60.68**	58.97	58.12	49.57
World-Politics	**62.50**	59.82	51.79	55.36
World-Society	59.68	62.90	**67.74**	62.90
World-UK	54.44	54.44	**55.56**	51.11
Average	61.21	**61.94**	59.94	54.16

Table 5: Accuracy of different choices for dimensionality reduction with **typed** features. The pattern is similar for **untyped**. $d = 50$ is significantly better than no SVD (p=0.0009), but not significantly different from $d = 25$ (p=0.291) or $d = 100$ (p=0.211).

in standard SCL and that the median-based binarization function improves the modeling of such features.

6.3 Dimensionality Reduction Choices

Table 5 compares different choices for the dimensionality reduction parameter d, as well as the possibility of not performing any dimensionality reduction at all ("No-SVD"). While each value of d yields the best performance on some of the train/test scenarios, $d = 50$ achieves the highest average accuracy (61.94). Removing the SVD entirely generally performs worse, and though on a small number of train/test scenarios it outperforms $d = 25$ and $d = 100$, it is always worse than $d = 50$.

This shows that SCL's feature correspondences alone are not sufficient to achieve domain adaptation. Without the SVD, performance is barely above a model without SCL: 54.16 vs. 53.62 (see Section 6.4). Much of the benefit appears to be coming from the SVD's basis-shift, since $d = 100$ outperforms no-SVD by more than 5 points[3], while $d = 50$ only outperforms $d = 100$ by 2 points. These results are consistent with SCL's origins in alternating structural optimization (Ando and Zhang, 2005), where SVD is derived as a necessary step for identifying a shared low-dimensional subspace.

6.4 Replacing vs. Concatenating Features

Table 6 and Table 7 compare the performance of different choices for the feature combination function $C(\mathbf{x}, \mathbf{z})$ on **untyped** and **typed** features, respectively. Our proposed *pivot+new* combination function, which replaces the non-pivot features with the new correspondence features, performs better on average than the two state-of-the-art baselines with no domain adaptation (*pivot+nonpivot*) and than the two state-of-the-art baselines augmented with classic SCL (*pivot+nonpivot+new*): 61.80 vs. 56.43 and 56.93 for untyped, and 61.94 vs. 53.62 and 54.23 for typed). These 5-8 point performance gains confirm the utility of our proposed *pivot+new* combination function, which replaces the old non-pivot features with the new correspondence features. These gains are consistent with (Blitzer et al., 2006), who included both pivot and non-pivot features, but found that they had to give pivot features a weight "five times that of the [non-pivot] features" to see improved performance.

While our approach is better on average, in some individual scenarios, it performs worse than classic SCL or no domain adaptation. For example, on Politics-Society, Politics-UK, and World-UK, using **typed** features, *pivot+new* performs worse than no domain adaptation (*pivot+nonpivot*). Our results suggest a rule for predicting when this degradation will happen: *pivot+new* will outperform

[3]Recall that $p = 100$, so $d = 100$ means the full matrix.

Dataset	*pivot*	*nonpivot*	*new*	*pivot+nonpivot*	*pivot+nonpivot+new*	*pivot+new*
Politics-Society	54.84	75.81	62.9	75.81	**77.42**	61.29
Politics-UK	63.33	68.89	58.89	70.00	**71.11**	66.67
Politics-World	58.12	63.25	53.85	64.96	**65.41**	58.97
Society-Politics	61.11	46.30	48.15	46.30	46.30	**62.96**
Society-UK	67.5	45.00	60.00	47.50	47.50	**72.50**
Society-World	50.00	42.31	53.85	46.15	46.15	**56.62**
UK-Politics	62.50	42.86	59.82	42.86	44.64	**68.75**
UK-Society	59.68	43.55	55.83	45.16	45.16	**66.13**
UK-World	45.30	38.46	48.72	39.32	39.32	**57.27**
World-Politics	55.36	69.64	56.25	68.75	**69.64**	62.5
World-Society	46.77	67.74	53.23	**69.35**	**69.35**	61.29
World-UK	43.33	**61.11**	50.00	**61.11**	**61.11**	46.67
Average	55.65	55.41	55.12	56.43	56.93	**61.80**

Table 6: Accuracy of different **untyped** feature combinations. The best performance for each dataset is in bold. The performance of *pivot+new* is not significantly different from *pivot+nonpivot* (p=0.258) or *pivot+nonpivot+new* (p=0.305).

Dataset	*pivot*	*nonpivot*	*new*	*pivot+nonpivot*	*pivot+nonpivot+new*	*pivot+new*
Politics-Society	48.39	70.97	59.68	**72.58**	**72.58**	67.74
Politics-UK	52.22	68.89	66.67	71.11	**72.22**	63.33
Politics-World	46.15	61.54	61.54	63.25	**64.10**	**64.10**
Society-Politics	55.56	48.15	61.11	48.15	50.00	**62.96**
Society-UK	65.00	45.00	65.00	45.00	45.00	**72.50**
Society-World	38.46	46.15	**53.85**	44.23	46.15	48.08
UK-Politics	48.21	44.64	55.36	45.54	45.54	**60.71**
UK-Society	51.61	41.94	66.13	41.94	41.94	**67.74**
UK-World	44.44	33.33	45.30	35.90	35.90	**58.97**
World-Politics	50.89	51.79	**61.39**	57.14	57.14	59.82
World-Society	54.84	59.68	43.55	59.68	61.29	**62.9**
World-UK	44.44	56.67	50.00	**58.89**	**58.89**	54.44
Average	50.02	52.40	57.47	53.62	54.23	**61.94**

Table 7: Accuracy of different **typed** feature combinations. The best performance for each dataset is in bold. The performance of *pivot+new* is significantly better than *pivot+nonpivot* (p=0.041) but not significantly different from *pivot+nonpivot+new* (p=0.059).

both *pivot+nonpivot* and *pivot+nonpivot+new* iff the *new* features alone outperform the *nonpivot* features alone. This rule holds in all 12 of 12 train/test scenarios for **untyped** features and 11 of 12 scenarios for **typed** features (failing on only World-Society). Intuitively, if the *new* correspondence features that result from SCL aren't better than the features they were meant to replace, then it is unlikely that they will result in performance gains. This might happen if the pivot features are not strong enough predictors, either because they have been selected poorly or because there are too few of them.

7 Discussion

To the best of our knowledge, we are the first to introduce a domain adaption model for authorship attribution that combines labeled data in a source domain with unlabeled data from a target domain to improve performance on the target domain. We proposed several extensions to the popular structural correspondence learning (SCL) algorithm for domain adaptation to make it more amenable to tasks like authorship attribution. The SCL algorithm requires the manual identification of domain independent *pivot* features for each task, so we proposed two feature formulations using charac-

ter n-grams as the pivot features, and showed that both yielded state-of-the-art performance. We also showed that for the binary classification task that is used by SCL to learn the feature correspondences, replacing the traditional greater-than-zero classification task with a median-based classification task allowed the model to better handle our count-based features. We explored the dimensionality reduction step of SCL and showed that singular value decomposition (SVD) over the feature correspondence matrix is critical to achieving high performance. Finally, we introduced a new approach to combining the original features with the learned correspondence features, and showed that replacing (rather than concatenating) the non-pivot features with the correspondence features generally yields better performance.

In the future, we would like to extend this work in several ways. First, though our median-based approach was successful in converting pivot feature values to binary classification problems, learning a regression model might be an even better approach for count-based features. Second, since the SVD basis-shift seems to be the source of much of the gains, we would like to explore replacing the SVD with other algorithms, such as independent component analysis. Finally, we would like to explore further our finding that the performance of the overall model seems to be predicted by the difference in performance between the non-pivot features and the new correspondence features, especially to see if this can be predicted at training time rather than as a post-hoc analysis.

8 Acknowledgments

This work was supported in part by CONACYT Project number 247870, and by the National Science Foundation award number 1462141.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, December.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP 2006, pages 120–128, Stroudsburg, PA, USA.

Hal Daumé. 2007. Frustratingly easy domain adaptation. In *Annual meeting-association for computational linguistics*, volume 45, pages 256–256, 2007.

H. J. Escalante, T. Solorio, and M. Montes-y Gomez. 2011. Local histograms of character n-grams for authorship attribution. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 288–298, Portland, Oregon, USA, June. Association for Computational Linguistics.

Jade Goldstein-Stewart, Ransom Winder, and Roberta Evans Sabin. 2009. Person identification from text and speech genre samples. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 336–344, Stroudsburg, PA, USA. Association for Computational Linguistics.

Graeme Hirst and Vanessa Wei Feng. 2012. Changes in style in authors with alzheimer's disease. *English Studies*, 93(3):357–370.

Qi Li. 2012. Literature survey: Domain adaptation algorithms for natural language processing. Technical report, February.

Kim Luyckx and Walter Daelemans. 2008. Authorship attribution and verification with many authors and limited data. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, pages 513–520, Manchester, UK, August.

D. Madigan, A. Genkin, S. Argamon, D. Fradkin, and L. Ye. 2005. Author identification on the large scale. In *Proceedings of CSNA/Interface 05*.

George K. Mikros and Eleni K. Argiri. 2007. Investigating topic influence in authorship attribution. In *Proceedings of the International Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection*, pages 29–35.

Peter Prettenhofer and Benno Stein. 2011. Cross-lingual adaptation using structural correspondence learning. *ACM Trans. Intell. Syst. Technol.*, 3(1):13:1–13:22, October.

Upendra Sapkota, Thamar Solorio, Manuel Montes, Steven Bethard, and Paolo Rosso. 2014. Cross-topic authorship attribution: Will out-of-topic data help? In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1228–1237, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. 2015. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies, pages 93–102, Denver, Colorado, May–June. Association for Computational Linguistics.

Andrew I. Schein, Johnnie F. Caver, Randale J. Honaker, and Craig H. Martell. 2010. Author attribution evaluation with novel topic cross-validation. In *KDIR '10*, pages 206–215.

Nobuyuki Shimizu and Hiroshi Nakagawa. 2007. Structural correspondence learning for dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 1166–1169, Prague, Czech Republic, June. Association for Computational Linguistics.

E. Stamatatos. 2006. Authorship attribution based on feature set subspacing ensembles. *International Journal on Artificial Intelligence tools*, 15(5):823–838.

Efstathios Stamatatos. 2013. On the robustness of authorship attribution based on character n-gram features. *Journal of Law & Policy*, 21(2):421 – 439.

Songbo Tan and Xueqi Cheng. 2009. Improving SCL model for sentiment-transfer learning. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers*, pages 181–184, Boulder, Colorado, June. Association for Computational Linguistics.

I. H. Witten and E. Frank. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kauffmann, 2nd edition.

Yanbo Zhang, Youli Qu, and Junsan Zhang. 2010. A new method of selecting pivot features for structural correspondence learning in domain adaptive sentiment analysis. In *Database Technology and Applications (DBTA), 2010 2nd International Workshop on*, pages 1–3.

A Corpus-Based Analysis of Canonical Word Order of Japanese Double Object Constructions

Ryohei Sasano **Manabu Okumura**
Tokyo Institute of Technology
{sasano,oku}@pi.titech.ac.jp

Abstract

The canonical word order of Japanese double object constructions has attracted considerable attention among linguists and has been a topic of many studies. However, most of these studies require either manual analyses or measurements of human characteristics such as brain activities or reading times for each example. Thus, while these analyses are reliable for the examples they focus on, they cannot be generalized to other examples. On the other hand, the trend of actual usage can be collected automatically from a large corpus. Thus, in this paper, we assume that there is a relationship between the canonical word order and the proportion of each word order in a large corpus and present a corpus-based analysis of canonical word order of Japanese double object constructions.

1 Introduction

Japanese has a much freer word order than English. For example, a Japanese double object construction has six possible word orders as follows:

(1) a: *Ken-ga Aya-ni camera-wo miseta.*
Ken-NOM Aya-DAT camera-ACC showed

b: *Ken-ga camera-wo Aya-ni miseta.*
Ken-NOM camera-ACC Aya-DAT showed

c: *Aya-ni Ken-ga camera-wo miseta.*
Aya-DAT Ken-NOM camera-ACC showed

d: *Aya-ni camera-wo Ken-ga miseta.*
Aya-DAT camera-ACC Ken-NOM showed

e: *Camera-wo Ken-ga Aya-ni miseta.*
camera-ACC Ken-NOM Aya-DAT showed

f: *Camera-wo Aya-ni Ken-ga miseta.*
camera-ACC Aya-DAT Ken-NOM showed

In these examples, the position of the verb *miseta* (showed) is fixed but the positions of its nominative (NOM), dative (DAT), and accusative (ACC) arguments are scrambled. Note that, although the word orders are different, they have essentially the same meaning "Ken showed a camera to Aya."

In the field of linguistics, each language is assumed to have a basic word order that is fundamental to its sentence structure and in most cases there is a generally accepted theory on the word order for each structure. That is, even if there are several possible word orders for essentially same sentences consisting of the same elements, only one of them is regarded as the canonical word order and the others are considered to be generated by scrambling it. However, in the case of Japanese double object constructions, there are several claims on the canonical argument order.

There have been a number of studies on the canonical word order of Japanese double object constructions ranging from theoretical studies (Hoji, 1985; Miyagawa and Tsujioka, 2004) to empirical ones based on psychological experiments (Koizumi and Tamaoka, 2004; Nakamoto et al., 2006; Shigenaga, 2014) and brain science (Koso et al., 2004; Inubushi et al., 2009). However, most of them required either manual analyses or measurements of human characteristics such as brain activities or reading times for each example. Thus, while these analyses are reliable for the example they focus on, they cannot be easily generalized to other examples[1]. That is, another manual analysis or measurement is required to consider the canonical word order of another example.

On the other hand, the trend of actual usage can be collected from a large corpus. While it is difficult to say whether a word order is canonical or

[1]Note that in this work, we assume that there could be different canonical word orders for different double-object sentences as will be explained in Section 2.2.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2236–2244,
Berlin, Germany, August 7-12, 2016.

not from one specific example, we can consider that a word order would be canonical if it is overwhelmingly dominant in a large corpus. For example, since the DAT−ACC order[2] is overwhelmingly dominant in the case that the verb is *kanjiru* (feel), its dative argument is *kotoba* (word), and its accusative argument is *aijô* (affection) as shown in Example (2), we can consider that the DAT−ACC order would be canonical in this case. Note that, the numbers in parentheses represent the proportion of each word order in Examples (2) and (3); ϕ_X denotes the omitted noun or pronoun X in this paper.

(2) **DAT−ACC:** *Kotoba-ni aijô-wo kanjiru.*
(97.5%) word−DAT affection−ACC feel

ACC−DAT: *Aijô-wo kotoba-ni kanjiru.*
(2.5%) affection−ACC word−DAT feel

(ϕ_I feel the affection in ϕ_{your} words.)

On the contrary, since the ACC−DAT order is overwhelmingly dominant in the case that the verb is *sasou* (ask), its dative argument is *dêto* (date), and its accusative argument is *josei* (woman) as shown in Example (3), the ACC−DAT order is considered to be canonical in this case.

(3) **DAT−ACC:** *Dêto-ni josei-wo sasou.*
(0.4%) date−DAT woman−ACC ask

ACC−DAT: *Josei-wo dêto-ni sasou.*
(99.6%) woman−ACC date−DAT ask

(ϕ_I ask a woman out on a date.)

Therefore, in this paper, we assume that there is a relationship between the canonical word order and the proportion of each word order in a large corpus and attempt to evaluate several claims on the canonical word order of Japanese double object constructions on the basis of a large corpus. Since we extract examples of double object constructions only from reliable parts of parses of a very large corpus, consisting of more than 10 billion unique sentences, we can reliably leverage a large amount of examples. To the best of our knowledge, this is the first attempt to analyze the canonical word order of Japanese double object constructions on the basis of such a large corpus.

[2]Since Japanese word order is basically subject-object-verb (SOV) and thus the canonical position of nominative argument is considered to be the first position, we simply call the nominative, dative, accusative order as the DAT−ACC order, and the nominative, accusative, dative order as the ACC−DAT order in this paper.

2 Japanese double object constructions

2.1 Relevant Japanese grammar

We briefly describe the relevant Japanese grammar. Japanese word order is basically subject-object-verb (SOV) order, but the word order is often scrambled and does not mark syntactic relations. Instead, postpositional case particles function as case markers. For example, nominative, dative, and accusative cases are represented by case particles *ga*, *ni*, and *wo*, respectively.

In a double object construction, the subject, indirect object, and direct object are typically marked with the case particles *ga* (nominative), *ni* (dative), and *wo* (accusative), respectively, as shown in Example (4)-a.

(4) a: *Watashi-ga kare-ni camera-wo miseta.*
I−NOM him−DAT camera−ACC showed

b: *Watashi-wa kare-ni camera-wo miseta.*
I−TOP him−DAT camera−ACC showed

c: ϕ_I *kare-ni camera-wo miseta.*
ϕ_I-NOM him−DAT camera−ACC showed

(I showed him a camera.)

However, when an argument represents the topic of the sentence (TOP), the topic marker *wa* is used as a postpositional particle, and case particles *ga* and *wo* do not appear explicitly. For example, since *watashi* (I) in Example (4)-b represents the topic of the sentence, the nominative case particle *ga* is replaced by the topic marker *wa*.

Similarly, an argument modified by its predicate does not accompany a postpositional case particle that represents the syntactic relation between the predicate and argument. For example, since *camera* in Example (5) is modified by the predicate *miseta* (showed), the accusative case particle *wo* does not appear explicitly.

(5) *Watashi-ga kare-ni miseta camera.*
I−NOM him−DAT showed camera

(A camera that I showed him.)

In addition, arguments are often omitted in Japanese when we can easily guess what the omitted argument is or we do not suppose a specific object. For example, the nominative argument is omitted in Example (4)-c, since we can easily guess the subject is the first person.

These characteristics make it difficult to automatically extract examples of word orders in double object construction from a corpus.

2.2 Canonical argument order

There are three major claims as to the canonical argument order of Japanese double object constructions (Koizumi and Tamaoka, 2004).

One is the traditional analysis by Hoji (1985), which argues that only the nominative, dative, accusative (DAT−ACC) order like in Example (1)-a is canonical for all cases. The second claim, made by Matsuoka (2003), argues that Japanese double object constructions have two canonical word orders, the DAT−ACC order and the ACC−DAT order, depending on the verb types. The third claim, by Miyagawa (1997), asserts that both the DAT−ACC order and ACC−DAT order are canonical for all cases.

Note that, the definition of the term *canonical word order* varies from study to study. Some studies presume that there is only one canonical word order for one construction (Hoji, 1985), while others presume that a canonical word order can be different for each verb or each tuple of a verb and its arguments (Matsuoka, 2003). In addition, some studies presume that there can be multiple canonical word orders for one sentence (Miyagawa, 1997). In this paper, we basically adopt the position that there is only one canonical word order for one tuple of a verb and its arguments but the canonical word orders can be different for different tuples of a verb and its arguments.

2.3 Other features related to word order

There are a number of known features that affect word order. For example, it is often said that long arguments tend to be placed far from the verb, whereas short arguments tend to be placed near the verb. The From-Old-to-New Principle (Kuno, 2006) is also well known; it argues that the unmarked word order of constituents is old, predictable information first; and new, unpredictable information last. Note that these types of features are not specific to argument orders in Japanese double object constituents. For example, Bresnan et al. (2007) reported the similar features were also observed in the English dative alternation and useful for predicting the dative alternation.

However, since we are interested in the canonical word order, we do not want to take these features into account. In this work, we assume that these features can be ignored by using a very large corpus and analyzing the word order on the basis of statistical information acquired from the corpus.

3 Claims on the canonical word order of Japanese double object constructions

In this paper, we will address the following five claims on the canonical word order of Japanese double object constructions.

Claim A: The DAT−ACC order is canonical.

Claim B: There are two canonical word orders, the DAT−ACC and the ACC−DAT order, depending on the verb types.

Claim C: An argument whose grammatical case is infrequently omitted with a given verb tends to be placed near the verb.

Claim D: The canonical word order varies depending on the semantic role and animacy of the dative argument.

Claim E: An argument that frequently co-occurs with the verb tends to be placed near the verb.

Claim A (Hoji, 1985) presumes that there is only one canonical word order for Japanese double object constructions regardless of the verb type. On the other hand, Claims B and C argue that the canonical word order varies depending on verb, but they still do not take into account the lexical information of the arguments. Thus, these claims can be verified by investigating the distribution of word orders for each verb.

With regard to Claim B, Matsuoka (2003) classified causative-inchoative alternating verbs into two types: show-type and pass-type, and claimed the DAT−ACC order is the canonical order for show-type verbs, whereas the ACC−DAT order is the canonical order for pass-type verbs. The definitions of each verb type are as follows. In the case of show-type verbs, the dative argument of a causative sentence is the subject of its corresponding inchoative sentence as shown in Example (6). On the other hand, in the case of pass-type verbs, the accusative argument is the subject of its corresponding inchoative sentence as shown in Example (7).

(6) **Causative:** *Kare-ni camera-wo miseta.*
him−DAT camera−ACC showed
(ϕ_I showed him a camera.)

Inchoative: *Kare-ga mita.*
he−NOM saw
(He saw $\phi_{something}$.)

(7) **Causative:** *Camera-wo kare-ni watashita.*
camera−ACC him−DAT passed
(ϕ_I passed him a camera.)

Inchoative: *Camera-ga watatta.*
camera−NOM passed
(A camera passed to $\phi_{someone}$.)

Claim C is based on our observation. It is based on the assumption that if an argument of a verb is important for interpreting the meaning of the verb, it tends to be placed near the verb and does not tend to be omitted.

Claims D and E take into account the lexical information of arguments and assume that the canonical word order of Japanese double object constructions is affected by the characteristics of the dative and/or accusative arguments. With regard to Claim D, Matsuoka (2003) asserted that the canonical order varies depending on the semantic role of the dative argument. Specifically, the DAT−ACC order is more preferred when the semantic role of dative argument is animate *Possessor* than when the semantic role is inanimate *Goal*.

Claim E is based on our observation again, which argues that if the dative or accusative argument frequently co-occurs with the verb, it has a strong relationship with the verb, and thus is placed nearby. A typical example that satisfies this claim is idiomatic expressions as will be discussed in Section 5.4.

4 Example collection

A corpus-based analysis of canonical word order can leverage a much larger number of examples than approaches based on theoretical analysis, psychological experiments, or brain science can. However, automatically collected examples sometimes include inappropriate ones. For example, if we extract all sequences of a verb and its preceding argument candidates, the sequence "*Kagi-wo kare-ni iwareta*" (ϕ_I am told the key by him) is mistakenly extracted from Example (8), although *kagi-wo* is not actually an argument of *iwareta* but an argument of *oita*.

(8) *Kagi-wo kare-ni iwareta basho-ni oita.*
key−ACC him−DAT told place−DAT put
(ϕ_I put the key on the place where he told ϕ_{me}.)

As predicted, we can alleviate this problem by using a dependency parser. However, the accuracy of the state-of-the-art Japanese dependency parser is not very high, specifically about 92% for news paper articles (Yoshinaga and Kitsuregawa, 2014), and thus, inappropriate examples would be extracted even if we used one.

Therefore, in this work, we decided to extract examples only from reliable parts of dependency parses. Specifically, we used a corpus consisting of more than 10 billion unique sentences extracted from the Web, selected parse trees that have no syntactic ambiguity, and then extracted examples only from the selected parse trees. This strategy basically follows Kawahara and Kurohashi (2002)'s strategy for automatic case frame construction. The detailed procedure of example collection is as follows:

1. Extract Japanese Web pages using linguistic information, split the Web pages into sentences using periods and HTML tags, and merge sentences that are the exactly same into one sentence to avoid collecting the same example several times, which might be extracted from a mirror site.

2. Employ the Japanese morphological analyzer JUMAN[3] and the syntactic analyzer KNP[4], and extract examples of verbs and their arguments from parse trees that have no syntactic ambiguity[5].

3. Collect the examples if the verb satisfies all the following conditions:

   (a) The verb has an entry in the JUMAN dictionary and appears in the active voice.

   (b) The verb has more than 500 different examples of dative and accusative argument pairs.

   (c) The proportion of examples that include both the dative and accusative arguments out of all examples that include the target verb is larger than 5%.

We employ the syntactic analyzer KNP with options "-dpnd-fast -tab -check." KNP with these

[3]http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?JUMAN

[4]http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?KNP

[5]Murawaki and Kurohashi (2012) reported that 20.7% of the dependency relations were extracted from a newspaper corpus and the accuracy was 98.3% when they adopted Kawahara and Kurohashi (2002)'s strategy.

options outputs all head candidates for each *bunsetsu*[6] on the basis of heuristic rules. We then extract the example of a verb and its argument if the argument candidate have only one head candidate.

For example, since Japanese is a head-final language and only the verb *bunsetsu* can be the head of the most noun *bunsetsu* in Japanese, *basho-ni* in Example (8) has only one head candidate *oita* (put), whereas *kagi-wo* and *kare-ni* have two head candidates *iwareta* (told) and *oita* (put). Thus, we extract only the example "*basho-ni oita*" from Example (8). In addition, when an argument consists of a compound noun, we only extract the head noun and its postpositional particle as the argument to avoid data sparsity.

Condition 3-(c) is set in order to extract only ditransitive verbs, which take both dative and accusative arguments. Although the threshold of 5% seems small at first glance, most verbs that satisfy it are actually ditransitive. This is because arguments are often omitted in Japanese, and thus, only some of the examples explicitly include both dative and accusative arguments even in the case of ditransitive verb.

Out of a corpus consisting of more than 10 billion unique sentences, 648 verbs satisfied these conditions. Hereafter, we will focus on these 648 verbs. The average number of occurrences of each verb was about 350 thousand and the average number of extracted examples that include both dative and accusative arguments was about 59 thousand.

5 Corpus-based analysis of canonical word order

Here, we present a corpus-based analysis of the canonical word order of Japanese double object constructions. We will address Claims A and C in Section 5.1, Claim B in Section 5.2, Claim D in Section 5.3, and Claim E in Section 5.4.

5.1 Word order for each verb

Let us examine the relation between the proportion of the DAT only example $R_{\text{DAT-only}}$ and the proportion of the ACC−DAT order $R_{\text{ACC-DAT}}$ for each of the 648 verbs to inspect Claims A and C.

[6]In Japanese, *bunsetsu* is a basic unit of dependency, consisting of one or more content words and the following zero or more function words. In this paper, we segment each example sentence into a sequence of *bunsetsu*.

$R_{\text{DAT-only}}$ is calculated as follows:

$$R_{\text{DAT-only}} = \frac{N_{\text{DAT-only}}}{N_{\text{DAT-only}} + N_{\text{ACC-only}}},$$

where $N_{\text{DAT/ACC-only}}$ is the number of example types that only include the corresponding argument out of the dative and accusative arguments. For example, we count the number of example types like Example (9) that include an accusative argument but do not include a dative argument to get the value of $N_{\text{ACC-only}}$. Accordingly, the large $R_{\text{DAT-only}}$ value indicates that the dative argument is less frequently omitted than the accusative argument.

(9) *Gakuchô-ga gakui-wo juyo-shita.*
president−NOM degree−ACC conferred
(The president conferred a degree on $\phi_{someone}$.)

However, if we use all extracted examples that include only one of the dative and accusative arguments for calculating $R_{\text{DAT-only}}$, the value is likely to suffer from a bias that the larger $R_{\text{ACC-DAT}}$ is, the larger $R_{\text{DAT-only}}$ becomes. This is because the arguments that tend to be placed near the verb have relatively few syntactic ambiguity. Since we extract the examples from the reliable parts of parses that have no syntactic ambiguity, these arguments tend to be included in the extracted examples more frequently than the other arguments.

To avoid this bias, we use only these examples in which the nominative case is also extracted for calculating $R_{\text{DAT-only}}$. This is based on the assumption that since Japanese word order is basically subject-object-verb order, if the nominative argument is collected but one of the dative and accusative arguments is not collected, the argument is actually omitted. Through a preliminary investigation on Kyoto University Text Corpus[7], we confirmed the effect of this constraint to avoid the bias.

On the other hand, $R_{\text{ACC-DAT}}$ is calculated as follows:

$$R_{\text{ACC-DAT}} = \frac{N_{\text{ACC-DAT}}}{N_{\text{DAT-ACC}} + N_{\text{ACC-DAT}}},$$

where $N_{\text{DAT-ACC/ACC-DAT}}$ is the number of example types that include both the dative and accusative arguments in the corresponding order.

Figure 1 shows the results. The left figure shows the relation between the proportion of the DAT

[7]Kyoto University Text Corpus 4.0: http://nlp.ist.i.kyoto-u.ac.jp/EN/index.php?Kyoto University Text Corpus

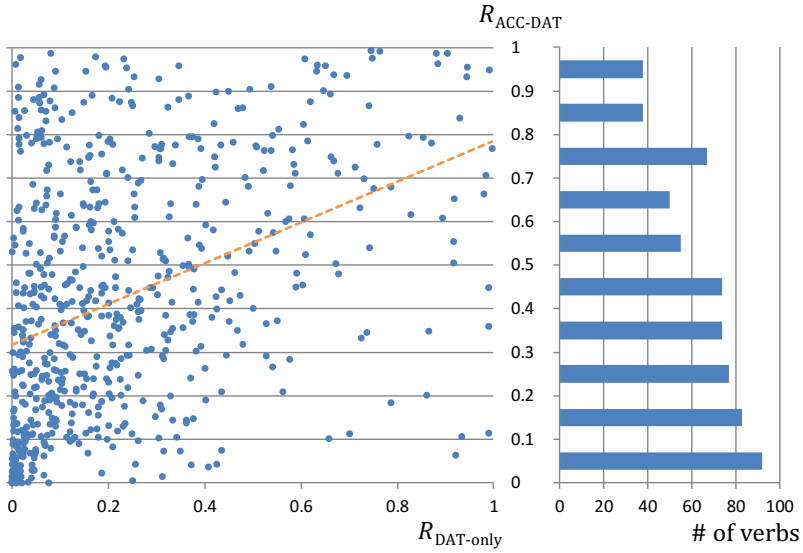


Figure 1: The left figure shows the relation between the proportion of the DAT only example $R_{\text{DAT-only}}$ (x-axis) and the proportion of the ACC−DAT order $R_{\text{ACC-DAT}}$ (y-axis). The right figure shows the number of verbs in the corresponding range of $R_{\text{ACC-DAT}}$.

only example $R_{\text{DAT-only}}$ and the proportion of the ACC−DAT order $R_{\text{ACC-DAT}}$ for each of the 648 verbs. The x-axis denotes $R_{\text{DAT-only}}$, the y-axis denotes $R_{\text{ACC-DAT}}$, and each point in the figure represents one of the 648 verbs. The dashed line is a linear regression line. The right figure shows the number of verbs in the corresponding range of $R_{\text{ACC-DAT}}$.

Pearson's correlation coefficient between $R_{\text{DAT-only}}$ and $R_{\text{ACC-DAT}}$ is 0.391, which weakly supports Claim C: an argument whose grammatical case is infrequently omitted with a given verb tends to be placed near the verb. The proportion of the ACC−DAT order for all 648 verbs is 0.328. Thus, if we presume that there is only one canonical word order for Japanese double object constructions, this result suggests that the DAT−ACC order is the canonical one, as claimed by Hoji (Claim A). However, the right figure shows that the proportions of the ACC−DAT order differ from verb to verb. Moreover, the values of $R_{\text{ACC-DAT}}$ for 435 out of 648 verbs are between 0.2 and 0.8. From these observations, we can say the preferred word order cannot be determined even if the verb is given in most cases.

5.2 Word order and verb type

To inspect Matsuoka (2003)'s claim that the DAT−ACC order is canonical for show-type verbs, whereas the ACC−DAT order is canonical for pass-type verbs, we investigated the proportions of the ACC−DAT order for several pass-type and show-type verbs. In this investigation, we used 11 pass-type verbs and 22 show-type verbs that were used by Koizumi and Tamaoka (2004) in their psychological experiments[8].

Table 1 shows the results. Although we can see that the macro average of $R_{\text{ACC-DAT}}$ of pass-type verbs is larger than that of show-type verbs, the difference is not significant[9]. Moreover, even in the case of pass-type verbs, the DAT−ACC order is dominant, which suggests Matsuoka (2003)'s claim is not true. Note that this conclusion is consistent with the experimental results reported by both Miyamoto and Takahashi (2002) and Koizumi and Tamaoka (2004).

5.3 Relation between word order and semantic role of the dative argument

Next, let us examine the relation between the category of the dative argument and the word order to verify the effect of the semantic role of the dative argument. We selected eight categories in the JUMAN dictionary[10] that appear more than 1 million times as dative arguments. Table 2 shows the results. We can see that there are differences in the

[8] We excluded a show-type verb *hakaseru* (dress), since it is divided into two morphemes by JUMAN. Instead, we added two show-type verbs *shiraseru* (notify) and *kotodukeru* (leave a message).

[9] The two-tailed p-value of permutation test is about 0.177.

[10] In JUMAN dictionary, 22 categories are defined and tagged to common nouns.

Show-type		Pass-type			
verb	$R_{\texttt{ACC-DAT}}$	verb	$R_{\texttt{ACC-DAT}}$	verb	$R_{\texttt{ACC-DAT}}$
shiraseru (notify)	0.522	*modosu* (put back)	0.771	*otosu* (drop)	0.351
azukeru (deposit)	0.399	*tomeru* (lodge)	0.748	*morasu* (leak)	0.332
kotodukeru (leave a message)	0.386	*tsutsumu* (wrap)	0.603	*ukaberu* (float)	0.255
satosu (admonish)	0.325	*tsutaeru* (inform)	0.522	*mukeru* (direct)	0.251
miseru (show)	0.301	*noseru* (place on)	0.496	*nokosu* (leave)	0.238
kabuseru (cover)	0.256	*todokeru* (deliver)	0.491	*umeru* (bury)	0.223
osieru (teach)	0.235	*naraberu* (range)	0.481	*mazeru* (blend)	0.200
sazukeru (give)	0.186	*kaesu* (give back)	0.448	*ateru* (hit)	0.185
abiseru (shower)	0.177	*butsukeru* (knock)	0.436	*kakeru* (hang)	0.108
kasu (lend)	0.118	*tsukeru* (attach)	0.368	*kasaneru* (pile)	0.084
kiseru (dress)	0.113	*watasu* (pass)	0.362	*tateru* (build)	0.069
Macro average	0.274			Macro average	0.365

Table 1: Proportions of the ACC−DAT order for each pass-type verb and show-type verb.

Category	# of examples	$R_{\texttt{ACC-DAT}}$	Typical examples
PLACE-FUNCTION	1376990	0.499	*shita* (bottom), *yoko* (side), *soto* (outside), *hôkô* (direction), ...
ANIMAL-PART	1483885	0.441	*te* (hand), *mi* (body), *atama* (head), *hada* (skin), *mune* (chest), ...
PERSON	5511281	0.387	*tomodachi* (friend), *hito* (human), *shichô* (mayor), *watashi* (I), ...
ARTIFACT-OTHER	2751008	0.372	*pasokon* (PC), *fairu* (file), *furo* (bath), *hon* (book), ...
PLACE-INSTITUTION	1618690	0.342	*heya* (room), *mise* (shop), *tokoro* (location), *gakkô* (school), ...
PLACE-OTHER	2439188	0.341	*basho* (place), *sekai* (world), *ichi* (position), *zenmen* (front), ...
QUANTITY	1100222	0.308	*zu* (figure), *hyô* (table), *hanbun* (half), *atai* (value), ...
ABSTRACT	10219318	0.307	*blog* (blog), *kokoro* (mind), *list* (list), *shiya* (sight), ...
Total	26500582	0.353	

Table 2: Proportions of the ACC−DAT order for each category of dative argument.

proportions of the ACC−DAT order. In particular, when the dative argument's category is PLACE-FUNCTION such as *shita* (bottom) and *yoko* (side) or ANIMAL-PART such as *te* (hand) and *mi* (body), the ACC−DAT order is more preferred than otherwise.

As mentioned in Section 3, Matsuoka (2003) claimed the DAT−ACC order is more preferred when the semantic role of the dative argument is animate *Possessor* than when the semantic role is inanimate *Goal*. Thus, we thought the DAT−ACC order would be preferred when the dative argument's category is PERSON, but we could not find such a trend. We think, however, this is due to that dative arguments of the PERSON category do not always have the semantic role of an animate *Possessor*. Thus, we conducted a further investigation in an attempt to verify Matsuoka (2003)'s claim.

First, we collected examples that satisfied the following two conditions: the accusative argument belongs to ARTIFACT-OTHER category, and the dative argument belongs to either PLACE-INSTITUTION or PERSON category. We call the former Type-A[11], and the latter Type-B hereafter, and consider that the semantic role of the dative argument is inanimate *Goal* in most cases of Type-A, whereas it is animate *Possessor* in most cases of Type-B. Example (10) shows typical examples of Type-A and Type-B. Here, the categories of *hon* (book), *gakkô* (school), and *sensei* (teacher) are ARTIFACT-OTHER, PLACE-INSTITUTION, and PERSON, respectively, and the semantic roles of dative arguments are considered to be *Goal* in (10)-a and *Possessor* in (10)-b.

(10) a: *Hon-wo* *gakkô-ni* *kaeshita.*
book−ACC school−DAT returned
($\phi_{someone}$ returned the book to school.)

b: *Sensei-ni* *hon-wo* *kaeshita.*
teacher−DAT book−ACC returned
($\phi_{someone}$ returned the book to the teacher.)

Next, we extracted verbs that had at least 100 examples of both types, calculated the proportion of the ACC−DAT order $R_{\texttt{ACC-DAT}}$ for each verb and type, and counted the number of verbs for which the values of $R_{\texttt{ACC-DAT}}$ were significantly different between Type-A and Type-B[12]. Out of 126 verbs that have at least 100 examples for both types, 64 verbs show the trend that Type-A prefers the ACC−DAT order more than Type-B does, and only 30 verbs have the opposite trend. This fact supports Matsuoka (2003)'s claim.

[11] That is, the categories of the accusative and dative arguments of a Type-A example are ARTIFACT-OTHER and PLACE-INSTITUTION, respectively.

[12] We conducted a two-proportion z-test with a significance level of 0.05.

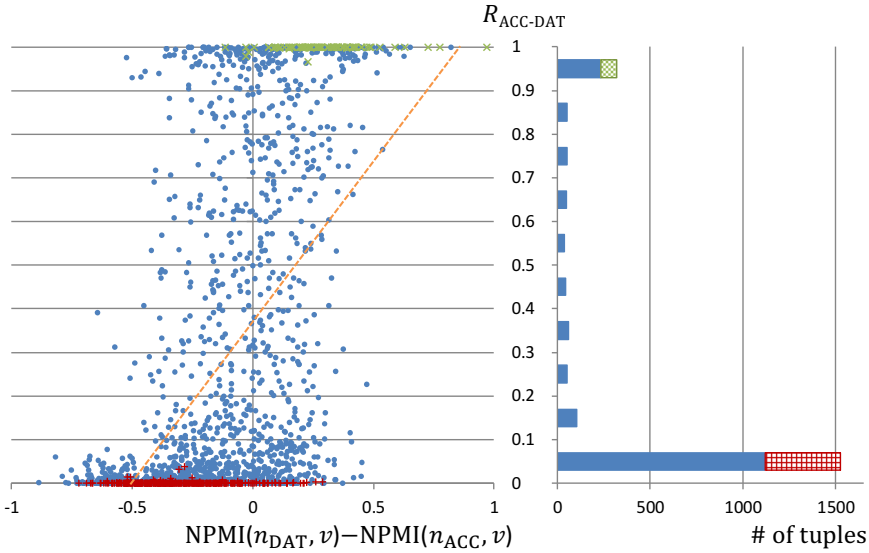


Figure 2: The left figure shows the relation between the difference of NPMI(n_{DAT}, v) from NPMI(n_{ACC}, v) (x-axis) and the proportion of the ACC−DAT order $R_{\text{ACC-DAT}}$ (y-axis). The tuples whose verb and accusative/dative argument are used as an idiom are represented by +/×. The right figure shows the number of tuples of a verb and its dative and accusative arguments in the corresponding range of $R_{\text{ACC-DAT}}$.

5.4 Relation between word order and degree of co-occurrence of verb and arguments

Now let us turn to the relation between the proportion of the ACC−DAT order $R_{\text{ACC-DAT}}$ and the degree of co-occurrence of a verb and its argument to verify Claim E. Here, we leverage the normalized pointwise mutual information (NPMI) for measuring the degree of co-occurrence between a verb and its argument. NPMI is a normalized version of PMI. The value ranges between [-1,+1] and takes -1 for never occurring together, 0 for independence, +1 for complete co-occurrence. The NPMI of a verb v and its argument n_c ($c \in \{\text{DAT}, \text{ACC}\}$) is calculated as

$$\text{NPMI}(n_c, v) = \frac{\text{PMI}(n_c, v)}{-\log(p(n_c, v))},$$

$$\text{where PMI}(n_c, v) = \log \frac{p(n_c, v)}{p(n_c)p(v)}.$$

We investigate the relation between the proportion of the ACC−DAT order $R_{\text{ACC-DAT}}$ and the difference of NPMI(n_{DAT}, v) from NPMI(n_{ACC}, v), i.e., NPMI(n_{DAT}, v) − NPMI(n_{ACC}, v). If Claim E is true, when the dative argument co-occurs with the verb frequently, the dative argument tends to be placed near the verb and thus the proportion of the ACC−DAT order would take a large value.

We investigated 2302 tuples of a verb and its dative and accusative arguments that appear more than 500 times in the corpus. The average number of occurrences of each tuple was 1532. Figure 2 shows the results. The left figure shows the relation between the difference of NPMI(n_{DAT}, v) from NPMI(n_{ACC}, v) and the proportion of the ACC−DAT order $R_{\text{ACC-DAT}}$. Each point in the figure represents one of the 2302 tuples. The dashed line is a linear regression line. The right figure shows the number of tuples in the corresponding range of $R_{\text{ACC-DAT}}$.

Pearson's correlation coefficient between the difference of NPMI and $R_{\text{ACC-DAT}}$ is 0.567, which supports Claim E: an argument that frequently co-occurs with the verb tends to be placed near the verb. Moreover, the values of $R_{\text{ACC-DAT}}$ are larger than 0.9 or smaller than 0.1 for 1631 out of 2302 tuples. This result indicates that if a tuple of a verb and its dative and accusative arguments is given, the preferred word order is determined. This is contrastive to the conclusion that the preferred word order cannot be determined even if the verb is given as discussed in Section 5.1.

One of the typical examples that satisfy Claim E is an idiomatic expression. Indeed, a verb and its argument that are used as an idiom co-occur frequently and are usually placed adjacent to each other. In addition, it is well known that if the argument order is scrambled, the idiomatic meaning disappears (Miyagawa and Tsujioka, 2004). Thus, we investigated to what extent idiomatic expres-

sions affected the findings discussed above. For all 2302 tuples, we manually judged whether the verb and the adjacent argument are used as an idiom in most cases. As a result, the verbs and their accusative arguments are judged as idiomatic for 404; the verbs and their dative arguments are judged as idiomatic for 84 out of 2302 tuples. We show these tuples by $+$ and $\times$ in Figure 2, respectively. As predicted, the values of $R_{\texttt{ACC-DAT}}$ are smaller than 0.1 for all of the former examples, and larger than 0.9 for all of the latter examples. However, even if we ignore these idiomatic examples, Pearson's correlation coefficient between the difference of NPMI and $R_{\texttt{ACC-DAT}}$ is 0.513, which is usually considered as moderate correlation.

6 Conclusion

This paper presented a corpus-based analysis of canonical word order of Japanese double object constructions. Our analysis suggests 1) the canonical word order of such constructions varies from verb to verb, 2) there is only a weak relation between the canonical word order and the verb type: show-type and pass-type, 3) an argument whose grammatical case is infrequently omitted with a given verb tends to be placed near the verb, 4) the canonical word order varies depending on the semantic role of the dative argument, and 5) an argument that frequently co-occurs with the verb tends to be placed near the verb.

Acknowledgments

We would like to thank Sadao Kurohashi and Daisuke Kawahara for helping us to collect examples from the Web. This work was supported by JSPS KAKENHI Grant Number 25730131 and 16K16110.

References

Joan Bresnan, Anna Cueni, Tatiana Nikitina, and Harald Baayen. 2007. Predicting the dative alternation. In Gerlof Bouma, Irene Krämer, and Joost Zwarts, editors, *Cognitive foundations of interpretation*, pages 69–94. Amsterdam: Royal Netherlands Academy of Science.

Hajime Hoji. 1985. *Logical Form Constraints and Configurational Structures in Japanese*. Ph.D. thesis, University of Washington.

Tomoo Inubushi, Kazuki Iijima, Masatoshi Koizumi, and Kuniyoshi L. Sakai. 2009. The effect of canonical word orders on the neural processing of double object sentences: An MEG study. In *Proceedings of the 32nd Annual Meeting of the Japan Neuroscience Society*.

Daisuke Kawahara and Sadao Kurohashi. 2002. Fertilization of case frame dictionary for robust Japanese case analysis. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002)*, pages 425–431.

Masatoshi Koizumi and Katsuo Tamaoka. 2004. Cognitive processing of Japanese sentences with ditransitive verbs. *Gengo Kenkyu*, 125:173–190.

Ayumi Koso, Hiroko Hagiwara, and Takahiro Soshi. 2004. What a multi-channel EEG system reveals about the processing of Japanese double object constructions (in Japanese). In *Technical report of IEICE. Thought and language 104(170)*, pages 31–36.

Susumu Kuno. 2006. Empathy and direct discourse perspectives. In Laurence Horn and Gergory Ward, editors, *The Handbook of Pragmatics*, Blackwell Handbooks in Linguistics, pages 315–343. Wiley.

Mikinari Matsuoka. 2003. Two types of ditransitive consturctions in Japanese. *Journal of East Asian Linguistics*, 12:171–203.

Shigeru Miyagawa and Takae Tsujioka. 2004. Argument structure and ditransitive verbs in Japanese. *Journal of East Asian linguistics*, 13:1–38.

Shigeru Miyagawa. 1997. Against optional scrambling. *Linguistic Inquiry*, 28:1–26.

Edson T. Miyamoto and Shoichi Takahashi. 2002. Sources of difficulty in processing scrambling in Japanese. In Mineharu Nakayama, editor, *Sentence processing in East Asian languages. Stanford, Calif*, pages 167–188. CSLI Publications.

Yugo Murawaki and Sadao Kurohashi. 2012. Semi-supervised noun compound analysis with edge and span features. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*, pages 1915–1932.

Keiko Nakamoto, Jae-Ho Lee, and Kow Kuroda. 2006. Cognitive mechanisms for sentence comprehension preferred word orders correlate with "sentential" meanings that cannot be reduced to verb meanings: A new perspective on "construction effects" in Japanese (in Japanese). *Cognitive Studies*, 13(3):334–352.

Yasumasa Shigenaga. 2014. Canonical word order of Japanese ditransitive sentences: A preliminary investigation through a grammaticality judgment survey. *Advances in Language and Literary Studies*, 5(2):35–45.

Naoki Yoshinaga and Masaru Kitsuregawa. 2014. A self-adaptive classifier for efficient text-stream processing. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2014)*, pages 1091–1102.

Knowledge-Based Semantic Embedding for Machine Translation

Chen Shi[†*] **Shujie Liu**[‡] **Shuo Ren**[‡] **Shi Feng**[§]
Mu Li[‡] **Ming Zhou**[‡] **Xu Sun**[†] **Houfeng Wang**[†¶]
[†]MOE Key Lab of Computational Linguistics, Peking University
[‡]Microsoft Research Asia [§]Shanghai Jiao Tong University
[¶]Collaborative Innovation Center for Language Ability
{shichen, xusun, wanghf}@pku.edu.cn sjtufs@gmail.com
{shujliu, v-shuren, muli, mingzhou}@microsoft.com

Abstract

In this paper, with the help of knowledge base, we build and formulate a semantic space to connect the source and target languages, and apply it to the sequence-to-sequence framework to propose a Knowledge-Based Semantic Embedding (KBSE) method. In our KBSE method, the source sentence is firstly mapped into a knowledge based semantic space, and the target sentence is generated using a recurrent neural network with the internal meaning preserved. Experiments are conducted on two translation tasks, the electric business data and movie data, and the results show that our proposed method can achieve outstanding performance, compared with both the traditional SMT methods and the existing encoder-decoder models.

1 Introduction

Deep neural network based machine translation, such as sequence-to-sequence (S2S) model (Cho et al., 2014; Sutskever et al., 2014), try to learn translation relation in a continuous vector space. As shown in Figure 1, the S2S framework contains two parts: an encoder and a decoder. To compress a variable-length source sentence into a fixed-size vector, with a recurrent neural network (RNN), an encoder reads words one by one and generates a sequence of hidden vectors. By reading all the source words, the final hidden vector should contain the information of source sentence, and it is called the context vector. Based on the context vector, another RNN-based neural network is used to generate the target sentence.

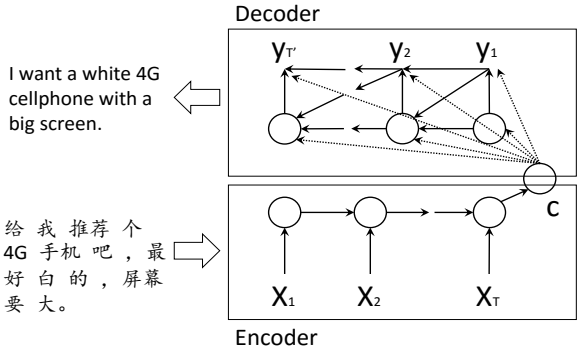


Figure 1: An illustration of the RNN-based neural network model for Chinese-to-English machine translation

The context vector plays a key role in the connection of source and target language spaces, and it should contain all the internal meaning extracted from source sentence, based on which, the decoder can generate the target sentence keeping the meaning unchanged. To extract the internal meaning and generate the target sentence, S2S framework usually needs large number of parameters, and a big bilingual corpus is acquired to train them.

In many cases, the internal meaning is not easy to learn, especially when the language is informal. For the same intention, there are various expressions with very different surface string, which aggravates the difficulty of internal meaning extraction. As shown in Table 1, there are three different expressions for a same intention, a customer wants a white 4G cellphone with a big screen. The first and second expressions (Source1 and Source2) are wordy and contain lots of verbiage. To extract the internal meaning, the encoder should ignore these verbiage and focus on key information. This is hard for the encoder-decoder mechanism, since it is not defined or formulated that what kind of information is key information. The meaning s-

This work was done while the first author was visiting Microsoft Research.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2245–2254,
Berlin, Germany, August 7-12, 2016.

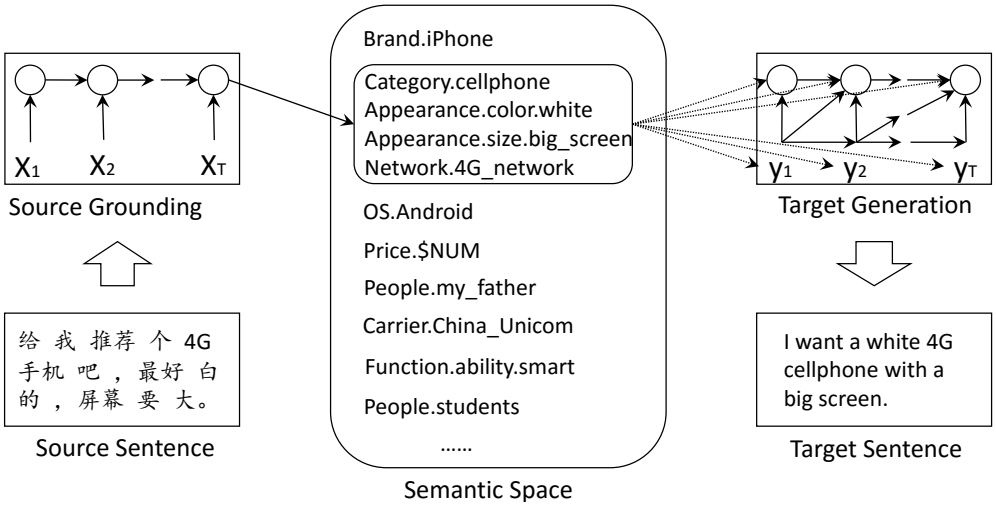


Figure 2: An illustration of Knowledge-Based Semantic Embedding (KBSE).

Source1	啊， 那个 有 大 屏幕 的 4G 手机 吗？ 要 白色 的 。
Source2	给 我 推荐 个 4G 手机 吧， 最好 白 的， 屏幕 要 大 。
Source3	我 想 买个 白色 的 大 屏幕 的 4G 手机 。
Intention	I want a white 4G cellphone with a big screen.
Enc-Dec	I need a 4G cellphone with a big screen.

Table 1: An example of various expressions for a same intention.

pace of the context vector is only a vector space of continuous numbers, and users cannot add external knowledge to constrain the internal meaning space. Therefore, the encoder-decoder system (Enc-Dec) does not generate the translation of "白色的"/"white", and fails to preserve the correct meaning of Source1, shown in Table 1.

No matter how different between the surface strings, the key information is the same (*want, white, 4G, big screen, cellphone*). This phenomenon motivates a translation process as: we firstly extract key information (such as entities and their relations) from the source sentence; then based on that, we generate target sentence, in which entities are translated with unchanged predication relations. To achieve this, background knowledge (such as, phone/computer, black/white, 3G/4G) should be considered.

In this paper, we propose a Knowledge-Based Semantic Embedding (KBSE) method for machine translation, as shown in Figure 2. Our KBSE contains two parts: a **Source Grounding** part to extract semantic information in source sentence, and a **Target Generation** part to generate target sentence. In KBSE, source monolingual data and a knowledge base is leveraged to learn an explicit semantic vector, in which the grounding space is defined by the given knowledge base, then the same knowledge base and a target monolingual data are used to learn a natural language generator, which produce the target sentence based on the learned explicit semantic vector. Different from S2S models using large bilingual corpus, our KBSE only needs monolingual data and corresponding knowledge base. Also the context/semantic vector in our KBSE is no longer implicit continuous number vector, but explicit semantic vector. The semantic space is defined by knowledge base, thus key information can be extracted and grounded from source sentence. In such a way, users can easily add external knowledge to guide the model to generate correct translation results.

We conduct experiments to evaluate our KBSE on two Chinese-to-English translation tasks, one in electric business domain, and the other in movie domain. Our method is compared with phrasal SMT method and the encoder-decoder method, and achieves significant improvement in both BLEU and human evaluation. KBSE is also combined with encoder-decoder method to get further improvement.

In the following, we first introduce our framework of KBSE in section 2, in which the details of **Source Grounding** and **Target Generation** are illustrated. Experiments is conducted in Section 3. Discussion and related work are detailed in Section 4, followed by conclusion and future work.

2 KBSE: Knowledge-Based Semantic Embedding

Our proposed KBSE contains two parts: **Source Grounding** part (in Section 2.1) embeds the source sentence into a knowledge semantic space, in which the grounded semantic information can be represented by semantic tuples; and **Target Generation** part (in Section 2.2) generates the target sentence based on these semantic tuples.

2.1 Source Grounding

Source	啊，那个有大屏幕的4G手机吗？要白色的。
Tuples	*Category.cellphone* *Appearance.color.white* *Appearance.size.big_screen* *Network.4G_network*

Table 2: Source sentence and the grounding result. Grounding result is organized as several tuples.

As shown in Table 2, given the source sentence, Source Grounding part tries to extract the semantic information, and map it to the tuples of knowledge base. It is worth noticing that the tuples are language-irrelevant, while the name of the entities inside can be in different languages. To get the semantic tuples, we first use RNN to encode the source sentence into a real space to get the sentence embedding, based on which, corresponding semantic tuples are generated with a neural-network-based hierarchical classifier. Since the knowledge base is organized in a tree structure, the tuples can be seen as several paths in the tree. For tuples in Table 2, Figure 3 shows the corresponding paths (in solid lines).

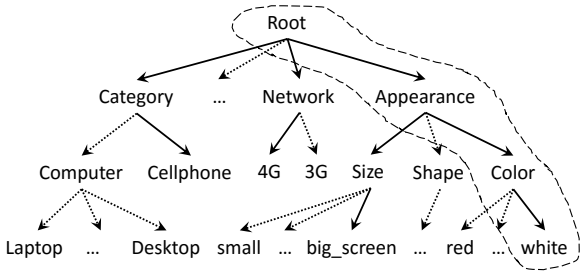


Figure 3: Illustration of the tuple tree for Table 2. Each tuple extracted from source sentence can be represented as a single path (solid line) in tuple tree. There are 4 solid line paths representing 4 tuples of Table 2. The path circled in dashed lines stands for the tuple *Appearance.color.white*.

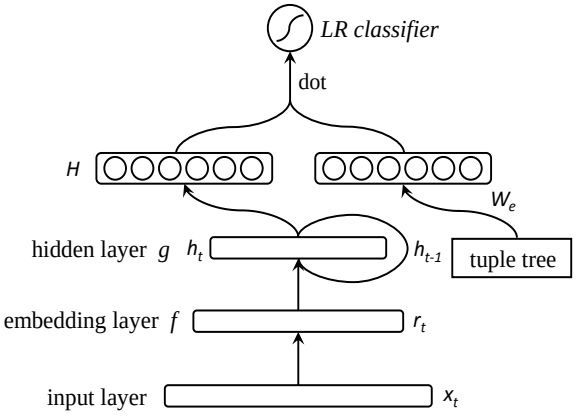


Figure 4: Illustration of Source Grounding. The input sentence **x** is transformed through an embedding layer f and a hidden layer g. Once we get the sentence embedding H, we calculate the inner product of H and the weight W_e for the specific edge e, and use a logistic regression as the classifier to decide whether this edge should be chosen.

2.1.1 Sentence Embedding

Sentence embedding is used to compress the variable-length source sentence into a fixed-size context vector. Given the input sentence $\mathbf{x} = (x_1 \ ... \ x_T)$, we feed each word one by one into an RNN, and the final hidden vector is used as the sentence embedding. In detail, as shown in Figure 4, at time-stamp t, an input word x_t is fed into the neural network. With the embedding layer f, the word is mapped into a real vector $r_t = f(x_t)$. Then the word embedding r_t is fed into an RNN g to get the hidden vector $h_t = g(r_t, h_{t-1})$. We input the words one by one at time $1, 2, ..., T$, and get the hidden vectors $h_1, h_2, ..., h_T$. The last hidden state h_T should contain all the information of the input sentence, and it is used as the sentence embedding H. To model the long dependency and memorize the information of words far from the end, Gated Recurrent Unit(GRU) (Cho et al., 2014) is leveraged as the recurrent function g.

2.1.2 Tuple Generation

In our system, we need a tuple tree for tuple generation. For those knowledge base who is naturally organized as tree structure, such as Freebase, we use its own stucture. Otherwise, we manually build the tuple tree as the representation of the introduced knowledge base. Given a knowl-

edge base for a specific domain, we divide the intention of this domain into several classes, while each class has subclasses. All the classes above can be organized as a tree structure, which is the tuple tree we used in our system, as shown in Figure 3. It is worth noticing that the knowledge base captures different intentions separately in different tree structures.

Following the hierarchical log-bilinear model (HLBL) (Mnih and Hinton, 2009; Mikolov et al., 2013), based on the sentence embedding H, we build our neural-network-based hierarchical classifier as follows: Each edge e of tuple tree has a weight vector w_e, which is randomly initialized, and learned with training data. We go through the tuple tree top-down to find the available paths. For each current node, we have a classifier to decide which children can be chosen. Since several children can be chosen at the same time independently, we use logistic regression as the classifier for each single edge, rather than a softmax classifier to choose one best child node.

For the source sentence and corresponding tuples in table 2, in the first layer, we should choose three children nodes: *Category*, *Appearance* and *Network*, and in the second layer with the parent node *Appearance*, two children nodes *color* and *size* should be selected recursively. As shown in Figure 4, the probability to choose an edge e with its connected child is computed as follows:

$$p(1|e,H) = \frac{1}{1+e^{-w_e \cdot H}} \tag{1}$$

where the operator $\cdot$ is the dot product function. The probability of the tuples conditioned on the source sentence $p(S|x_1 \ ... \ x_T)$ is the product of all the edges probabilities, calculated as follows:

$$\begin{aligned} p(S|x_1 \ ... \ x_T) &= p(S|H) \\ &= \prod_{e \in C} p(1|e,H) \prod_{e' \notin C} p(0|e',H) \end{aligned}$$

where $p(1|e,H)$ is the probability for an edge e belonging to the tuple set S, and $p(0|e',H)$ is the probability for an edge e' not in the tuple set S.

2.2 Target Generation

With the semantic tuples grounded from source sentence, in this section, we illustrate how to generate target sentence. The generation of the target sentence is another RNN, which predicts the next word y_{t+1} conditioned on the semantic vector C and all the previously predicted words $y_1, \ ..., \ y_t$. Given current word y_t, previous hidden vector h_{t-1}, and the semantic vector C, the probability of next target word y_{t+1} is calculated as:

$$h_t = g(h_{t-1}, y_t, C) \tag{2}$$

$$p(y_{t+1}|y_1 ... y_t, C) = \frac{e^{s(y_{t+1},h_t)}}{\sum_{y'} e^{s(y',h_t)}} \tag{3}$$

where equation (2) is used to generate the next hidden vector h_t, and equation (3) is the *softmax* function to compute the probability of the next word y_{t+1}. For the recurrent function g in equation (2), in order to generate target sentence preserving the semantic meaning stored in C , we modified GRU (Cho et al., 2014) following (Wen et al., 2015; Feng et al., 2016):

$$\begin{aligned} r_t &= \sigma(W^r y_t + U^r h_{t-1} + V^r c_t) \\ h_t^{'} &= \tanh(W y_t + U(r_t \odot h_{t-1}) + V c_t) \\ z_t &= \sigma(W^z y_t + U^z h_{t-1} + V^z c_t) \\ d_t &= \sigma(W^d y_t + U^d h_{t-1} + V^d c_t) \\ c_t &= d_t \odot c_{t-1} \\ h_t &= (1 \ - \ z_t) \odot h_t^{'} \ + \ z_t \odot h_{t-1} + \tanh(V^h c_t) \end{aligned}$$

in which, c_t is the semantic embedding at time t, which is initialized with C, and changed with a extraction gate d_t. The introduced extraction gate d_t retrieve and remove information from the semantic vector C to generate the corresponding target word.

To force our model to generate the target sentence keeping information contained in C unchanged, two additional terms are introduced into the cost function:

$$\sum_t \log(p(y_t|C)) + \|c_T\|_2 + \frac{1}{T}\sum_{j=1}^{T} \|d_t - d_{t-1}\|_2$$

where the first term is log-likelihood cost, the same as in the encoder-decoder. And the other two terms are introduced penalty terms. $\|c_T\|_2$ is for forcing the decoding neural network to extract as much information as possible from the semantic vector C, thus the generated target sentence keeps the same meaning with the source sentence. The third term is to restrict the extract gate from extracting too much information in semantic vector C at each time-stamp.

For the semantic tuples in Table 2, our modified RNN generates the target sentence word by word, until meets the end symbol character: "*I want a white 4G cellphone with a big screen.*".

2.3 Combination

The two components of KBSE (**Source Grounding** and **Target Generation**) are separately trained, and can be used in three ways:

- **Source Grounding** can be used to do semantic grounding for a given sentence and get the key information as a form of tuples;
- **Target Generation** can generate a natural language sentence based on the existing semantic tuples;
- Combining them, KBSE can be used to translation a source sentence into another language with a semantic space defined by a given knowledge base.

3 Experiments

To evaluate our proposed KBSE model, in this section, we conduct experiments on two Chinese-to-English translation tasks. One is from electric business domain, and the other is from movie domain.

3.1 Baseline and Comparison Systems

We select two baseline systems. The first one is an in-house implementation of hierarchical phrase-based SMT (Koehn et al., 2003; Chiang, 2007) with traditional features, which achieves a similar performance to the state-of-the-art phrase-based decoder in Moses [1] (Koehn et al., 2007). The 4-gram language model is trained with target sentences from training set plus the Gigaword corpus [2]. Our phrase-based system is trained with MERT (Och, 2003). The other system is the encoder-decoder system (van Merriënboer et al., 2015) [3], based on which our KBSE is implemented.

We also combine KBSE with encoder-decoder system, by adding the knowledge-based semantic embedding to be another context vector. Hence, for the decoder there are two context vectors, one from the encoder and the other is generated by the **Semantic Grounding** part. We call this model Enc-Dec+KBSE.

For our proposed KBSE, the number of hidden units in both parts are 300. Embedding size of both source and target are 200. Adadelta (Zeiler, 2012) is leveraged as the optimizer for neural network training. The batch size is set to 128, and learning rate is initialized as 0.5. The model weights are randomly initialized from uniform distribution between [-0.005, 0.005].

Source Sentence	Semantic Tuples
我要 iPhone 移 版 的	*Category.cellphone* *Carrier.China_Mobile* *Brand.iPhone*
黑客帝国 是 一部 由 沃卓斯基 兄弟 执导 的 科幻 电影 ， 影片 语言 为 英语 。	*Name.The_Matrix* *Genre.science_fiction* *Director.Wachowski_bro* *Language.English*
Semantic Tuples	**Target Sentence**
Category.cellphone *Appearance.color.white* *Appearance.size.big_screen* *Network.4G_network*	I want a white 4G phone with a big screen .
Name.Pirates_of_Caribbean *Released.2003* *Country.America* *Starring.Johnny_Depp*	The Pirates of the Caribbean is a 2003 American film, starring Johnny Depp .

Table 3: Illustration of dataset structure in this paper. We show one example for both corpus in both part, respectively.

3.2 Dataset Details

To train our KBSE system, we only need two kinds of pairs: the pair of source sentence and semantic tuples to train our **Source Grounding**, the pair of semantic tuples and target sentence to train our **Target Generation**. Examples of our training data in the electric business and movie domains are shown in Table 3. To control the training process of KBSE, we randomly split 1000 instances from both corpus for validation set and another 1000 instances for test set. Our corpus of electric business domain consists of bilingual sentence pairs labeled with KB tuples manually [4], which is a collection of source-KB-target triplets. For the Movie domain, all the data are mined from web, thus we only have small part of source-KB-target triplets. In order to show the advantage of our proposed KBSE, we also mined source-KB pairs and KB-target pairs separately. It should be noted that, similar as the encoder-decoder method, bilingual data is needed for Enc-Dec+KBSE, thus with the added knowledge tuples, Enc-Dec+KBSE are trained with source-KB-target triplets.

[1] http://www.statmt.org/moses/

[2] https://catalog.ldc.upenn.edu/LDC2011T07

[3] The implementation is from https://github.com/mila-udem/blocks-examples

[4] Due to the coverage problem, knowledge bases of common domain (such as Freebase) are not used in this paper.

	Electric Business			Movie		
Model	**BLEU**	**HumanEval**	**Tuple F-score**	**BLEU**	**HumanEval**	**Tuple F-score**
SMT	54.30	78.6	-	42.08	51.4	-
Enc-Dec	60.31	90.8	-	44.27	65.8	-
KBSE	62.19	97.1	92.6	47.83	72.4	80.5
Enc-Dec + KBSE	64.52	97.9	-	46.35	74.6	-
KBSE upperbound	63.28	98.2	100	49.68	77.1	100

Table 4: The BLEU scores, human evaluation accuracy, tuple F-score for the proposed KBSE model and other benchmark models.

Our electric business corpus contains **50,169** source-KB-target triplets. For this data, we divide the intention of electric business into 11 classes, which are *Category, Function, Network, People, Price, Appearance, Carrier, Others, Performance, OS* and *Brand*. Each class above also has subclasses, for example *Category* class has subclass *computer* and *cellphone*, and *computer* class can be divided into *laptop, tablet_PC, desktop* and *AIO*.

Our movie corpus contains **44,826** source-KB-target triplets, together with **76,134** source-KB pairs and **85,923** KB-target pairs. The data is crawling from English Wikipedia [5] and the parallel web page in Chinese Wikipedia [6]. Simple rule method is used to extract sentences and KB pairs by matching the information in the infobox and the sentences in the page content. Since not all the entities from Chinese wikipedia has english name, we have an extra entity translator to translate them. For a fair comparison, this entity translator are also used in other systems. Due to the whole process is semi-automatic, there may be a few irregular results within. We divided the intention of movie data into 14 classes, which are *BasedOn, Budget, Country, Director, Distributor, Genre, Language, Name, Producer, Released, Starring, Studio, Theme* and *Writer*.

3.3 Evaluation

We use BLEU (Papineni et al., 2002) as the automatical evaluation matrix, significant testing is carried out using bootstrap re-sampling method (Koehn, 2004) with a 95% confidence level. As an addition, we also do human evaluation for all the comparison systems. Since the first part **Source Grounding** of our KBSE is separately trained, the F-score of KB tuples is also evaluated. Table 4 lists evaluation results for the electric business and movie data sets.

3.3.1 BLEU Evaluation

From Table 4, we can find that our proposed method can achieve much higher BLEU than SMT system, and we can also achieve 1.9 and 3.6 BLEU points improvement compared with the raw encoder-decoder system on both eletric business and movies data.

For the Enc-Dec+KBSE method, with the same training data on electric business domain, introducing knowledge semantic information can achieve about 4 BLEU points compared with the encoder-decoder and more than 2 BLEU points compared with our KBSE. Compared with encoder-decoder, Enc-Dec+KBSE method leverages the constrained semantic space, so that key semantic information can be extracted. Compared with KBSE, which relies on the knowledge base, Enc-Dec+KBSE method can reserve the information which is not formulated in the knowledge base, and also may fix errors generated in the source grounding part.

Since Enc-Dec+KBSE can only be trained with source-KB-target triplets, for the movie dataset, the performance is not as good as our KBSE, but still achieves a gain of more than 2 BLEU point compared with the raw Enc-Dec system. On movie data, our KBSE can achieve significant improvement compared with the models (SMT, Enc-Dec, Enc-Dec+KBSE) only using bilingual data. This shows the advantage of our proposed method, which is our model can leverage monolingual data to learn **Source Grounding** and **Target Generation** separately.

We also separately evaluate the **Source Grounding** and **Target Generation** parts. We evaluate the F-score of generated KB tuples

[5] https://en.wikipedia.org
[6] https://zh.wikipedia.org

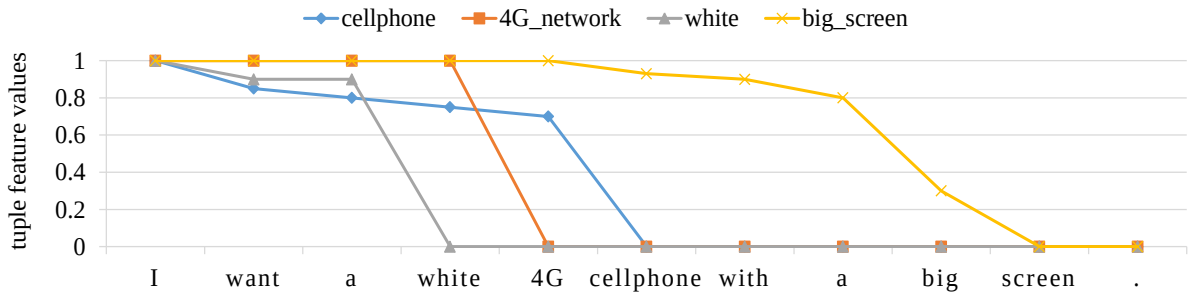


Figure 5: An example showing how the KB tuples control the tuple features flowing into the network via its learned semantic gates.

compared with the golden KB tuples. The result shows that our semantic grounding performance is quite high (92.6%), which means the first part can extract the semantic information in high coverage and accuracy. We evaluate the translation result by feeding the Target Generation network with human labeled KB tuples. The translation result (shown as KBSE upperbound in Table 4) with golden KB tuples can achieve about 1.1 and 1.8 BLEU scores improvement compared with KBSE with generated KB tuples in both dataset.

3.3.2 Human Evaluation

For the human evaluation, we do not need the whole sentence to be totally right. We focus on the key information, and if a translation is right by main information and grammar correction, we label it as correct translation, no matter how different of the translation compared with the reference on surface strings. Examples of correct and incorrect translations are shown in Table 5. As shown in Table 4, the human evaluation result shares the same trend as in BLEU evaluation. Our proposed method achieves the best results compared with SMT and raw encoder-decoder. In our method, important information are extracted and normalized by encoding the source sentence into the semantic space, and the correct translation of important information is key for human evaluation, thus our method can generate better translation.

3.4 Qualitative Analysis

In this section, we compare the translation result with baseline systems. Generally, since KB is introduced, our model is good at memorizing the key information of the source sentence. Also thanks to the strong learning ability of GRU, our model rarely make grammar mistakes. In many translations generated by traditional SMT, key information is lost. Encoder-Decoder system does much better, but some key information is also lost or even repetitively generated. Even for a long source sentence with a plenty of intentions, our model can generate the correct translation.

Target	I want a black Dell desktop.
Correct	I want a **Dell black** desktop. Could you please recommend me a black Dell desktop?
Incorrect	I want a **white** Dell desktop. I want a black Dell **laptop**. I want a black Dell desktop **desktop**.

Table 5: Some examples of which kind of sentence can be seen as a correct sentence and which will be seen as incorrect in the part of human evaluation.

To show the process of Target Generation, Figure 5 illustrates how the KB-tuples control the target sentence generation. Taking the semantic tuple *Appearance.color.white* as an example, the GRU keeps the feature value almost unchanged until the target word "white" is generated. Almost all the feature values drop from 1 to 0, when the corresponding words generated, except the tuple *Appearance.size.big_screen*. To express the meaning of this tuple, the decoding neural network should generate two words, "big" and "screen". When the sentence finished, all the feature values should be 0, with the constraint loss we introduced in Section 2.2.

Table 6 lists several translation example generated by our system, SMT system and the Encoder-Decoder system. The traditional SMT model sometimes generate same words or phrases several times, or some information is not translated. But our model rarely repeats or lose information. Besides, SMT often generate sentences unreadable, since some functional words are lost. But for KB-

Source	啊， 那个 有 大 屏幕 的 4G 手机 吗 ？ 要 白色 的 。
Reference	I want a 4G network cellphone with China Telecom supported.
KBSE	I need a white 4G cellphone with China Telecom supported.
Enc-Dec	I want a 3G cellphone with China Telecom.
SMT	Ah, that has a big screen, 4G network cellphone? give white.
Source	黑客帝国 是 一部 2003 年 由 沃卓斯基 兄弟 执导 的 电影， 里维斯 主演， 影片 语言 为 英语 。
Reference	The Matrix is a 2003 English film directed by Wachowski Brothers, starring Keanu Reeves.
KBSE	The Matrix is a 2003 English movie starring Keanu Reeves, directed by Wachowski Brothers.
Enc-Dec	The Matrix is a 2013 English movie directed by Wachowski, starring Johnny Depp.
SMT	The Matrix is directed by the Wachowski brothers film, and starring film language English.

Table 6: Examples of some translation results for our proposed KBSE system and the baseline systems.

SE, the target sentence is much easier to read. The Encoder-Decoder model learns the representation of the source sentence to a hidden vector, which is implicit and hard to tell whether the key information is kept. However KBSE learns the representation of the source sentence to a explicit tuple embedding, which contains domain specific information. So sometimes when encoder-decoder cannot memorize intention precisely, KBSE can do better.

3.5 Error Analysis

Our proposed KBSE relies on the knowledge base. To get the semantic vector of source sentence, our semantic space should be able to represent any necessary information in the sentence. For example, since our designed knowledge base do not have tuples for number of objects, some results of our KBSE generate the entities in wrong plurality form. Since our KBSE consists of two separate parts, the **Source Grounding** part and the **Target Generation** part, the errors generated in the first part cannot be corrected in the following process. As we mentioned in Section 3.3.1, combining KBSE with encoder-decoder can alleviate these two problems, by preserving information not captured and correct the errors generated in source grounding part.

4 Related Work

Unlike previous works using neural network to learn features for traditional log-linear model (Liu et al., 2013; Liu et al., 2014), Sutskever et al. (2014) introduced a general end-to-end approach based on an encoder-decoder framework. In order to compress the variable-sized source sentence into a fixed-length semantic vector, an encoder RNN reads the words in source sentence and generate a hidden state, based on which another decoder RNN is used to generate target sentence. Different from our work using a semantic space defined by knowledge base, the hidden state connecting the source and target RNNs is a vector of implicit and inexplicable real numbers.

Learning the semantic information from a sentence, which is also called semantic grounding, is widely used for question answering tasks (Liang et al., 2011; Berant et al., 2013; Bao et al., 2014; Berant and Liang, 2014). In (Yih et al., 2015), with a deep convolutional neural network (CNN), the question sentence is mapped into a query graph, based on which the answer is searched in knowledge base. In our paper, we use RNN to encode the sentence to do fair comparison with the encoder-decoder framework. We can try using CNN to replace RNN as the encoder in the future.

To generate a sentence from a semantic vector, Wen et al. (2015) proposed a LSTM-based natural language generator controlled by a semantic vector. The semantic vector memorizes what information should be generated for LSTM, and it varies along with the sentence generated. Our **Target Generation** part is similar with (Wen et al., 2015), while the semantic vector is not predefined, but generated by the **Source Grounding** part.

5 Conclusion and Future Work

In this paper, we propose a Knowledge Based Semantic Embedding method for machine translation, in which **Source Grounding** maps the source sentence into a semantic space, based on which **Target Generation** is used to generate the translation. Unlike the encoder-decoder neural network, in which the semantic space is implicit, the semantic space of KBSE is defined by a given knowledge base. Semantic vector generated by KBSE can extract and ground the key information, with the help of knowledge base, which is preserved in the translation sentence. Experiments are conducted on a electronic business and movie data sets,

and the results show that our proposed method can achieve significant improvement, compared with conventional phrase SMT system and the state-of-the-art encoder-decoder system.

In the future, we will conduct experiments on large corpus in different domains. We also want to introduce the attention method to leverage all the hidden states of the source sentence generated by recurrent neural network of **Source Grounding**.

Acknowledgement

We thank Dongdong Zhang, Junwei Bao, Zhirui Zhang, Shuangzhi Wu and Tao Ge for helpful discussions. This research was partly supported by National Natural Science Foundation of China (No.61333018 No.61370117) and Major National Social Science Fund of China (No.12&ZD227).

References

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976, Baltimore, Maryland, June. Association for Computational Linguistics.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 1415–1425.

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

David Chiang. 2007. Hierarchical phrase-based translation. *computational linguistics*, 33(2):201–228.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October. Association for Computational Linguistics.

Shi Feng, Shujie Liu, Mu Li, and Ming Zhou. 2016. Implicit distortion and fertility models for attention-based encoder-decoder NMT model. *CoRR*, abs/1601.03317.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn, 2004. *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, chapter Statistical Significance Tests for Machine Translation Evaluation.

Percy Liang, Michael I. Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *ACL*, pages 590–599.

Lemao Liu, Taro Watanabe, Eiichiro Sumita, and Tiejun Zhao. 2013. Additive neural networks for statistical machine translation. In *ACL (1)*, pages 791–801.

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. 2014. A recursive recurrent neural network for statistical machine translation.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Andriy Mnih and Geoffrey E Hinton. 2009. A scalable hierarchical distributed language model. In *Advances in neural information processing systems*, pages 1081–1088.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July. Association for Computational Linguistics.

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

Bart van Merriënboer, Dzmitry Bahdanau, Vincent Dumoulin, Dmitriy Serdyuk, David Warde-Farley, Jan Chorowski, and Yoshua Bengio. 2015. Blocks and fuel: Frameworks for deep learning. *arXiv preprint arXiv:1506.00619*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, September. Association for Computational Linguistics.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China, July. Association for Computational Linguistics.

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

One for All: Towards Language Independent Named Entity Linking

Avirup Sil and **Radu Florian**
IBM T. J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
avi@us.ibm.com, raduf@us.ibm.com

Abstract

Entity linking (EL) is the task of disambiguating mentions in text by associating them with entries in a predefined database of mentions (persons, organizations, etc). Most previous EL research has focused mainly on one language, English, with less attention being paid to other languages, such as Spanish or Chinese. In this paper, we introduce LIEL, a Language Independent Entity Linking system, which provides an EL framework which, once trained on one language, works remarkably well on a number of different languages without change. LIEL makes a joint global prediction over the entire document, employing a discriminative reranking framework with many domain and language-independent feature functions. Experiments on numerous benchmark datasets, show that the proposed system, once trained on one language, English, outperforms several state-of-the-art systems in English (by 4 points) and the trained model also works very well on Spanish (14 points better than a competitor system), demonstrating the viability of the approach.

1 Introduction

We live in a golden age of information, where we have access to vast amount of data in various forms: text, video and audio. Being able to analyze this data automatically, usually involves filling a relational database, which, in turn, requires the processing system to be able to identify actors across documents by assigning unique identifiers to them. Entity Linking (EL) is the task of mapping specific textual mentions of entities in a text document to an entry in a large catalog of entities, often called a knowledge base or KB, and is one of the major tasks in the Knowledge-Base Population track at the Text Analysis Conference (TAC) (Ji et al., 2014). The task also involves grouping together (clustering) NIL entities which do not have any target referents in the KB.

Previous work, pioneered by (Bunescu and Pasca, 2006; Cucerzan, 2007; Sil et al., 2012; Ratinov et al., 2011; Guo et al., 2013), have used Wikipedia as this target catalog of entities because of its wide coverage and its frequent updates made by the community. As with many NLP approaches, most of the previous EL research have focused on English, mainly because it has many NLP resources available, it is the most prevalent language on the web, and the fact that the English Wikipedia is the largest among all the Wikipedia datasets. However, there are plenty of web documents in other languages, such as Spanish (Fahrni et al., 2013; Ji et al., 2014), and Chinese (Cao et al., 2014; Shi et al., 2014), with a large number of speakers, and there is a need to be able to develop EL systems for these languages (and others!) quickly and inexpensively.

In this paper, we investigate the hypothesis that we can train an EL model that is entirely unlexicalized, by only allowing features that compute similarity between the text in the input document and the text/information in the KB. For this purpose, we propose a novel approach to entity linking, which we call Language Independent Entity Linking (henceforth LIEL). We test this hypothesis by applying the English-trained system on Spanish and Chinese datasets, with great success.

This paper has three novel contributions: 1) extending a powerful inference algorithm for global entity linking, built using similarity measures, corpus statistics, along with knowledge base statis-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2255–2264,
Berlin, Germany, August 7-12, 2016.

tics, 2) integrates many language-agnostic and domain independent features in an exponential framework, and 3) provide empirical evidence on a large variety of popular benchmark datasets that the resulting model outperforms or matches the best published results, and, most importantly, the trained model transfers well across languages, outperforming the state-of-the-art (SOTA) in Spanish and matching it in Chinese.

We organize the paper as follows: the next section motivates the problem and discusses the language-independent model along with the features. Section 3 describes our experiments and comparison with the state-of-the-art. Section 4 illustrates the related previous work and Section 5 concludes.

2 Problem Formulation

2.1 Motivation for Language Independence

Our strategy builds an un-lexicalized EL system by training it on labeled data, which consists of pairs of mentions in text and entries in a database extracted from a Wikipedia collection in English. Unlike traditional EL, however, the purpose here is to be able to perform entity linking with respect to any Wikipedia collection. Thus the strategy must take care to build a model that can transfer its learned model to a new Wikipedia collection, without change.

At a first glance, the problem seems very challenging - learning how to discriminate `Lincoln,_Nebraska` and `Abraham_Lincoln` [1], the former US President, seemingly bears little resemblance to disambiguating between different Spanish person entities named "Ali Quimico". The crux of the problem lies in the fact that Wikipedia-driven features are language-specific: for instance, counting how many times the category `2010 Deaths` appears in the context of an entity is highly useful in the English EL task, but not directly useful for Spanish EL. Also, learning vocabulary-specific information like the list of "deaths", "presidents", etc. is very useful for disambiguating person entities like "Lincoln" in English, but the same model, most likely, will not work for mentions like "李娜" in a Chinese document which might either refer to the famous athlete 李娜 (网球运动员) or the singer 李娜 (歌手).

[1] Teletype font denotes Wikipedia titles and categories.

Practically we assume the existence of a knowledge base that defines the space of entities we want to disambiguate against, where each entry contains a document with the entity; Wikipedia is a standard example for this[2]. If there are other properties associated with the entries, such as categories, in-links, out-links, redirects, etc., the system can make use of them, but they are theoretically not required. The task is defined as: given a mention m in a document d, find the entry e in the knowledge base that m maps to.

We expand on the architecture described in (Sil and Yates, 2013) (henceforth NEREL), because of the flexibility provided by the feature-based exponential framework which results in an English SOTA EL system. However, we design all our features in such a way that they measure the similarity between the context where the mention m appears in d and the entries in the knowledge base. For example, instead of counting how often the category `2010 Deaths` [3] appears in the context around an entity mention, we create a feature function such as **CATEGORY FREQUENCY**(m, e), which counts how often any category of entity referent e appears in the context of mention m. For entities like `Lincoln,_Nebraska` in the English EL, **CATEGORY FREQUENCY** will add together counts for appearances of categories like `Cities in Lancaster County, Nebraska` and `Lincoln metropolitan area`, among other categories. At the same time, in the Spanish EL domain, **CATEGORY FREQUENCY** will add together counts for `Políticos de Irak` and `Militares de Irak` for the KB id corresponding to "Ali Quimico". This feature is well-defined in both domains, and larger values of the feature indicate a better match between m and e. As mentioned earlier, it is our hypothesis, that the parameters trained for such features on one language (English, in our case) can be successfully used, without retraining, on other languages, namely Spanish and Chinese.

While training, the system will take as input a knowledge base in source language S, KB_S (extracted from Wikipedia) and a set of training examples (m_i, e_i, g_i), where instances m_i are mentions in a document of language S, e_i are entity links, $e_i \in KB_S$, and g_i are Boolean val-

[2] We will assume, without loss of generality, that the knowledge base is derived from Wikipedia.

[3] Or a specific Freebase type.

ues indicating the gold-standard match / mismatch between m_i and e_i. During decoding, given language T[4], the system must classify examples (m_j, e_j) drawn from a *target* language T and knowledge-base KB_T.

2.2 LIEL: Training and Inference

Our language-independent system consists of two components: 1. extracting mentions of named-entities from documents and 2. linking the detected mentions to a knowledge base, which in our case is Wikipedia (focus of this paper). We run the IBM Statistical Information and Relation Extraction (SIRE) [5] system which is a toolkit that performs mention detection, relation extraction, coreference resolution, *etc.* We use the system to extract mentions and perform coreference resolution: in particular, we use the CRF model of IBM SIRE for mention detection and a maximum entropy clustering algorithm for coreference resolution. The system identifies a set of 53 entity types. To improve the mention detection and resolution, case restoration is performed on the input data. Case restoration is helpful to improve the mention detection system's performance, especially for discussion forum data. Obviously, this processing step is language-dependent, as the information extraction system is - but we want to emphasize that the entity linking system is language independent.

In the EL step, we perform a full document entity disambiguation inference, described as follows. Given a document d, and a selected mention $m \in d$, our goal is to identify its label $\hat{e}$ that maximizes

$$\begin{aligned} \hat{e} &= P(e|m,d) \qquad (1) \\ &= \arg\max_{e:m} \sum_{k, m \in m_1^k, e_1^k} P\left(m_1^k|m,d\right) P\left(e_1^k|m_1^k,d\right) \end{aligned}$$

where m_1^k are mentions found in document d, and e_1^k are some label assignment. In effect, we are looking for the best mention labeling of the entire document m_1^k (that contains m) and a label to these mentions that would maximize the information extracted from the entire document. Since direct inference on Equation 1 is hard, if not intractable, we are going to select the most likely mention assignment instead (as found by an information extraction system): we will only consider the detected mentions $(m_1, \ldots, m_k)$, and other optional information that can be extracted from the document, such as links l, categories r, etc. The goal becomes identifying the set of labels $(e_1, \ldots, e_k)$ that maximize

$$P\left(e_1^k|m_1^k,d\right) \qquad (2)$$

Since searching over all possible sets of (mention, entity)-pairs for a document is still intractable for reasonable large values of k, typical approaches to EL make simplifying assumption on how to compute the probability in Equation 2. Several full-document EL approaches have investigated generating up to N global tuples of entity ids $(e_1, \ldots, e_k)$, and then build a model to rank these tuples of entity ids (Bunescu and Pasca, 2006; Cucerzan, 2007). However, Ratinov *et al.* (Ratinov et al., 2011) argue that this type of global model provides a relatively small improvement over the purely-local approach (where $P\left(e_1^k|m_1^k,d\right) = \prod_i P\left(e_i|m_i,d\right)$). In this paper, we follow an approach which combines both of these strategies.

Following the recent success of (Sil and Yates, 2013), we partition the full set of extracted mentions, $(m_i)_{i=1,n}^{-}$ of the input document d into smaller subsets of mentions which appear near one another: we consider two mentions that are closer then 4 words to be in the same connected component, then we take the transitive closure of this relation to partition the mention set. We refer to these sets as the *connected components* of d, or $CC(d)$. We perform classification over the set of entity-mention tuples $T(C) = \left\{\left(e_{i_1}, \ldots, e_{i_{n_C}}|m_{i_1}, \ldots, m_{i_{n_C}}\right)|e_{i_j} \in KB, \forall j\right\}$ [6] that are formed using candidate entities within the same connected component $C \in CC(d)$. Consider this small snippet of text:

"... Home Depot CEO Nardelli quits ..."

In this example text, the phrase "Home Depot CEO Nardelli" would constitute a connected component. Two of the entity-mention tuples for this connected component would be: (`Home_Depot`, `Robert_Nardelli` |"Home Depot", "Nardelli") and (`Home_Depot`, `Steve_Nardelli` | "Home Depot","Nardelli").

[4] Language prediction can be done relatively accurately, given a document; however, in this paper, we focus on the EL task, so we assume we know the identity of the target language T.

[5] The IBM SIRE system can be currently accessed at : http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/relationship-extraction.html

[6] For simplicity, we denote by $(e|m)$ the tuple (e, m), written like that to capture the fact that m is fixed, while e is predicted.

2.2.1 Collective Classification Model

To estimate $P(t|d, C)$, the probability of an entity-mention tuple t for a given connected component $C \in CC(d)$, LIEL uses a maximum-entropy model:

$$P(t|d, C) = \frac{\exp(\mathbf{w} \cdot \mathbf{f}(t, d, C))}{\sum_{t' \in T(C)} \exp(\mathbf{w} \cdot \mathbf{f}(t', d, C))} \quad (3)$$

where $\mathbf{f}(t, d, C)$ is a feature vector associated with t, d, and C, and w is a weight vector. For training, we use L2-regularized conditional log likelihood (CLL) as the objective

$$CLL(G, \mathbf{w}) = \sum_{(t,d,C) \in G} \log P(t|d, C, \mathbf{w}) - \sigma \|\mathbf{w}\|_2^2 \quad (4)$$

where G is the gold-standard training data, consisting of pairs (t, d, C), where t is the correct tuple of entities and mentions for connected component C in document d, and σ is a regularization parameter. Given that the function 4 is convex, we use LBFGS (Liu and Nocedal, 1989) to find the globally optimal parameter settings over the training data.

2.3 Extracting potential target entities

From the dump of our Wikipedia data, we extract all the mentions that can refer to Wikipedia titles, and construct a set of disambiguation candidates for each mention (which are basically the hyperlinks in Wikipedia). This is, hence, an anchor-title index that maps each distinct hyperlink anchor-text to its corresponding Wikipedia titles and also stores their relative popularity score. For example, the anchor text (or mention) "Titanic" is used in Wikipedia to refer both to the ship or to the movie. To retrieve the disambiguation candidates e_i for a given mention m_i, we query the anchor-title index that we constructed and use lexical sub-word matching. e_i is taken to be the set of titles (or entities, in the case of EL) most frequently linked to with anchor text m_i in Wikipedia. We use only the top 40 most frequent Wikipedia candidates for the anchor text for computational efficiency purposes for most of our experiments. We call this step "Fast Search" since it produces a bunch of candidate links by just looking up an index.

2.3.1 Decoding

At decoding time, given a document d, we identify its connected components $CC(d)$ and run inference on each component C containing the desired input mention m. To further reduce the run time, for each mention $m_j \in C$, we obtain the set of potential labels e_j using the algorithm described in Section 2.3, and then exhaustively find the pair that maximizes equation 3. For each candidate link, we also add a NIL candidate to fast match to let the system link mentions to ids not in a KB.

2.4 Language-Independent Feature Functions

LIEL makes use of new as well as well-established features in the EL literature. However, we make sure to use only non-lexical features. The local and global feature functions computed from this extracted information are described below.

Generically, we have two types of basic features: one that takes as input a KB entry e, the mention m and its document and a second type that scores two KB entries, e_1 and e_2. When computing the probability in Equation 3, where we consider a set of KB entries t[7], we either sum or apply a boolean AND operator (in case of boolean features) among all entities $e \in t$, while the entity-entity functions are summed/and'ed for consecutive entities in t. We describe the features in these terms, for simplicity.

2.4.1 Mention-Entity Pair Features

Text-based Features: We assume the existence of a document with most entries in the KB, and the system uses similarity between the input document and these KB documents. The basic intuition behind these features, inspired by Ratinov *et al.*(2011), is that a mention $m \in d$ is more likely to refer to entity e if its KB page, $W(e)$, has high textual similarity to input document d. Let $Text(W(e))$ be the vector space model associated with $W(e)$, $Top(W(e))$ be the vector of the top most frequently occurring words (excluding stop-words) from $W(e)$, and $Context(W(e))$ be the vector space of the 100 word window around the first occurrence of m in $W(e)$. Similarly, we create vector space models $Text(m)$ and $Context(m)$. We then use cosine similarity over these vector space models as features:
i. cosine($Text(W(e)), Text(m)$),
ii. cosine($Text(W(e)), Context(m)$),
iii. cosine($Context(W(e)), Text(m)$),
iv. cosine($Context(W(e)), Context(m)$),
v. cosine($Top(W(e)), Text(m)$),

[7]Recall that the probability is computed for all the entity assignments for mentions in a clique.

vi. cosine $(Top\,(W\,(e))\,, Context\,(m))$.

KB Link Properties: LIEL can make use of existing relations in the KB, such as inlinks, outlinks, redirects, and categories. Practically, for each such relation l, a KB entry e has an associated set of strings $I(l, e)$[8]; given a mention-side set M (either $Text(m)$ or $Context(m)$), LIEL computes **FREQUENCY** feature functions for the names of the Categories, Inlinks, Outlinks and Redirects, we compute

$$f(e, m, d) = |I(l, e) \cap M|$$

Title Features: LIEL also contains a number of features that make use of the Wikipedia title of the entity links in t (remember t = entity mention tuples and not a Wikipedia title) :

- **NIL FREQUENCY**: Computes the frequency of entities that link to NIL
- **EXACT MATCH FREQUENCY**: returns 1 if the surface form of m is a redirect for e;
- **MATCH ALL**: returns true if m matches *exactly* the title of e;
- **MATCH ACRONYM**: returns true if m is an acronym for a redirect of e;
- **LINK PRIOR**: the prior link probability $P(e|m)$, computed from anchor-title pairs in KB (described in Section 2.3).

2.4.2 Entity-Entity Pair Features

Coherence Features: To better model consecutive entity assignments, LIEL computes a coherence feature function called **OUTLINK OVERLAP**. For every consecutive pair of entities (e_1, e_2) that belongs to mentions in t, the feature computes $\text{Jaccard}(Out(e_1), Out(e_2))$, where $Out(e)$ denotes the Outlinks of e. Similarly, we also compute **INLINK OVERLAP**.

LIEL also uses categories in Wikipedia which exist in all languages. The first feature **ENTITY CATEGORY PMI**, inspired by Sil and Yates (2013), make use of Wikipedia's category information system to find patterns of entities that commonly appear next to one another. Let $\mathcal{C}(e)$ be the set of Wikipedia categories for entity e. We manually inspect and remove a handful of common Wikipedia categories based on threshold frequency on our training data, which are associated with almost every entity in text, like `Living People` etc., since they have lower discriminating power. These are analogous to all WP languages. From the training data, the system first computes point-wise mutual information (PMI) (Turney, 2002) scores for the Wikipedia categories of pairs of entities, (e_1, e_2):

$$PMI(\mathcal{C}(e_1), \mathcal{C}(e_2)) = \frac{\sum_{j=1}^{n_{t_\mathcal{C}}-1} \mathbf{1}[\mathcal{C}(e_1) = \mathcal{C}(e_{i_j}) \wedge \mathcal{C}(e_2) = \mathcal{C}(e_{i_{j+1}})]}{\sum_j \mathbf{1}[\mathcal{C}(e_1) = \mathcal{C}(e_{i_j})] \times \sum_j \mathbf{1}[\mathcal{C}(e_2) = \mathcal{C}(e_{i_j})]}$$

- **ENTITY CATEGORY PMI** adds these PMI scores up for every consecutive (e_1, e_2) pair in t.
- **CATEGORICAL RELATION FREQUENCY** We would like to boost consecutive entity assignments that have been seen in the training data. For instance, for the text "England captain Broad fined for..", we wish to encourage the tuple that links "England" to the entity id of the team name `England cricket team`, and "Broad" to the entity id of the person `Stuart Broad`. Wikipedia contains a relation displayed by the category called `English_cricketers` that indicates that `Stuart Broad` is a team member of `England cricket team`, and counts the number of such relations between every consecutive pair of entities in $(e, e') \in t$.
- **TITLE CO-OCCURRENCE FREQUENCY** feature computes for every pair of consecutive entities $(e, e') \in t$, the number of times that e' appears as a link in the Wikipedia page for e, and *vice versa* (similar to (Cucerzan, 2007). It adds these counts up to get a single number for t.

3 Experiments

We evaluate LIEL's capability by testing against several state-of-the-art EL systems on English, then apply the English-trained system to Spanish and Chinese EL tasks to test its language transcendability.

3.1 Datasets

English: The 3 benchmark datasets for the English EL task are: **i)** ACE (Ratinov et al., 2011), **ii)** MSNBC (Cucerzan, 2007) and **iii)** TAC 2014 (Ji et

[8]For instance, redirect strings for "Obama" are "Barack Obama", "Barack Obama Jr." and "Barack Hussein Obama".

Name	$\|M\|$	In KB	Not in KB
ACE	257	100%	0
MSNBC	747	90%	10%
TAC_En14	5234	54%	46%
TAC_Es13	2117	62%	38%
TAC_Es14	2057	72%	28%
TAC_Zh13	2155	57%	43%
WikiTrain	158715	100%	0%

Table 1: Data statistics: number of mention queries, % of mention queries that have their referents present in the Wikipedia/KB, and % of mention queries that have no referents in Wikipedia/KB as per our datasets. En=English, Es=Spanish and Zh=Chinese for the evaluation data for TAC for the years 2013 and 2014.

al., 2014)[9], which contain data from diverse genre like discussion forum, blogs and news. Table 1 provides key statistics on these datasets. In the TAC[10] evaluation setting, EL systems are given as input a document and a query mention with its offsets in the input document. As the output, systems need to predict the KB id of the input query mention if it exists in the KB or NIL if it does not. Further, they need to cluster the mentions which contain the same NIL ids across queries.

The training dataset, WikiTrain, consists of 10,000 random Wikipedia pages, where all of the phrases that link to other Wikipedia articles are treated as mentions, and the target Wikipedia page is the label. The dataset was made available by Ratinov *et al.* and (Sil and Yates, 2013), added Freebase to Wikipedia mappings resulting in 158,715 labeled mentions with an average of 12.62 candidates per mention. The total number of unique mentions in the data set is 77,230 with a total of 974,381 candidate entities and 643,810 unique candidate entities. The Wikipedia dump that we used as our knowledge-base for English, Spanish and Chinese is the April 2014 dump. The TAC dataset involves the TAC KB which is a dump of May 2008 of English Wikipedia. LIEL links entities to the Wikipedia 2014 dump and uses the redirect information to link back to the TAC KB.

Spanish: We evaluate LIEL on both the 2013 and 2014 benchmark datasets of the TAC Spanish evaluation.

Chinese: We test LIEL on the TAC 2013 Chinese dataset.

3.2 Evaluation Metric

We follow standard measures used in the literature for the entity linking task. To evaluate EL accuracy on ACE and MSNBC, we report on a Bag-of-Titles (BOT) F1 evaluation as introduced by (Milne and Witten, 2008; Ratinov et al., 2011). In BOT-F1, we compare the set of Wikipedia titles output for a document with the gold set of titles for that document (ignoring duplicates), and compute standard precision, recall, and F1 measures. On the TAC dataset, we use standard metrics B^3+ variant of precision, recall and F1. On these datasets, the $B^3 + F1$ metric includes the clustering score for the NIL entities, and hence systems that only perform binary NIL prediction would be heavily penalized[11].

3.3 Comparison with the State-of-the-art

To follow the guidelines for the TAC NIST evaluation, we anonymize participant system names as System 1 through 9. Interested readers may look at their system description and scores in (Ji et al., 2014; Fahrni et al., 2013; Miao et al., 2013; Mayfield, 2013; Merhav et al., 2013). Out of these systems, System 1 and System 7 obtained the top score in Spanish and Chinese EL evaluation at TAC 2013 and hence can be treated as the current state-of-the-art for the respective EL tasks. We also compare LIEL with some traditional "wikifiers" like MW08 (Milne and Witten, 2008) and UIUC (Cheng and Roth, 2013) and also NEREL (Sil and Yates, 2013) which is the system which LIEL resembles the most.

3.4 Parameter Settings

LIEL has two tuning parameters: σ, the regularization weight; and the number of candidate links per mention we select from the Wikipedia dump. We set the value of σ by trying five possible values in the range [0.1, 10] on held-out data (the TAC 2009 data). We found $\sigma = 0.5$ to work best for our experiments. We chose to select a maximum of 40 candidate entities from Wikipedia for each candidate mention (or fewer if the dump had fewer than 40 links with nonzero probability).

[9]This is the traditional Entity Linking (EL) task and not Entity Discovery and Linking (EDL), since we are comparing the linking capability in this paper.

[10]For more details on TAC see http://nlp.cs.rpi.edu/kbp/2014/index.html

[11]For more details on the scoring metric used for TAC EL see: http://nlp.cs.rpi.edu/kbp/2014/scoring.html

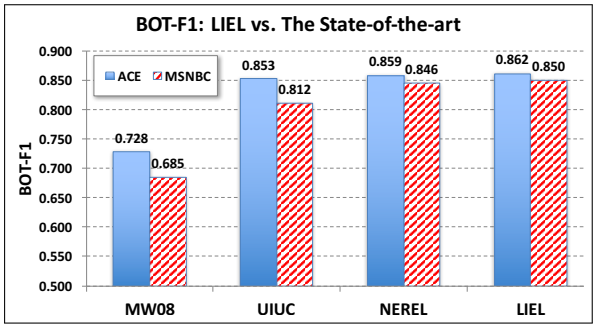


Figure 1: LIEL outperforms all its competitors on both ACE and MSNBC.

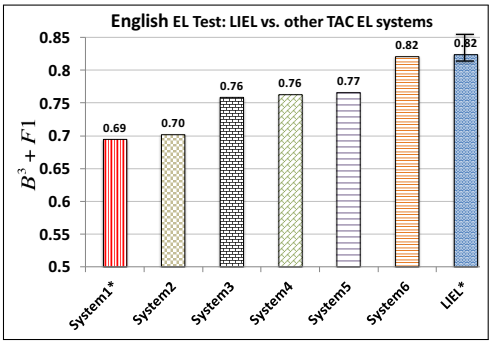


Figure 2: Comparison of several state-of-the-art English EL systems along with LIEL on the latest TAC 2014 dataset and LIEL obtains the best score. * indicates systems that perform multilingual EL.

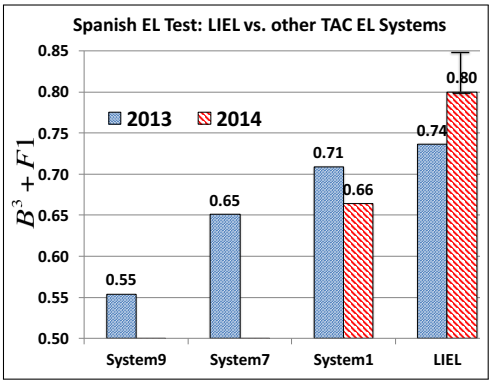


Figure 3: System performance on the TAC 2013 and 2014 Spanish datasets are shown. LIEL outperforms all the systems in terms of overall F1 score.

3.5 Results

English: Figure 1 compares LIEL with previously reported results by MW08, UIUC and NEREL on the ACE and MSNBC datasets in (Cheng and Roth, 2013; Sil and Yates, 2013). LIEL achieves an F1 score of 86.2 on ACE and 85.0 on MSNBC, clearly outperforming the others *e.g.* 3.8% absolute value higher than UIUC on MSNBC. We believe that LIEL's strong model comprising relational information (coherence features from large corpus statistics), textual and title lets it outperform UIUC and MW08 where the former uses relational information and the latter a naive version of LIEL's coherence features. Comparison with NEREL is slightly unfair (though we outperform them marginally) since they use both Freebase and Wikipedia as their KB whereas we are comparing with systems which only use Wikipedia as their KB.

To test the robustness of LIEL on a diverse genre of data, we also compare it with some of the other state-of-the-art systems on the latest benchmark TAC 2014 dataset. Figure 2 shows our results when compared with the top systems in the evaluation. Encouragingly, LIEL's performance is tied with the top performer, System 6, and outperforms all the other top participants from this challenging annual evaluation. Note that LIEL obtains 0.13 points more than System 1, the only other multi-lingual EL system and, in that sense, LIEL's major competitor. Several other factors are evident from the results: System 1 and 2 are statistically tied and so are System 3, 4 and 5. We also show the bootstrapped percentile confidence intervals (Singh and Xie, 2008) for LIEL which are [0.813, 0.841]: (we do not have access to the other competing systems).

3.5.1 Foreign Language Experiments

Note that LIEL was *trained only* on the English Wikitrain dataset (Section 3.1), and then applied, *unchanged*, to all the evaluation datasets across languages and domains described in Section 3.1. Hence, it is the same instance of the model for all languages. As we will observe, this one system consistently outperforms the state of the art, even though it is using exactly the same trained model across the datasets. We consider this to be the take-away message of this paper.

Spanish: LIEL obtains a $B^3 + F1$ score of 0.736 on the TAC 2013 dataset and clearly outperforms the SOTA, System 1, which obtains 0.709 as shown in Figure 3 and considerably higher than the other participating systems. We could only obtain the results for Systems 9 and 7 on 2013. On the 2014 evaluation dataset, LIEL obtains a higher gain of 0.136 points (precision of 0.814 and recall of 0.787) over its major competitor System 1, showing the power of its language-independent model.

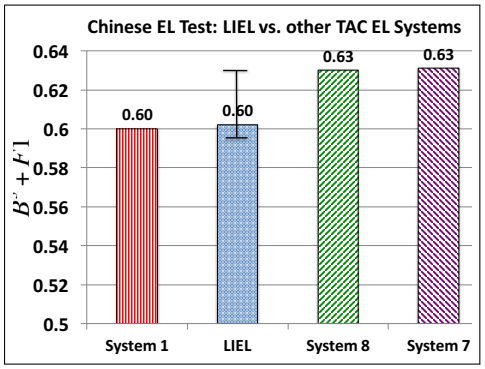


Figure 4: LIEL achieves competitive performance in Chinese EL further proving its robustness to multilingual data.

Chinese: Figure 4 shows the results of LIEL's performance on the Chinese benchmark dataset compared to the state-of-the-art. Systems 7 and 8 obtains almost similar scores. We observe that LIEL is tied with System 1 and achieves competitive performance compared to Systems 7 and 8 (note that LIEL has a confidence interval of [0.597, 0.632]) which requires labeled Chinese TAC data to be trained on and the same model does not work for other languages. Emphasizing again: LIEL is trained only *once*, on English, and tested on Chinese unchanged.

3.5.2 Error Analysis

While we see LIEL's strong multi-lingual empirical results, it is important to note some of the areas which confuses the system. Firstly, a major source of error which affects LIEL's performance is due to coreference resolution *e.g.* from the text "*Beltran Leyva, also known as "The Bearded One," is ...*", TAC's mention query asks the systems to provide the disambiguation for *The Bearded One*. LIEL predicts that the *The Bearded One* refers to the entity `Richard Branson`, which is the most common entity in Wikipedia that refers to that nickname (based on our dump), while, clearly, the correct entity should have been *Beltran Levya*. We believe that this type of an error can be handled by performing joint EL and coreference resolution, which is a promising future research area for LIEL.

Contextual information can also hurt system performance *e.g.* from the text, ".. *dijo Alex Sánchez , analista..*", LIEL predicts the Wikipedia title `Alex Sánchez (outfielder)` for the mention *Alex Sánchez* since the document talks about sports and player names. The query mention was actually referring to a journalist, not in the KB, and hence a NIL. Handling sparse entities, similar to this, are also an important future direction.

4 Related Work

Entity linking has been introduced and actively developed under the NIST-organized Text Analysis Conference, specifically the Knowledge Base Population track. The top performing English EL system in the TAC evaluation has been the MS_MLI system (Cucerzan and Sil, 2013), which has obtained the top score in TAC evaluation in the past 4 years (2011 through 2014): the system links all mentions in a document simultaneously, with the constraint that their resolved links should be globally consistent on the category level as much as possible. Since global disambiguation can be expensive, (Milne and Witten, 2008) uses the set of unambiguous mentions in the text surrounding a mention to define the mention's context, and uses the Normalized Google Distance (Cilibrasi and Vitanyi, 2007) to compute the similarity between this context and the candidate Wikipedia entry. The UIUC system, (Cheng and Roth, 2013), another state-of-the-art EL system, which is an extension of (Ratinov et al., 2011), adds relational inference for wikification. NEREL (Sil and Yates, 2013) is a powerful joint entity extraction and linking system. However, by construction their model is not language-independent due to the heavy reliance on type systems of structured knowledgebases like Freebase. It also makes use of lexical features from Wikipedia as their model performs joint entity extraction and disambiguation. Some of the other systems which use a graph based algorithm such as partitioning are LCC, NYU (Ji et al., 2014) and HITS (Fahrni et al., 2013) which obtained competitive score in the TAC evaluations. Among all these systems, only the HITS system has ventured beyond English and has obtained the top score in Spanish EL evaluation at TAC 2013. It is the only multilingual EL system in the literature which performs reliably well across a series of languages and benchmark datasets. Recently, (Wang et al., 2015) show a new domain and language-independent EL system but they make use of translation tables for non-English (Chinese) EL; thereby not making the system entirely language-independent. Empirically their performance comes close to System 1 which LIEL outperforms. The BASIS system (Merhav et al.,

2013), is the state-of-the-art for Chinese EL as it obtained the top score in TAC 2013. The FUJITSU system (Miao et al., 2013) obtained similar scores. It is worth noting that these systems, unlike LIEL, are heavily language dependent, *e.g.* performing lexicon specific information extraction, using inter-language links to map between the languages or training using labeled Chinese data.

In more specialized domains, Dai *et al.* (2011) employed a Markov logic network for building an EL system with good results in a bio-medical domain; it would be interesting to find out how their techniques might extended to other languages/corpora. Phan *et al.* (2008) utilize topic models derived from Wikipedia to help classify short text segment, while Guo *et al.* (2013) investigate methods for disambiguating entities in tweets. Neither of these methods do show how to transfer the EL system developed for short texts to different languages, if at all.

The large majority of entity linking research outside of TAC involves a closely related task - wikification (Bunescu and Pasca, 2006; Cucerzan, 2007; Ratinov et al., 2011; Guo et al., 2013), and has been mainly performed on English datasets, for obvious reasons (data, tools availability). These systems usually achieve high accuracy on the language they are trained on. Multilingual studies, e.g. (McNamee et al., 2011), use a large number of pipelines and complex statistical machine translation tools to first translate the original document contexts into English equivalents and transform the cross-lingual EL task into a monolingual EL one. The performance of the entity linking system is highly dependent on the existence and potential of the statistical machine translation system in the given pair of languages.

5 Conclusion

In this paper we discussed a new strategy for multilingual entity linking that, once trained on one language source with accompanying knowledge base, performs without adaptation in multiple target languages. Our proposed system, LIEL is trained on the English Wikipedia corpus, after building its own knowledge-base by exploiting the rich information present in Wikipedia. One of the main characteristics of the system is that it makes effective use of features that are built exclusively around computing similarity between the text/context of the mention and the document text of the candidate entity, allowing it to transcend language and perform inference on a completely new language or domain, without change or adaptation.

The system displays a robust and strong empirical evidence by not only outperforming all state-of-the-art English EL systems, but also achieving very good performance on multiple Spanish and Chinese entity linking benchmark datasets, and it does so without the need to switch, retrain, or even translate, a major differentiating factor from the existing multi-lingual EL systems out there.

Acknowledgments

We would like to thank the anonymous reviewers for their suggestions. We also thank Salim Roukos, Georgiana Dinu and Vittorio Castelli for their helpful comments. This work was funded under DARPA HR0011-12-C-0015 (BOLT). The views and findings in this paper are those of the authors and are not endorsed by DARPA.

References

R. Bunescu and M. Pasca. 2006. Using encyclopedic knowledge for named entity disambiguation. In *EACL*.

Ziqiang Cao, Sujian Li, and Heng Ji. 2014. Joint learning of chinese words, terms and keywords. In *EMNLP*.

X. Cheng and D. Roth. 2013. Relational inference for wikification. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

R.L. Cilibrasi and P.M.B. Vitanyi. 2007. The google similarity distance. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):370–383.

Silviu Cucerzan and Avirup Sil. 2013. The MSR Systems for Entity Linking and Temporal Slot Filling at TAC 2013. In *Text Analysis Conference*.

Silviu Cucerzan. 2007. Large-scale named entity disambiguation based on wikipedia data. In *EMNLP-CoNLL*, pages 708–716.

Hong-Jie Dai, Richard Tzong-Han Tsai, Wen-Lian Hsu, et al. 2011. Entity disambiguation using a markov-logic network. In *IJCNLP*.

Angela Fahrni, Benjamin Heinzerling, Thierry Göckel, and Michael Strube. 2013. Hits monolingual and cross-lingual entity linking system at TAC 2013. In *Text Analysis Conference*.

Stephen Guo, Ming-Wei Chang, and Emre Kıcıman. 2013. To link or not to link? a study on end-to-end tweet entity linking. In *NAACL*.

Heng Ji, HT Dang, J Nothman, and B Hachey. 2014. Overview of tac-kbp2014 entity discovery and linking tasks. In *Proc. Text Analysis Conference (TAC2014)*.

D.C. Liu and J. Nocedal. 1989. On the limited memory method for large scale optimization. *Mathematical Programming B*, 45(3):503–528.

James Mayfield. 2013. Overview of the kbp 2013 entity linking track.

Paul McNamee, James Mayfield, Douglas W Oard, Tan Xu, Ke Wu, Veselin Stoyanov, and David Doermann. 2011. Cross-language entity linking in maryland during a hurricane. In *Text Analysis Conference*.

Yuval Merhav, Joel Barry, James Clarke, David Murgatroyd, and One Alewife Center. 2013. Basis technology at tac 2013 entity linking.

Qingliang Miao, Ruiyu Fang, Yao Meng, and Shu Zhang. 2013. Frdc's cross-lingual entity linking system at tac 2013.

David Milne and Ian H. Witten. 2008. Learning to link with wikipedia. In *CIKM*.

Xuan-Hieu Phan, Le-Minh Nguyen, and Susumu Horiguchi. 2008. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web*.

L. Ratinov, D. Roth, D. Downey, and M. Anderson. 2011. Local and global algorithms for disambiguation to wikipedia. In *Proc. of the Annual Meeting of the Association of Computational Linguistics (ACL)*.

Xing Shi, Kevin Knight, and Heng Ji. 2014. How to speak a language without knowing it. In *ACL*.

Avirup Sil and Alexander Yates. 2013. Re-ranking for Joint Named-Entity Recognition and Linking. In *CIKM*.

Avirup Sil, Ernest Cronin, Penghai Nie, Yinfei Yang, Ana-Maria Popescu, and Alexander Yates. 2012. Linking Named Entities to Any Database. In *EMNLP-CoNLL*.

Kesar Singh and Minge Xie. 2008. Bootstrap: a statistical method.

P. D. Turney. 2002. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. In *Procs. of ACL*, pages 417–424.

Han Wang, Jin Guang Zheng, Xiaogang Ma, Peter Fox, and Heng Ji. 2015. Language and domain independent entity linking with quantified collective validation. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP2015)*.

On Approximately Searching for Similar Word Embeddings

Kohei Sugawara **Hayato Kobayashi** **Masajiro Iwasaki**
Yahoo Japan Corporation
1-3 Kioicho, Chiyoda-ku, Tokyo 102-8282, Japan
{ksugawar, hakobaya, miwasaki}@yahoo-corp.jp

Abstract

We discuss an approximate similarity search for word embeddings, which is an operation to approximately find embeddings close to a given vector. We compared several metric-based search algorithms with hash-, tree-, and graph-based indexing from different aspects. Our experimental results showed that a graph-based indexing exhibits robust performance and additionally provided useful information, e.g., vector normalization achieves an efficient search with cosine similarity.

1 Introduction

An embedding or distributed representation of a word is a real-valued vector that represents its "meaning" on the basis of *distributional semantics*, where the meaning of a word is determined by its context or surrounding words. For a given meaning space, searching for similar embeddings is one of the most basic operations in natural language processing and can be applied to various applications, e.g., extracting synonyms, inferring the meanings of polysemous words, aligning words in two sentences in different languages, solving analogical reasoning questions, and searching for documents related to a query.

In this paper, we address how to quickly and accurately find similar embeddings in a continuous space for such applications. This is important from a practical standpoint, e.g., when we want to develop a real-time query expansion system on a search engine on the basis of an embedding similarity. A key difference from the existing work is that embeddings are not high-dimensional sparse (traditional count) vectors, but (relatively) low-dimensional dense vectors. We therefore need to use approximate search methods instead of inverted-index-based methods (Zobel and Moffat, 2006). Three types of indexing are generally used in approximate similarity search: hash-, tree-, and graph-based indexing. Hash-based indexing is the most common in natural language processing due to its simplicity, while tree/graph-based indexing is preferred in image processing because of its performance. We compare several algorithms with these three indexing types and clarify which algorithm is most effective for similarity search for word embeddings from different aspects.

To the best of our knowledge, no other study has compared approximate similarity search methods focusing on neural word embeddings. Although one study has compared similarity search methods for (count-based) vectors on the basis of distributional semantics (Gorman and Curran, 2006), our study advances this topic and makes the following contributions: (a) we focus on neural word embeddings learned by a recently developed skip-gram model (Mikolov, 2013), (b) show that a graph-based search method clearly performs better than the best one reported in the Gorman and Curran study from different aspects, and (c) report the useful facts that normalizing vectors can achieve an effective search with cosine similarity, the search performance is more strongly related to a learning model of embeddings than its training data, the distribution shape of embeddings is a key factor relating to the search performance, and the final performance of a target application can be far different from the search performance. We believe that our timely results can lead to the practical use of embeddings, especially for real-time applications in the real world.

The rest of the paper is organized as follows. In Section 2, we briefly survey hash-, tree-, and graph-based indexing methods for achieving similarity search in a metric space. In Section 3, we

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2265–2275,
Berlin, Germany, August 7-12, 2016.

compare several similarity search algorithms from different aspects and discuss the results. Finally, Section 4 concludes the paper.

2 Similarity Search

We briefly survey similarity search algorithms for real-valued vectors, where we focus on approximate algorithms that can deal with large scale data. In fact, word embeddings are usually trained on a very large corpus. For example, well known pre-trained word embeddings (Mikolov, 2013) were trained on the Google News dataset and consist of about 1,000 billion words with 300-dimensional real-valued vectors. Search tasks on large-scale real-valued vectors have been more actively studied in the image processing field than in the natural language processing field, since such tasks naturally correspond to searching for similar images with their feature vectors.

Many similarity search algorithms have been developed and are classified roughly into three indexing types: hash-, tree-, and graph-based. In natural language processing, hash-based indexing seems to be preferred because of its simplicity and ease of treating both sparse and dense vectors, while in image processing, tree- and graph-based indexing are preferred because of their performance and flexibility in adjusting parameters. We explain these three indexing types in more detail below.

2.1 Hash-based Indexing

Hash-based indexing is a method to reduce the dimensionality of high-dimensional spaces by using some hash functions so that we can efficiently search in the reduced space. Locality-sensitive hashing (LSH) (Gionis et al., 1999) is a widely used hash-based indexing algorithm, which maps similar vectors to the same hash values with high probability by using multiple hash functions.

There are many hash-based indexing algorithms that extend LSH for different metric spaces. Datar et al. (2004) applied the LSH scheme to L^p spaces, or Lebesgue spaces, and experimentally showed that it outperformed the existing methods for the case of $p = 2$. Weiss et al. (2009) showed that the problem of finding the best hash function is closely related to the problem of graph partitioning and proposed an efficient approximate algorithm by reducing the problem to calculating thresholded eigenvectors of the graph Laplacian. In this paper, we focus on approximation of k-nearest neighbors and are not concerned about the hash-based indexing algorithms, since they are basically designed for finding (not k-nearest) neighbors within a fixed radius of a given point, i.e., a so-called *radius search*.

2.2 Tree-based Indexing

Tree-based indexing is used to recursively divide the entire search space into hierarchical subspaces, where the subspaces are not necessarily disjointed, so that the search space forms a tree structure. Given a search query, we can efficiently find the subspaces including the query by descending from the root note to the leaf nodes in the tree structure and then obtain its search results by scanning only neighbors belonging to the subspaces. Note that in contrast to the hash-based indexing, we can easily extend the size of search results or the number of nearest neighbors by ascending to the parent subspaces.

Arya et al. (1998) proposed the balanced box-decomposition tree (BBD-tree) as a variant of the kd-tree (Bentley, 1975) for approximately searching for similar vectors on the basis of Minkowski metrics, i.e., in L^p spaces when $p \geq 1$. Fast library for approximate nearest neighbors (FLANN) (Muja and Lowe, 2008) is an open-source library for approximate similarity search. FLANN automatically determines the optimal one from three indices: a randomized kd-tree where multiple kd-trees are searched in parallel (Silpa-Anan and Hartley, 2008), a k-means tree that is constructed by hierarchical k-means partitioning (Nister and Stewenius, 2006), and a mix of both kd-tree and k-means tree. Spatial approximation sample hierarchy (SASH) (Houle and Sakuma, 2005) achieves approximate search with multiple hierarchical structures created by random sampling. According to the results in the previous study (Gorman and Curran, 2006), SASH performed the best for vectors on the basis of distributional semantics, and its performance surpassed that of LSH.

2.3 Graph-based Indexing

Graph-based indexing is a method to approximately find nearest neighbors by using a neighborhood graph, where each node is connected to its nearest neighbors calculated on the basis of a certain metric. A simple search procedure for a given query is achieved as follows. An arbitrary node in the graph is selected as a candidate for the

true nearest neighbor. In the process of checking the nearest neighbor of the candidate, if the query is closer to the neighbor than the candidate, the candidate is replaced by the neighbor. Otherwise, the search procedure terminates by returning the current candidate as the nearest neighbor of the query. This procedure can be regarded as a best-first search, and the result is an approximation of that of an exact search.

Sebastian and Kimia (2002) first used a k-nearest neighbor graph (KNNG) as a search index, and Hajebi et al. (2011) improved the search performance by performing hill-climbing starting from a randomly sampled node of a KNNG. Their experimental results with image features, i.e., scale invariant feature transform (SIFT), showed that a similarity search based on a KNNG outperforms randomized kd-trees and LSH. Although the brute force construction cost of a KNNG drastically increases as the number of nodes increases because the construction procedure needs to calculate the nearest neighbors for each node, we can efficiently approximate a KNNG (so-called ANNG) by incrementally constructing an ANNG with approximate k-nearest neighbors calculated on a partially constructed ANNG. Neighborhood graph and tree for indexing (NGT) (Iwasaki, 2015) is a library released from Yahoo! JAPAN that achieves a similarity search on an ANNG; it has already been applied to several services.

3 Experiments

In this paper, we focused on the pure similarity search task of word embeddings rather than complex application tasks for avoiding extraneous factors, since many practical tasks can be formulated as k-nearest neighbor search. For example, assuming search engines, we can formalize query expansion, term deletion, and misspelling correction as finding frequent similar words, infrequent similar words, and similar words with different spellings, respectively.

We chose FLANN from the tree-based methods and NGT from the graph-based methods since they are expected to be suitable for practical use. FLANN and NGT are compared with SASH, which was the best method reported in a previous study (Gorman and Curran, 2006). In addition, we consider LSH only for confirmation, since it is widely used in natural language processing, although several studies have reported that LSH performed worse than SASH and FLANN. We used the E2LSH package (Andoni, 2004), which includes an implementation of a practical LSH algorithm.

3.1 Problem Definition

The purpose of an approximate similarity search is to quickly and accurately find vectors close to a given vector. We formulate this task as a problem to find k-nearest neighbors as follows. Let (X, d) be a metric space. We denote by $N_k(x, d)$ the set of k-nearest neighbors of a vector $x \in X$ with respect to a metric d. Formally, the following condition holds: $\forall y \in N_k(x, d), \forall z \in X \setminus N_k(x, d)$, $d(x, y) \leq d(x, z)$. Our goal with this problem is to approximate $N_k(x, d)$ for a given vector x.

We calculate the precision of an approximate search method A using the so-called *precision at k or P@k*, which is a widely used evaluation measure in information retrieval. The precision at k of A is defined as $|N_k(x, d) \cap \tilde{N}_k(x, A)|/k$, where $\tilde{N}_k(x, A)$ is the set of approximate k-nearest neighbors of a vector x calculated by A. Since we use the same size k for an exact set $N_k(x, d)$ and its approximate set $\tilde{N}_k(x, A)$, there is no trade-off between precision and recall.

3.2 Basic Settings

This section describes the basic settings in our experiments, where we changed a specific setting (e.g., number of dimensions) in order to evaluate the performance in each experiment. All the experiments were conducted on machines with two Xeon L5630 2.13-GHz processors and 24 GB of main memory running Linux operating systems.

We prepared 200-dimensional word embeddings learned from English Wikipedia in February 2015, which contains about 3 billion sentences spanning about 2 million words and 35 billion tokens, after preprocessing with the widely used script (Mahoney, 2011), which was also used for the word2vec demo (Mikolov, 2013). We used the skip-gram learning model with hierarchical softmax training in the word2vec tool, where the window size is 5, and the down-sampling parameter is 0.001.

We constructed and evaluated the index by dividing the learned embeddings into 2 million embeddings for training and 1,000 embeddings for testing by random sampling, after normalizing them so that the norm of each embedding was one. We built the search index of each search method

for the training set on the basis of the Euclidean distance. The Euclidean distance of normalized vectors is closely related to the cosine similarity, as described later. We prepared the top-10 (exact) nearest neighbors in the training set corresponding to each embedding in the testing set and plotted the average precision at 10 over the test set versus its computation time (log-scale), by changing the parameter for precision of each method as described below. Note that it is difficult to compare different algorithms in terms of either precision or computation time, since there is a trade-off between precision and computation time in approximate search.

We set the parameters of the three search methods SASH, FLANN, and NGT as follows. We determined stable parameters for indexing using grid search and changed an appropriate parameter that affected the accuracy when evaluating each method. For confirmation, we added LSH in the first experiment but did not use it in the other experiments since it clearly performs worse than the other methods.

SASH We set the maximum number (p) of parents per node to 6 for indexing and changed the scale factor for searching[1].

FLANN We set the target precision to 0.8, the build weight to 0, and the sample fraction to 0.01 for indexing, and we changed the number of features to be checked in the search[2]. The k-means index was always selected as the optimal index in our experiments.

NGT We set the edge number (E) to 10 for indexing and changed the search range (e) for searching.

LSH We set the success probability ($1 - \delta$) to 0.9 and changed the radius (R) for indexing. Note that there are no parameters for searching since LSH was developed to reduce dimensionality, and we need to construct multiple indices for adjusting its accuracy.

[1]The scale factor is implemented as "scaleFactor" in the source code (Houle, 2005), although there is no description in the original paper (Houle and Sakuma, 2005).

[2]Since FLANN is a library integrating several algorithms, the parameters can be described only by variables in the source code (Muja and Lowe, 2008). The target precision, build weight, and sample fraction for auto-tuned indexing are implemented as "target_precision", "build_weight", and "sample_fraction" in the structure "AutotunedIndexParams", respectively. The number of features is implemented as "checks" in the structure "SearchParams".

3.3 Results

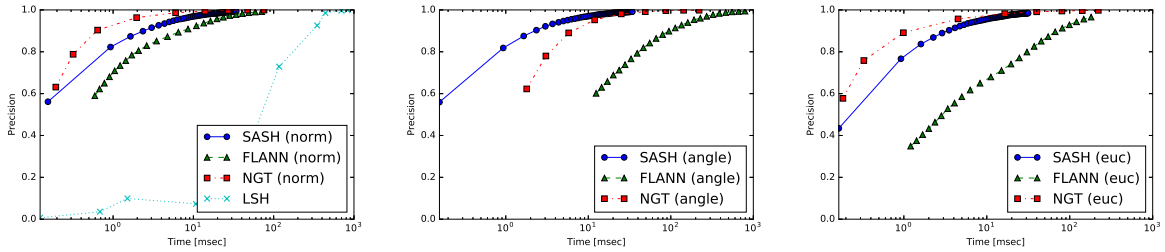
In this section we report the results of the performance comparison of SASH, FLANN, and NGT from the following different aspects: the distance function for indexing, the number of dimensions of embeddings, the number of neighbors to be evaluated, the size of a training set for indexing, the learning model/data used for embeddings, and the target task to be solved.

3.3.1 Distance Function for Indexing

We evaluated the performance by changing the distance function for indexing. In natural language processing, cosine similarity $\cos(x, y) = \frac{x \cdot y}{\|x\| \, \|y\|}$ of two vectors x and y is widely used from a practical perspective, and cosine distance $d^{\cos}(x, y) = 1 - \cos(x, y)$ as its complement seems to be appropriate for the distance function for indexing. Unfortunately, however, the cosine distance is not strictly metric but *semimetric* since the triangle inequality is not satisfied. Thus, we cannot directly use the cosine distance because the triangle inequality is a key element for efficient indexing in a metric space. In this paper, we use two alternatives: normalized and angular distances.

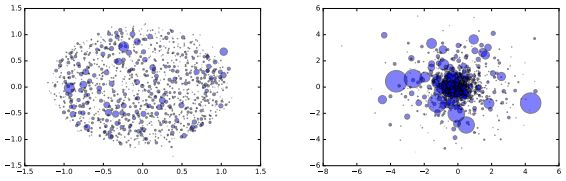
The former is the Euclidean distance after normalizing vectors, i.e., $d^{\text{norm}}(x, y) = d^{\text{euc}}(\frac{x}{\|x\|}, \frac{y}{\|y\|})$, where $d^{\text{euc}}(x, y) = \|x - y\|$. The set of k-nearest neighbors by d^{norm} is theoretically the same as that by $d^{\cos}$, i.e., $N_k(x, d^{\text{norm}}) = N_k(x, d^{\cos})$, since $d^{\text{norm}}(x, y)^2 = \frac{\|x\|^2}{\|x\|^2} + \frac{\|y\|^2}{\|y\|^2} - 2\frac{x}{\|x\|} \cdot \frac{y}{\|y\|} = 2d^{\cos}(x, y)$. The latter is the angle between two vectors, i.e., $d^{\text{arc}}(x, y) = \arccos(\cos(x, y))$. The set of k-nearest neighbors by d^{arc} is also the same as that by $d^{\cos}$, i.e., $N_k(x, d^{\text{arc}}) = N_k(x, d^{\cos})$, since $\arccos$ is a monotone decreasing function. Note that d^{arc} is not strictly metric, but it satisfies the triangle inequality, i.e., *pseudometric*.

Figure 1 plots the performances of SASH, FLANN, and NGT using the normalized, angular, and ordinal Euclidean distances. Higher precision at the same computational time (upper left line) indicates a better result. The graphs show that NGT performed the best for the normalized distance (a), while SASH performed the best for the angular distance (b). This large difference is caused by the long computational time of d^{arc}. Because we only want the maximum performance in graphs (a) and (b) for each method, we used only the normalized distance in the later experiments since the perfor-



(a) Normalized (b) Angular (c) Euclidean

Figure 1: Precision versus computation time of SASH, FLANN, and NGT using the normalized, angular, and Euclidean distances.



(a) Normalized (b) Un-normalized

Figure 2: 2D visualization of normalized and un-normalized embeddings by multi-dimensional scaling.

Distance	Method	Time (min)
Normalized	SASH	74.6
	FLANN	56.5
	NGT	**33.9**
	LSH	44.6
Angular	SASH	252.4
	FLANN	654.9
	NGT	**155.4**
Euclidean	SASH	58.1
	FLANN	**20.2**
	NGT	83.0

Table 1: Indexing time of SASH, FLANN, NGT, and LSH using the normalized, angular, Euclidean distance functions.

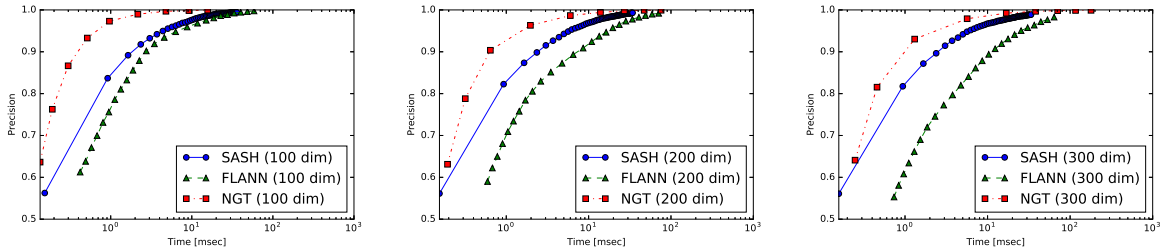
mance of SASH in graph (a) is almost the same as that in (b). For confirmation, we added the result of LSH in graph (a) only. The graph clearly indicates that the performance of LSH is very low even for neural word embeddings, which supports the results in the previous study (Gorman and Curran, 2006), and therefore we did not use LSH in the later experiments.

Graph (c) shows that the performance using the Euclidean distance has a similar tendency to that using the normalized distance, but its computation time is much worse than that using the normalized distance. The reason for this is that it is essentially difficult to search for distant vectors in a metric-based index, and normalization can reduce the number of distant embeddings by aligning them on a hypersphere. In fact, we can confirm that the number of distant embeddings was reduced after normalization according to Figure 2, which visualizes 1,000 embeddings before/after normalization on a two-dimensional space by multi-dimensional scaling (MDS) (Borg and Groenen, 2005), where the radius of each circle represents the search time of the corresponding embedding calculated by NGT. MDS is a dimensionality reduction method to place each point in a low-dimensional space such that the distances between any two points are preserved as much as possible. Note that the scale of graph (b) is about five times larger than that of graph (a). This also suggests that the normalized distance should be preferred even when it has almost the same precision as the Euclidean distance.

Table 1 lists the indexing times of SASH, FLANN, and NGT on the basis of the normalized, angular, and Euclidean distances, where LSH is also added only in the result of the normalized distance. The table indicates that NGT performed the best for the normalized and angular distances, while FLANN performed the best for the Euclidean distance. However, all methods seem to be suitable for practical use in terms of indexing because we can create an index of English Wikipedia embeddings in several hours (only once). The large indexing time with the angular distance also supports our suggestion that the normalized distance should be used.

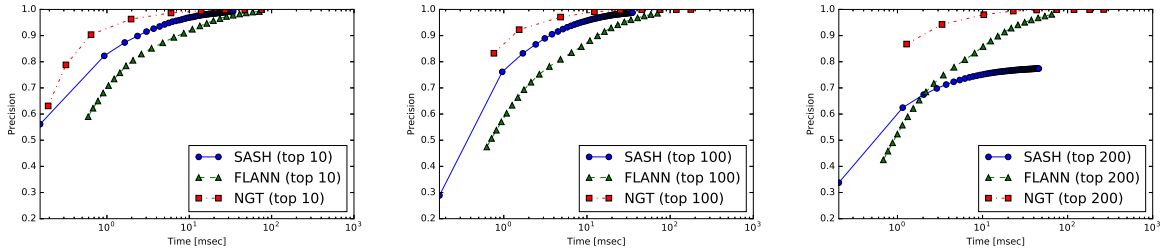
3.3.2 Number of Dimensions of Embeddings

We also evaluated the performances by changing the number of dimensions of embeddings. Since the optimal number of dimensions should depend on the tasks, we wanted to see how the search



(a) 100 dimensions (b) 200 dimensions (c) 300 dimensions

Figure 3: Precision versus computation time of SASH, FLANN, and NGT using 100-, 200-, and 300-dimensional embeddings.



(a) P@10 (b) P@100 (c) P@200

Figure 4: Precision versus computation time of SASH, FLANN, and NGT using precision at 10, 100, and 200.

methods performed when the number of dimensions varied, while the number of dimensions of image features is usually fixed. For example, SIFT features (Lowe, 1999) are represented as 128-dimensional vectors.

Figure 3 plots the performances of SASH, FLANN, and NGT using 100-, 200-, and 300-dimensional embeddings. The graphs indicate that NGT always performed the best. SASH is expected to perform well when the number of dimensions is large, since FLANN and NGT perform worse as the number of dimensions becomes larger. However, NGT would be a better choice since most existing pre-trained embeddings (Turian et al., 2010; Mikolov, 2013; Pennington et al., 2014a) have a few hundred dimensions.

3.3.3 Number of Neighbors to Be Evaluated

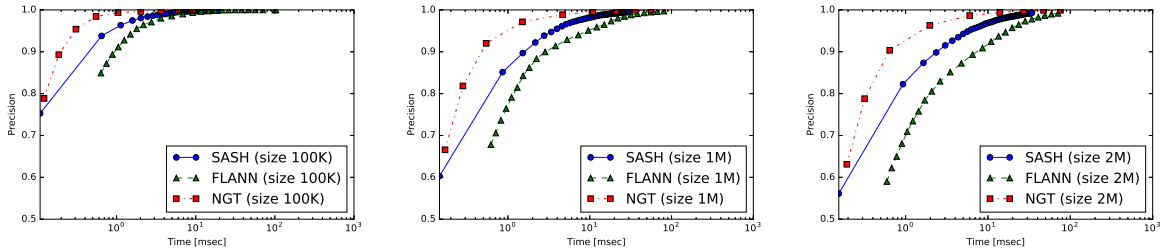
We also conducted performance evaluations by changing the number k of neighbors, i.e., the size of the set of k-nearest neighbors, to calculate the precision at k. We need to change the number k on demand from target applications. For example, we may use small numbers for extracting synonyms and large numbers for selecting candidates for news recommendations, where they will be reduced via another sophisticated selection process.

The performances of SASH, FLANN, and NGT using 10-, 100-, and 200-nearest neighbors are shown in Figure 4. The graphs indicate that NGT performed the best in this measure also. With 200-nearest neighbors, the performance of SASH dropped sharply, which means that SASH is not robust for the indexing parameter. One possible reason is that searching for relatively distant neighbors is difficult for a tree-based index, where the divided subspaces are not appropriate.

3.3.4 Size of Training Set for Indexing

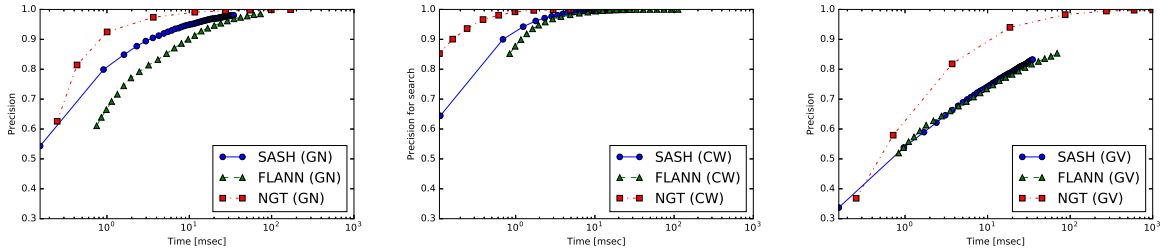
We conducted further performance evaluations by changing the size of a training set, i.e., the number of embeddings used for indexing. We wanted to know how the search methods performed with different sized search indices since a large search index will bring about extra operational costs in a practical sense, and a small search index is preferred for a small application system.

Figure 5 plots the performances of SASH, FLANN, and NGT using 100K, 1M, and 2M training sets, which were randomly sampled so that each training set can be virtually regarded as embeddings with a vocabulary of its training set size. The graphs indicate that NGT always performed the best for all search index sizes. Moreover, we can see that all results for each method have a similar tendency. This fact implies that a distribution of embeddings is related to the search per-



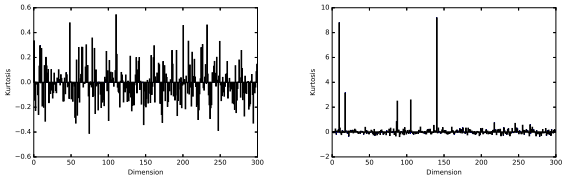
(a) 100K (b) 1M (c) 2M

Figure 5: Precision versus computation time of SASH, FLANN, and NGT using 100K, 1M, and 2M training sets.



(a) GN (b) CW (c) GV

Figure 6: Precision versus computation time of SASH, FLANN, and NGT using GN, CW, and GV embeddings.



(a) GN (b) GV

Figure 7: Kurtosis of each dimension of GN and GV embeddings.

formance, and the next section will actually confirm the same property on another dataset used for learning embeddings.

3.3.5 Model and Data Used for Embeddings

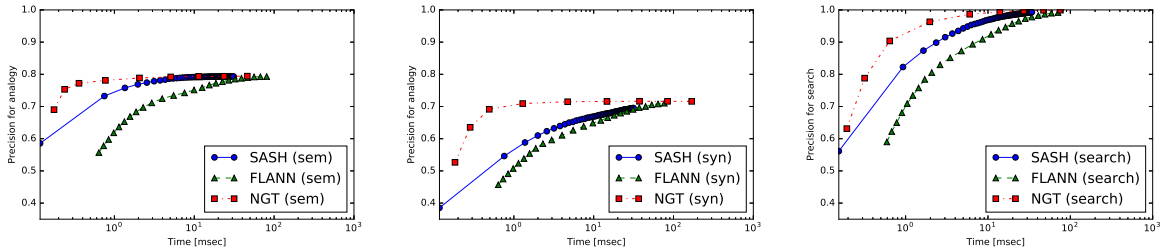
We also conducted performance evaluations by changing the learning models and training data for embeddings. We used the following three pre-trained embeddings to investigate the performance when changing the data distributions used for indexing.

GN 300-dimensional embeddings (Mikolov, 2013) learned by the skip-gram model with negative sampling (Mikolov et al., 2013a) using part of the Google News dataset, which contains about 3 million words and phrases and 100 billion tokens.

CW 200-dimensional embeddings (Turian et al., 2010) learned by deep neural networks (Collobert and Weston, 2008) using the RCV1 corpus, which contains about 269 thousand words and 63 million tokens.

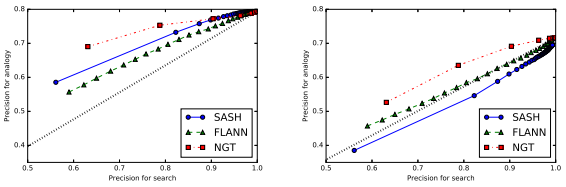
GV 300-dimensional embeddings (Pennington et al., 2014a) learned by the global vectors for word representation (GloVe) model (Pennington et al., 2014b) using Common Crawl corpora, which contain about 2 million words and 42 billion tokens.

The performances of SASH, FLANN, and NGT using GN, CW, and GV embeddings are plotted in Figure 6. The graphs indicate that NGT consistently performed the best over different learning models. A comparison of the results using GN embeddings and the previous results using Wikipedia embeddings reveals that they had almost the same tendency. This fast can be acceptable assuming an empirical rule that a corpus follows a power law or Zipf's law. On the other hand, graphs (a), (b), and (c) have quite different tendencies. Specifically, all search methods compete with each other for CW embeddings, while they could not perform well for GV embeddings. This implies that the performance of a search method can be affected by learning models rather than training sets used for embeddings.



(a) Semantic analogy (b) Syntactic analogy (c) Similarity search

Figure 8: Precision versus computation time of SASH, FLANN, and NGT using the semantic analogy, syntactic analogy, and similarity search tasks.



(a) Semantic analogy (b) Syntactic analogy

Figure 9: Precision of the semantic and syntactic analogy tasks versus that of the similarity search task.

	EW	GN	CW	GV
Variance	0.0033	0.0033	0.0050	0.0033
Kurtosis	0.034	-0.026	-0.075	0.57

Table 2: Variance and kurtosis of English Wikipedia (EW), GN, CW, and GV embeddings.

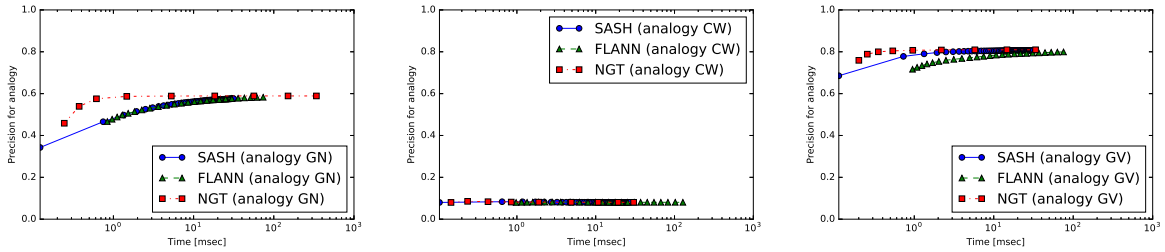
We further investigated why GV embeddings deteriorate the search performance. Table 2 lists the variance and kurtosis of Wikipedia, GN, CW, and GV embeddings for clarifying the variation or dispersion of these distributions. Kurtosis $K(X)$ is a measure of the "tailedness" of the probability distribution of a random variable X, defined by $K(X) = \mu_4/\mu_2^2 - 3$, where μ_n represents the n-th central moment, i.e., $\mathbb{E}[(X - \mathbb{E}[X])^n]$. The constant "3" in the above definition sets the kurtosis of a normal distribution to 0. The table clearly indicates that GV has a heavy tailed distribution in accordance with the kurtosis values, although all variances have almost the same value. In fact, GV has several high kurtosis peaks, while GN has only small values, according to Figure 7, which visualizes the kurtosis of each dimension. Note that the y-axis scale of graph (b) is about 20 times larger than that of graph (a). Because distant points in a metric space tend to deteriorate the performance in a search process, we need to pay attention to the distribution shape of embeddings as well as their quality, so as to efficiently search for similar embeddings.

3.3.6 Target Task to Be Solved

We finally evaluated the performance by changing the target task to be solved by using embeddings. We wanted to know how the search methods performed with different task settings since even if the precision of the search task is not good, it might be sufficient for another task to be solved on the basis of similarity search. In this section, we address well known analogy tasks (Mikolov et al., 2013a), where semantic and syntactic analogy questions are considered, e.g., "Which word corresponds to Japan when Paris corresponds to France?", the answer being "Tokyo". These questions can be solved by searching for the nearest neighbors of analogical vectors generated via arithmetic operations., i.e., vec("Paris") − vec("France") + vec("Japan"), where vec(w) represents an embedding of word w.

Figure 8 plots the performances of SASH, FLANN, and NGT using the semantic and syntactic analogy tasks as well as that using the similarity search task (in Figure 1), which is added for comparison. The graphs indicate that NGT clearly performed the best even in the analogy tasks. Comparing the curves of NGT, we can see that those in graphs (a) and (b) are quite different from that in (c), and the analogy precisions can maintain their quality, even when the search precision is about 0.9.

For further analysis, we aligned the precisions of the search task with those of the analogy tasks in Figure 9, where each point represents the results calculated with the same parameters. The dotted line without markers in each graph is a line from the origin (0, 0) to the point where the analogy precision is maximum when the search precision



(a) Analogy by GN (b) Analogy by CW (c) Analogy by GV

Figure 10: Precision versus computation time of SASH, FLANN, and NGT for the analogy task (including both semantic and syntactic questions) using GN, CW, and GV embeddings.

is 1.0, and thus it naively estimates a deterioration rate of the analogy precision on the basis of the search precision. The graphs indicate that the search precision can be far different from the estimated precision of another task. In fact, when the search precision by NGT is 0.8 in Figure 9 (a), the analogy precision 0.75 is unexpectedly high, since the naive estimation is 0.64 calculated by the maximum analogy precision 0.8 times the search precision 0.8. This suggests that it is a good idea to check the final performance of a target application, although the search performance is valuable from a standpoint of general versatility.

Finally, we conducted performance evaluations for the analogy task instead of the search task by changing the learning models and training data for embeddings as in Section 3.3.5, in order to support the robustness of NGT even for an operation more sophisticated than just finding similar words. Figure 10 plots the performances of SASH, FLANN, and NGT for the analogy task including both semantic and syntactic questions using GN, CW, and GV embeddings. The graphs indicate that NGT performed the best over different learning models even for the analogy task. Although the precisions of CW embeddings in graph (b) are very low, the result seems to be acceptable according to the previous work (Mikolov et al., 2013b), which reported that the precisions of a syntactic analogy task using CW embeddings in similar settings were at most 5 % (0.05). The results of GN and GV embeddings in graphs (a) and (c) show a similar tendency to those of Wikipedia embeddings in Figure 8. However, the overall performance for the analogy task using GV embeddings is unexpectedly high, contrary to the results for the search task in Figure 6 (c). One of the reasons is that arithmetic operations for solving analogy questions can reduce kurtosis peaks, although we omitted the kurtosis results due to space limitation. This fact also supports our finding that distant points in a metric space tend to deteriorate the performance in a search process.

4 Conclusion

We investigated approximate similarity search for word embeddings. We compared three methods: a graph-based method (NGT), a tree-based method (FLANN), the SASH method, which was reported to have the best performance in a previous study (Gorman and Curran, 2006). The results of experiments we conducted from various aspects indicated that NGT generally performed the best and that the distribution shape of embeddings is a key factor relating to the search performance. Our future research includes improving the search performance for embeddings with heavy-tailed distributions and creating embeddings that can keep both task quality and search performance high.

We will release the source code used for our comparative experiments from the NGT page (Iwasaki, 2015). Since we need to implement additional glue codes for running FLANN and SASH, our code would be useful for researchers who want to compare their results with ours.

Acknowledgments

We would like to thank the anonymous reviewers for giving us helpful comments.

References

Alexandr Andoni. 2004. LSH Algorithm and Implementation (E2LSH). `http://web.mit.edu/andoni/www/LSH/`.

Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. 1998. An Optimal Algorithm for Approximate Nearest Neighbor

Searching Fixed Dimensions. *Journal of the ACM (JACM)*, 45(6):891–923.

Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Communication of the ACM*, 18(9):509–517.

Ingwer Borg and Patrick J. F. Groenen. 2005. *Modern Multidimensional Scaling*. Springer Series in Statistics. Springer-Verlag New York.

Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 160–167. ACM.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. 2004. Locality-sensitive Hashing Scheme Based on P-stable Distributions. In *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG 2004)*, pages 253–262. ACM.

Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB 2009)*, pages 518–529. Morgan Kaufmann Publishers Inc.

James Gorman and James R. Curran. 2006. Scaling Distributional Similarity to Large Corpora. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL 2006)*, pages 361–368. Association for Computational Linguistics.

Kiana Hajebi, Yasin Abbasi-Yadkori, Hossein Shahbazi, and Hong Zhang. 2011. Fast Approximate Nearest-neighbor Search with K-nearest Neighbor Graph. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011)*, pages 1312–1317. AAAI Press.

Michael E. Houle and Jun Sakuma. 2005. Fast Approximate Similarity Search in Extremely High-Dimensional Data Sets. In *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, pages 619–630. IEEE Computer Society.

Michael E. Houle. 2005. The SASH Page. `http://research.nii.ac.jp/%7Emeh/sash/sashpage.html`.

Masajiro Iwasaki. 2015. NGT : Neighborhood Graph and Tree for Indexing. `http://research-lab.yahoo.co.jp/software/ngt/`.

David G. Lowe. 1999. Object Recognition from Local Scale-Invariant Features. In *Proceedings of the International Conference on Computer Vision (ICCV 1999)*, pages 1150–1157. IEEE Computer Society.

Matt Mahoney. 2011. About the Test Data. `http://mattmahoney.net/dc/textdata.html`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013a. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119. Curran Associates, Inc.

Tomas Mikolov, Wen tau Yih, and Geoffrey Zweig. 2013b. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 746–751. Association for Computational Linguistics.

Tomas Mikolov. 2013. word2vec: Tool for computing continuous distributed representations of words. `https://code.google.com/p/word2vec/`.

Marius Muja and David G. Lowe. 2008. FLANN — Fast Library for Approximate Nearest Neighbors. `http://www.cs.ubc.ca/research/flann/`.

David Nister and Henrik Stewenius. 2006. Scalable Recognition with a Vocabulary Tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 2161–2168. IEEE Computer Society.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014a. GloVe: Global Vectors for Word Representation. `http://nlp.stanford.edu/projects/glove/`.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014b. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1532–1543.

Thomas B. Sebastian and Benjamin B. Kimia. 2002. Metric-Based Shape Retrieval in Large Databases. In *Proceedings of the 16th International Conference on Pattern Recognition (ICPR 2002)*, pages 291–296.

Chanop Silpa-Anan and Richard Hartley. 2008. Optimised KD-trees for fast image descriptor matching. In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*, pages 1–8. IEEE Computer Society.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. CCG: RTE Annotation Data for ACL 2010 publication. `http://cogcomp.cs.illinois.edu/Data/ACL2010_NER_Experiments.php`.

Yair Weiss, Antonio Torralba, and Robert Fergus. 2009. Spectral Hashing. In *Advances in Neural Information Processing Systems 21 (NIPS 2008)*, pages 1753–1760. Curran Associates, Inc.

Justin Zobel and Alistair Moffat. 2006. Inverted Files for Text Search Engines. *ACM Computing Surveys*, 38(2).

Composing Distributed Representations of Relational Patterns

Sho Takase **Naoaki Okazaki** **Kentaro Inui**
Graduate School of Information Sciences, Tohoku University
{takase, okazaki, inui}@ecei.tohoku.ac.jp

Abstract

Learning distributed representations for relation instances is a central technique in downstream NLP applications. In order to address semantic modeling of relational patterns, this paper constructs a new dataset that provides multiple similarity ratings for every pair of relational patterns on the existing dataset (Zeichner et al., 2012). In addition, we conduct a comparative study of different encoders including additive composition, RNN, LSTM, and GRU for composing distributed representations of relational patterns. We also present Gated Additive Composition, which is an enhancement of additive composition with the gating mechanism. Experiments show that the new dataset does not only enable detailed analyses of the different encoders, but also provides a gauge to predict successes of distributed representations of relational patterns in the relation classification task.

1 Introduction

Knowledge about entities and their relations (relation instances) are crucial for a wide spectrum of NLP applications, e.g., information retrieval, question answering, and recognizing textual entailment. Learning distributed representations for relation instances is a central technique in downstream applications as a number of recent studies demonstrated the usefulness of distributed representations for words (Mikolov et al., 2013; Pennington et al., 2014) and sentences (Sutskever et al., 2014; Cho et al., 2014; Kiros et al., 2015).

In particular, semantic modeling of relations and their textual realizations (*relational patterns* hereafter) is extremely important because a relation (e.g., causality) can be mentioned by various expressions (e.g., "*X* cause *Y*", "*X* lead to *Y*", "*Y* is associated with *X*"). To make matters worse, relational patterns are highly productive: we can produce a emphasized causality pattern "*X* increase the severe risk of *Y*" from "*X* increase the risk of *Y*" by inserting *severe* to the pattern.

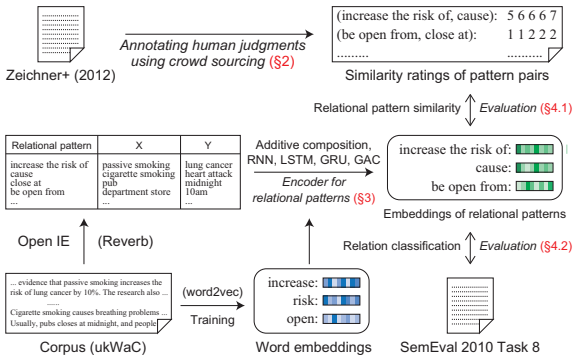


Figure 1: Overview of this study.

To model the meanings of relational patterns, the previous studies built a co-occurrence matrix between relational patterns (e.g., "*X* increase the risk of *Y*") and entity pairs (e.g., "*X*: smoking, *Y*: cancer") (Lin and Pantel, 2001; Nakashole et al., 2012). Based on the distributional hypothesis (Harris, 1954), we can compute a semantic vector of a relational pattern from the co-occurrence matrix, and measure the similarity of two relational patterns as the cosine similarity of the vectors. Nowadays, several studies adopt distributed representations computed by neural networks for semantic modeling of relational patterns (Yih et al., 2014; Takase et al., 2016).

Notwithstanding, the previous studies paid little attention to explicitly evaluate semantic modeling of relational patterns. In this paper, we construct a new dataset that contains a pair of relational patterns with five similarity ratings judged by human annotators. The new dataset shows a

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2276–2286,
Berlin, Germany, August 7-12, 2016.

high inter-annotator agreement, following the annotation guideline of Mitchell and Lapata (2010). The dataset is publicly available on the Web site[1].

In addition, we conduct a comparative study of different encoders for composing distributed representations of relational patterns. During the comparative study, we present Gated Additive Composition, which is an enhancement of additive composition with the gating mechanism. We utilize the Skip-gram objective for training the parameters of the encoders on a large unlabeled corpus. Experiments show that the new dataset does not only enable detailed analyses of the different encoders, but also provides a gauge to predict successes of distributed representations of relational patterns in another task (relation classification). Figure 1 illustrates the overview of this study.

2 Data Construction

2.1 Target relation instances

We build a new dataset upon the work of Zeichner et al. (2012), which consists of relational patterns with semantic inference labels annotated. The dataset includes 5,555 pairs[2] extracted by Reverb (Fader et al., 2011), 2,447 pairs with inference relation and 3,108 pairs (the rest) without one.

Initially, we considered using this high-quality dataset as it is for semantic modeling of relational patterns. However, we found that inference relations exhibit quite different properties from those of semantic similarity. Take a relational pattern pair "*X* be the part of *Y*" and "*X* be an essential part of *Y*" filled with "*X* = the small intestine, *Y* = the digestive system" as an instance. The pattern "*X* be the part of *Y*" does not entail "*X* be an essential part of *Y*" because the meaning of the former does not include 'essential'. Nevertheless, both statements are similar, representing the same relation (PART-OF). Another uncomfortable pair is "*X* fall down *Y*" and "*X* go up *Y*" filled with "*X* = the dude, *Y* = the stairs". The dataset indicates that the former entails the latter probably because falling down from the stairs requires going up there, but they present the opposite meaning. For this reason, we decided to re-annotate semantic similarity judgments on every pair of relational patterns on the dataset.

2.2 Annotation guideline

We use instance-based judgment in a similar manner to that of Zeichner et al. (2012) to secure a high inter-annotator agreement. In instance-based judgment, an annotator judges a pair of relational patterns whose variable slots are filled with the same entity pair. In other words, he or she does not make a judgment for a pair of relational patterns with variables, "*X* prevent *Y*" and "*X* reduce the risk of *Y*", but two instantiated statements "Cephalexin prevent the bacteria" and "Cephalexin reduce the risk of the bacteria" ("*X* = Cephalexin, *Y* = the bacteria"). We use the entity pairs provided in Zeichner et al. (2012).

We asked annotators to make a judgment for a pair of relation instances by choosing a rating from 1 (dissimilar) to 7 (very similar). We provided the following instructions for judgment, which is compatible with Mitchell and Lapata (2010): (1) rate 6 or 7 if the meanings of two statements are the same or mostly the same (e.g., "Palmer team with Jack Nicklaus" and "Palmer join with Jack Nicklaus"); (2) rate 1 or 2 if two statements are dissimilar or unrelated (e.g., "the kids grow up with him" and "the kids forget about him"); (3) rate 3, 4, or 5 if two statements have some relationships (e.g., "Many of you know about the site" and "Many of you get more information about the site", where the two statements differ but also reasonably resemble to some extent).

2.3 Annotation procedure

We use a crowdsourcing service CrowdFlower[3] to collect similarity judgments from the crowds. CrowdFlower has the mechanism to assess the reliability of annotators using Gold Standard Data (Gold, hereafter), which consists of pairs of relational patterns with similarity scores assigned. Gold examples are regularly inserted throughout the judgment job to enable measurement of the performance of each worker[4]. Two authors of this paper annotated 100 pairs extracted randomly from 5,555 pairs, and prepared 80 Gold examples showing high agreement. Ratings of the Gold examples were used merely for quality assessment of the workers. In other words, we discarded the

[1] http://github.com/takase/relPatSim

[2] More precisely, the dataset includes 1,012 *meaningless* pairs in addition to 5,555 pairs. A pair of relational patterns was annotated as *meaningless* if the annotators were unable to understand the meaning of the patterns easily. We ignore the *meaningless* pairs in this study.

[3] http://www.crowdflower.com/

[4] We allow ± 1 differences in rating when we measure the performance of the workers.

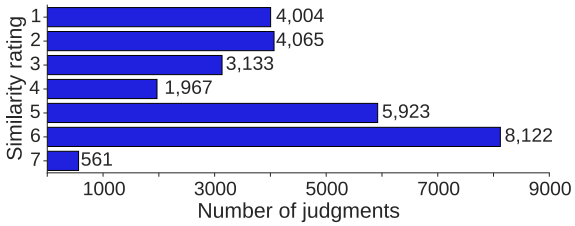


Figure 2: Number of judgments for each similarity rating. The total number of judgments is $27,775$ ($5,555$ pairs $\times$ 5 workers).

similarity ratings of the Gold examples, and used those judged by the workers.

To build a high quality dataset, we use judgments from workers whose confidence values (reliability scores) computed by CrowdFlower are greater than 75%. Additionally, we force every pair to have at least five judgments from the workers. Consequently, 60 workers participated in this job. In the final version of this dataset, each pair has five similarity ratings judged by the five most reliable workers who were involved in the pair.

Figure 2 presents the number of judgments for each similarity rating. Workers seldom rated 7 for a pair of relational patterns, probably because most pairs have at least one difference in content words. The mean of the standard deviations of similarity ratings of all pairs is 1.16. Moreover, we computed Spearman's ρ between similarity judgments from each worker and the mean of five judgments in the dataset. The mean of Spearman's ρ of workers involved in the dataset is 0.728. These statistics show a high inter-annotator agreement of the dataset.

3 Encoder for Relational Patterns

The new dataset built in the previous section raises two new questions — *What is the reasonable method (encoder) for computing the distributed representations of relational patterns? Is this dataset useful to predict successes of distributed representations of relational patterns in real applications?* In order to answer these questions, this section explores various methods for learning distributed representations of relational patterns.

3.1 Baseline methods without supervision

A naïve approach would be to regard a relational pattern as a single unit (word) and to train word/pattern embeddings as usual. In fact, Mikolov et al. (2013) implemented this approach as a preprocessing step, mining phrasal expressions with strong collocations from a training corpus. However, this approach might be affected by data sparseness, which lowers the quality of distributed representations.

Another simple but effective approach is *additive composition* (Mitchell and Lapata, 2010), where the distributed representation of a relational pattern is computed by the mean of embeddings of constituent words. Presuming that a relational pattern consists of a sequence of T words $w_1, ..., w_T$, then we let $x_t \in \mathbb{R}^d$ the embedding of the word w_t. This approach computes $\frac{1}{T}\sum_{t=1}^{T} x_t$ as the embedding of the relational pattern. Muraoka et al. (2014) reported that the additive composition is a strong baseline among various methods.

3.2 Recurrent Neural Network

Recently, a number of studies model semantic compositions of phrases and sentences by using (a variant of) Recurrent Neural Network (RNN) (Sutskever et al., 2014; Tang et al., 2015). For a given embedding x_t at position t, the vanilla RNN (Elman, 1990) computes the hidden state $h_t \in \mathbb{R}^d$ by the following recursive equation[5],

$$h_t = g(W_x x_t + W_h h_{t-1}). \quad (1)$$

Here, W_x and W_h are $d \times d$ matrices (parameters), $g(.)$ is the elementwise activation function ($\tanh$). We set $h_0 = 0$ at $t = 1$. In essence, RNN computes the hidden state h_t based on the one at the previous position (h_{t-1}) and the word embedding x_t. Applying Equation 1 from $t = 1$ to T, we use h_T as the distributed representation of the relational pattern.

3.3 RNN variants

We also employ Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) as an encoder for relational patterns. LSTM has been applied successfully to various NLP tasks including word segmentation (Chen et al., 2015), dependency parsing (Dyer et al., 2015), machine translation (Sutskever et al., 2014), and sentiment analysis (Tai et al., 2015). GRU is also successful in machine translation (Cho et al., 2014) and various

[5]We do not use a bias term in this study. We set the number of dimensions of hidden states identical to that of word embeddings (d) so that we can adapt the objective function of the Skip-gram model for training (Section 3.5).

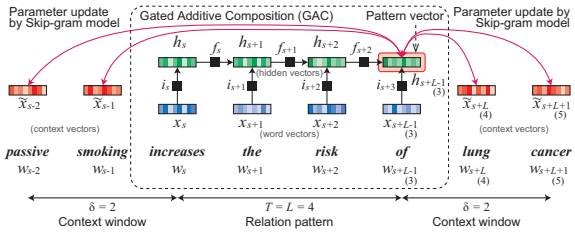


Figure 3: Overview of GAC trained with Skip-gram model. GAC computes the distributed representation of a relational pattern using the input gate and forget gate, and learns parameters by predicting surrounding words (Skip-gram model).

tasks including sentence similarity, paraphrase detection, and sentiment analysis (Kiros et al., 2015).

LSTM and GRU are similar in that the both architectures have gates (input, forget, and output for LSTM; reset and update for GRU) to remedy the gradient vanishing or explosion problem in training RNNs. Although some researchers reported that GRU is superior to LSTM (Chung et al., 2014), we have no consensus about the superiority. Besides, we are not sure whether LSTM or GRU is really necessary for relational patterns, which ususlly consist of a few words. Thus, we compare RNN, LSTM, and GRU empirically with the same training data and the same training procedure. Similarly to RNN, we use the hidden state h_T of LSTM[6] or GRU as the distributed representation of a relation pattern.

3.4 Gated Additive Composition (GAC)

In addition to the gradient problem, LSTM or GRU may be suitable for relational patterns, having the mechanism of adaptive control of gates for input words and hidden states. Consider the relational pattern "*X* have access to *Y*", whose meaning is mostly identical to that of "*X* access *Y*". Because 'have' in the pattern is a light verb, it may be harmful to incorporate the semantic vector of 'have' into the distributed representation of the pattern. The same may be true for the functional word 'to' in the pattern. However, the additive composition nor RNN does not have a mechanism to ignore the semantic vectors of these words. It is interesting to explore a method somewhere between additive composition and LSTM/GRU: additive composition with the gating mechanism.

For this reason, we present an another variant of RNN in this study. Inspired by the input and forget gates in LSTM, we compute the input gate $i_t \in \mathbb{R}^d$ and forget gate $f_t \in \mathbb{R}^d$ at position t. We use them to control the amount to propagate to the hidden state h_t from the current word x_t and the previous state h_{t-1}.

$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1}) \quad (2)$$

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1}) \quad (3)$$

$$h_t = g(f_t \odot h_{t-1} + i_t \odot x_t) \quad (4)$$

Here, W_{ix}, W_{ih}, W_{fx}, W_{fh} are $d \times d$ matrices. Equation 4 is interpreted as a weighted additive composition between the vector of the current word x_t and the vector of the previous hidden state h_{t-1}. The elementwise weights are controlled by the input gate i_t and forget gate f_t; we expect that input gates are closed (close to zero) and forget gates are opened (close to one) when the current word is a control verb or function word. We name this architecture *gated additive composition* (GAC).

3.5 Parameter estimation: Skip-gram model

To train the parameters of the encoders (RNN, LSTM, GRU, and GAC) on an unlabeled text corpus, we adapt the Skip-gram model (Mikolov et al., 2013). Formally, we designate an occurrence of a relational pattern p as a subsequence of L words $w_s, ..., w_{s+L-1}$ in a corpus. We define δ words appearing before and after pattern p as the context words, and let $C_p = (s-\delta, ..., s-1, s+L, ..., s+L+\delta)$ denote the indices of the context words. We define the log-likelihood of the relational pattern l_p, following the objective function of Skip-gram with negative sampling (SGNS) (Levy and Goldberg, 2014).

$$l_p = \sum_{\tau \in C_p} \left(\log \sigma(h_p^\top \tilde{x}_\tau) + \sum_{k=1}^{K} \log \sigma(-h_p^\top \tilde{x}_{\tilde{\tau}}) \right) \quad (5)$$

In this formula: K denotes the number of negative samples; $h_p \in \mathbb{R}^d$ is the vector for the relational pattern p computed by each encoder such as RNN; $\tilde{x}_\tau \in \mathbb{R}^d$ is the context vector for the word w_τ[7]; $x_{\tilde{\tau}'} \in \mathbb{R}^d$ is the context vector for the word

[6] We omitted peephole connections and bias terms.

[7] The Skip-gram model has two kinds of vectors x_t and $\tilde{x}_t$ assigned for a word w_t. Equation 2 of the original paper (Mikolov et al., 2013) denotes x_t (word vector) as v (input vector) and $\tilde{x}_t$ (context vector) as v' (output vector). The word2vec implementation does not write context (output) vectors but only word (input) vectors to a model file. Therefore, we modified the source code to save context vectors, and use them in Equation 5. This modification ensures the consistency of the entire model.

that were sampled from the unigram distribution[8] at every iteration of $\sum_k$.

At every occurrence of a relational pattern in the corpus, we use Stochastic Gradient Descent (SGD) and backpropagation through time (BPTT) for training the parameters (matrices) in encoders. More specifically, we initialize the word vectors x_t and context vectors $\tilde{x}_t$ with pre-trained values, and compute gradients for Equation 5 to update the parameters in encoders. In this way, each encoder is trained to compose a vector of a relational pattern so that it can predict the surrounding context words. An advantage of this parameter estimation is that the distributed representations of words and relational patterns stay in the same vector space. Figure 3 visualizes the training process for GAC.

4 Experiments

In Section 4.1, we investigate the performance of the distributed representations computed by different encoders on the pattern similarity task. Section 4.2 examines the contribution of the distributed representations on SemEval 2010 Task 8, and discusses the usefulness of the new dataset to predict successes of the relation classification task.

4.1 Relational pattern similarity

For every pair in the dataset built in Section 2, we compose the vectors of the two relational patterns using an encoder described in Section 3, and compute the cosine similarity of the two vectors. Repeating this process for all pairs in the dataset, we measure Spearman's ρ between the similarity values computed by the encoder and similarity ratings assigned by humans.

4.1.1 Training procedure

We used ukWaC[9] as the training corpus for the encoders. This corpus includes the text of 2 billion words from Web pages crawled in the .uk domain. Part-of-speech tags and lemmas are annotated by TreeTagger[10]. We used lowercased lemmas throughout the experiments. We apply word2vec to this corpus to pre-train word vectors x_t and context vectors $\tilde{x}_t$. All encoders use word vectors x_t to compose vectors of relational patterns; and the Skip-gram model uses context vectors $\tilde{x}_t$ to compute the objective function and gradients. We fix the vectors x_t and $\tilde{x}_t$ with pre-trained values during training.

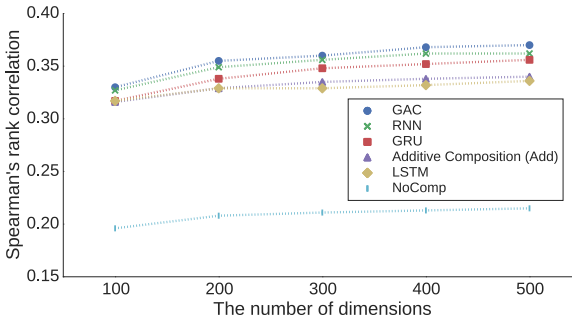


Figure 4: Performance of each method on the relational pattern similarity task with variation in the number of dimensions.

We used Reverb (Fader et al., 2011) to the ukWaC corpus to extract relational pattern candidates. To remove unuseful relational patterns, we applied filtering rules that are compatible with those used in the publicly available extraction result[11]. Additionally, we discarded relational patterns appearing in the evaluation dataset throughout the experiments to assess the performance under which an encoder composes vectors of unseen relational patterns. This preprocessing yielded $127,677$ relational patterns.

All encoders were implemented on Chainer[12], a flexible framework of neural networks. The hyperparameters of the Skip-gram model are identical to those in Mikolov et al. (2013): the width of context window $\delta = 5$, the number of negative samples $K = 5$, the subsampling of 10^{-5}. For each encoder that requires training, we tried 0.025, 0.0025, and 0.00025 as an initial learning rate, and selected the best value for the encoder. In contrast to the presentation of Section 3, we compose a pattern vector in backward order (from the last to the first) because preliminary experiments showed a slight improvement with this treatment.

4.1.2 Results and discussions

Figure 4 shows Spearman's rank correlations of different encoders when the number of dimensions of vectors is 100–500. The figure shows that GAC achieves the best performance on all dimensions.

Figure 4 includes the performance of the naïve approach, "NoComp", which regards a relational pattern as a single unit (word). In this approach,

[8] We use the probability distribution of words raised to the 3/4 power (Mikolov et al., 2013).

[9] http://wacky.sslmit.unibo.it

[10] http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/

[11] http://reverb.cs.washington.edu/

[12] http://chainer.org/

Length	#	NoComp	Add	LSTM	GRU	RNN	GAC
1	636	**0.324**	**0.324**	**0.324**	**0.324**	**0.324**	**0.324**
2	1,018	0.215	0.319	0.257	0.274	0.285	**0.321**
3	2,272	0.234	0.386	0.344	0.370	0.387	**0.404**
4	1,206	0.208	0.306	0.314	**0.329**	0.319	0.323
> 5	423	0.278	0.315	0.369	0.384	**0.394**	0.357
All	5,555	0.215	0.340	0.336	0.356	0.362	**0.370**

Table 1: Spearman's rank correlations on different pattern lengths (number of dimensions $d = 500$).

we allocated a vector h_p for each relational pattern p in Equation 5 instead of the vector composition, and trained the vectors of relational patterns using the Skip-gram model. The performance was poor for two reasons: we were unable to compute similarity values for 1,744 pairs because relational patterns in these pairs do not appear in ukWaC; and relational patterns could not obtain sufficient statistics because of data sparseness.

Table 1 reports Spearman's rank correlations computed for each pattern length. Here, the length of a relational-pattern pair is defined by the maximum of the lengths of two patterns in the pair. In length of 1, all methods achieve the same correlation score because they use the same word vector x_t. The table shows that additive composition (Add) performs well for shorter relational patterns (lengths of 2 and 3) but poorly for longer ones (lengths of 4 and 5+). GAC also exhibits the similar tendency to Add, but it outperforms Add for shorter patterns (lengths of 2 and 3) probably because of the adaptive control of input and forget gates. In contrast, RNN and its variants (RNN, GRU, and LSTM) enjoy the advantage on longer patterns (lengths of 4 and 5+).

To examine the roles of input and forget gates of GAC, we visualize the moments when input/forget gates are wide open or closed. More precisely, we extract the input word and scanned words when $|i_t|_2$ or $|f_t|_2$ is small (close to zero) or large (close to one) on the relational-pattern dataset. We restate that we compose a pattern vector in backward order (from the last to the first): GAC scans 'of', 'author', and 'be' in this order for composing the vector of the relational pattern 'be author of'.

Table 2 displays the top three examples identified using the procedure. The table shows two groups of tendencies. Input gates open and forget gates close when scanned words are only a preposition and the current word is a content word. In these situations, GAC tries to read the semantic vector of the content word and to ignore the semantic vector of the preposition. In contrast, input gates close and forget gates open when the current word is 'be' or 'a' and scanned words form a noun phrase (e.g., "charter member of"), a complement (e.g., "eligible to participate in"), or a passive voice (e.g., "require(d) to submit"). This behavior is also reasonable because GAC emphasizes informative words more than functional words.

	w_t	w_{t+1} w_{t+2} ...
large i_t (input open)	reimburse payable liable	for in to
small i_t (input close)	a a be	charter member of valuable member of an avid reader of
large f_t (forget open)	be be be	eligible to participate in require to submit request to submit
small f_t (forget close)	coauthor capital center	of of of

Table 2: Prominent moments for input/forget gates.

4.2 Relation classification

4.2.1 Experimental settings

To examine the usefulness of the dataset and distributed representations for a different application, we address the task of relation classification on the SemEval 2010 Task 8 dataset (Hendrickx et al., 2010). In other words, we explore whether high-quality distributed representations of relational patterns are effective to identify a relation type of an entity pair.

The dataset consists of $10,717$ relation instances ($8,000$ training and $2,717$ test instances) with their relation types annotated. The dataset

Method	Feature set	F1
SVM	BoW, POS	77.3
SVM + NoComp	embeddings, BoW, POS	79.9
SVM + LSTM	embeddings, BoW, POS	81.1
SVM + Add	embeddings, BoW, POS	81.1
SVM + GRU	embeddings, BoW, POS	81.4
SVM + RNN	embeddings, BoW, POS	81.7
SVM + GAC	embeddings, BoW, POS	82.0
	+ dependency, WordNet, NE	83.7
Ranking loss + GAC w/ fine-tuning	embeddings, BoW, POS	
	+ dependency, WordNet, NE	84.2
SVM (Rink and Harabagiu, 2010)	BoW, POS, dependency, Google n-gram, etc.	82.2
MV-RNN (Socher et al., 2012)	embeddings, parse trees	79.1
	+ WordNet, POS, NE	82.4
FCM (Gormley et al., 2015) w/o fine-tuning	embeddings, dependency	79.4
	+ WordNet	82.0
w/ fine-tuning	embeddings, dependency	82.2
	+ NE	83.4
RelEmb (Hashimoto et al., 2015)	embeddings	82.8
	+ dependency, WordNet, NE	83.5
CR-CNN (dos Santos et al., 2015) w/ Other	embeddings, word position embeddings	82.7
w/o Other	embeddings, word position embeddings	84.1
depLCNN (Xu et al., 2015)	embeddings, dependency	81.9
	+ WordNet	83.7
depLCNN + NS	embeddings, dependency	84.0
	+ WordNet	85.6

Table 3: F1 scores on the SemEval 2010 dataset.

defines 9 directed relations (e.g.,CAUSE-EFFECT) and 1 undirected relation OTHER. Given a pair of entity mentions, the task is to identify a relation type in 19 candidate labels (2×9 directed + 1 undirected relations). For example, given the pair of entity mentions e_1 = 'burst' and e_2 = 'pressure' in the sentence "The *burst* has been caused by water hammer *pressure*", a system is expected to predict CAUSE-EFFECT(e_2, e_1).

We used Support Vector Machines (SVM) with a Radial Basis Function (RBF) kernel implemented in libsvm[13]. Basic features are: part-of-speech tags (predicted by TreeTagger), surface forms, lemmas of words appearing between an entity pair, and lemmas of the words in the entity pair. Additionally, we incorporate distributed representations of a relational pattern, entities, and a word before and after the entity pair (number of dimensions $d = 500$). In this task, we regard words appearing between an entity pair as a relational pattern. We compare the vector representations of relational patterns computed by the five encoders presented in Section 4.1: additive composition, RNN, GRU, LSTM, and GAC. Hyperparameters related to SVM were tuned by 5-fold cross validation on the training data.

[13] https://www.csie.ntu.edu.tw/~cjlin/libsvm/

4.2.2 Results and discussions

Table 3 presents the macro-averaged F1 scores on the SemEval 2010 Task 8 dataset. The first group of the table provides basic features and enhancements with the distributed representations. We can observe a significant improvement even from the distributed representation of NoComp (77.3 to 79.9). Moreover, the distributed representation that exhibited the high performance on the pattern similarity task was also successful on this task; GAC, which yielded the highest performance on the pattern similarity task, also achieved the best performance (82.0) of all encoders on this task.

It is noteworthy that the improvements brought by the different encoders on this task roughly cor-

respond to the performance on the pattern similarity task. This fact implies two potential impacts. First, the distributed representations of relational patterns are useful and easily transferable to other tasks such as knowledge base population. Second, the pattern similarity dataset provides a gauge to predict successes of distributed representations in another task.

We could further improve the performance of SVM + GAC by incorporating external resources in the similar manner as the previous studies did. Concretely, SVM + GAC achieved 83.7 F1 score by adding features for WordNet, named entities (NE), and dependency paths explained in Hashimoto et al. (2015). Moreover, we could obtain 84.2 F1 score, using the ranking based loss function (dos Santos et al., 2015) and fine-tuning of the distributed representations initially trained by GAC. Currently, this is the second best score among the performance values reported in the previous studies on this task (the second group of Table 3). If we could use the negative sampling technique proposed by Xu et al. (2015), we might improve the performance further[14].

5 Related Work

Mitchell and Lapata (2010) was a pioneering work in semantic modeling of short phrases. They constructed the dataset that contains two-word phrase pairs with semantic similarity judged by human annotators. Korkontzelos et al. (2013) provided a semantic similarity dataset with pairs of two words and a single word. Wieting et al. (2015) annotated a part of PPDB (Ganitkevitch et al., 2013) to evaluate semantic modeling of paraphrases. Although the target unit of semantic modeling is different from that for these previous studies, we follow the annotation guideline and instruction of Mitchell and Lapata (2010) to build the new dataset.

The task addressed in this paper is also related to the Semantic Textual Similarity (STS) task (Agirre et al., 2012). STS is the task to measure the degree of semantic similarity between two sentences. Even though a relational pattern appears as a part of a sentence, it may be difficult to transfer findings from one to another: for example, the encoders of RNN and its variants explored in this study may exhibit different characteristics, influenced by the length and complexity of input text expressions.

In addition to data construction, this paper addresses semantic modeling of relational patterns. Nakashole et al. (2012) approached the similar task by constructing a taxonomy of relational patterns. They represented a vector of a relational pattern as the distribution of entity pairs co-occurring with the relational pattern. Grycner et al. (2015) extended Nakashole et al. (2012) to generalize dimensions of the vector space (entity pairs) by incorporating hyponymy relation between entities. They also used external resources to recognize the transitivity of pattern pairs and applied transitivities to find patterns in entailment relation. These studies did not consider semantic composition of relational patterns. Thus, they might suffer from the data sparseness problem, as shown by NoComp in Figure 4.

Numerous studies have been aimed at encoding distributed representations of phrases and sentences from word embeddings by using: Recursive Neural Network (Socher et al., 2011), Matrix Vector Recursive Neural Network (Socher et al., 2012), Recursive Neural Network with different weight matrices corresponding to syntactic categories (Socher et al., 2013) or word types (Takase et al., 2016), RNN (Sutskever et al., 2011), LSTM (Sutskever et al., 2014), GRU (Cho et al., 2014), PAS-CLBLM (Hashimoto et al., 2014), etc. As described in Section 3, we applied RNN, GRU, and LSTM to compute distributed representations of relational patterns because recent papers have demonstrated their superiority in semantic composition (Sutskever et al., 2014; Tang et al., 2015). In this paper, we presented a comparative study of different encoders for semantic modeling of relational patterns.

To investigate usefulness of the distributed representations and the new dataset, we adopted the relation classification task (SemEval 2010 Task 8) as a real application. On the SemEval 2010 Task 8, several studies considered semantic composition. Gormley et al. (2015) proposed Feature-rich Compositional Embedding Model (FCM) that can combine binary features (e.g., positional indicators) with word embeddings via outer products. dos Santos et al. (2015) addressed the task using Convolutional Neural Network (CNN). Xu et al. (2015) achieved a higher performance than dos

[14] In fact, we made substantial efforts to introduce the negative sampling technique. However, Xu et al. (2015) omits the detail of the technique probably because of the severe page limit of short papers. For this reason, we could not reproduce their method in this study.

Santos et al. (2015) by application of CNN on dependency paths.

In addition to the relation classification task, we briefly describe other applications. To populate a knowledge base, Riedel et al. (2013) jointly learned latent feature vectors of entities, relational patterns, and relation types in the knowledge base. Toutanova et al. (2015) adapted CNN to capture the compositional structure of a relational pattern during the joint learning. For open domain question answering, Yih et al. (2014) proposed the method to map an interrogative sentence on an entity and a relation type contained in a knowledge base by using CNN.

Although these reports described good performance on the respective tasks, we are unsure of the generality of distributed representations trained for a specific task such as the relation classification. In contrast, this paper demonstrated the contribution of distributed representations trained in a generic manner (with the Skip-gram objective) to the task of relation classification.

6 Conclusion

In this paper, we addressed the semantic modeling of relational patterns. We introduced the new dataset in which humans rated multiple similarity scores for every pair of relational patterns on the dataset of semantic inference (Zeichner et al., 2012). Additionally, we explored different encoders for composing distributed representations of relational patterns. The experimental results shows that Gated Additive Composition (GAC), which is a combination of additive composition and the gating mechanism, is effective to compose distributed representations of relational patterns. Furthermore, we demonstrated that the presented dataset is useful to predict successes of the distributed representations in the relation classification task.

We expect that several further studies will use the new dataset not only for distributed representations of relational patterns but also for other NLP tasks (e.g., paraphrasing). Analyzing the internal mechanism of LSTM, GRU, and GAC, we plan to explore an alternative architecture of neural networks that is optimal for relational patterns.

Acknowledgments

We thank the reviewers and Jun Suzuki for valuable comments. This work was partially supported by Grant-in-Aid for JSPS Fellows Grant no. 26.5820, JSPS KAKENHI Grant number 15H05318, and JST, CREST.

References

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *The First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*, pages 385–393.

Xinchi Chen, Xipeng Qiu, Chenxi Zhu, Pengfei Liu, and Xuanjing Huang. 2015. Long short-term memory neural networks for chinese word segmentation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1197–1206.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1724–1734.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling.

Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015)*, pages 626–634.

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

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

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011)*, pages 1535–1545.

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

Matthew R. Gormley, Mo Yu, and Mark Dredze. 2015. Improved relation extraction with feature-rich compositional embedding models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1774–1784.

Adam Grycner, Gerhard Weikum, Jay Pujara, James Foulds, and Lise Getoor. 2015. Relly: Inferring hypernym relationships between relational phrases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 971–981.

Zellig Harris. 1954. Distributional structure. *Word*, 10(23):146–162.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2014. Jointly learning word representations and composition functions using predicate-argument structures. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, pages 1544–1555.

Kazuma Hashimoto, Pontus Stenetorp, Makoto Miwa, and Yoshimasa Tsuruoka. 2015. Task-oriented learning of word embeddings for semantic relation classification. In *Proceedings of the 19th Conference on Computational Natural Language Learning (CoNLL 2015)*, pages 268–278.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 33–38.

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

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, pages 3276–3284.

Ioannis Korkontzelos, Torsten Zesch, Fabio Massimo Zanzotto, and Chris Biemann. 2013. Semeval-2013 task 5: Evaluating phrasal semantics. In *Second Joint Conference on Lexical and Computational Semantics (*SEM 2013)*, pages 39–47.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 2177–2185.

Dekang Lin and Patrick Pantel. 2001. Dirt – discovery of inference rules from text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 01)*, pages 323–328.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, pages 3111–3119.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1439.

Masayasu Muraoka, Sonse Shimaoka, Kazeto Yamamoto, Yotaro Watanabe, Naoaki Okazaki, and Kentaro Inui. 2014. Finding the best model among representative compositional models. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation (PACLIC 28)*, pages 65–74.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: A taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 1135–1145.

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

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT 2013)*, pages 74–84.

Bryan Rink and Sanda Harabagiu. 2010. Utd: Classifying semantic relations by combining lexical and semantic resources. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 256–259.

Richard Socher, Cliff Chiung-Yu Lin, Andrew Y. Ng, and Christopher D. Manning. 2011. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on Machine learning (ICML 2011)*, pages 129–136.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012)*, pages 1201–1211.

Richard Socher, John Bauer, Christopher D. Manning, and Ng Andrew Y. 2013. Parsing with compositional vector grammars. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013)*, pages 455–465.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pages 1017–1024.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, pages 3104–3112.

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

Sho Takase, Naoaki Okazaki, and Kentaro Inui. 2016. Modeling semantic compositionality of relational patterns. *Engineering Applications of Artificial Intelligence*, 50(C):256–264.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1422–1432.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 1499–1509.

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

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, pages 536–540.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014)*, pages 643–648.

Naomi Zeichner, Jonathan Berant, and Ido Dagan. 2012. Crowdsourcing inference-rule evaluation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 156–160.

The More Antecedents, the Merrier: Resolving Multi-Antecedent Anaphors

Hardik Vala[1], **Andrew Piper**[2], **Derek Ruths**[1]
[1]School of Computer Science, [2]Dept. of Languages, Literartures, & Cultures
McGill University
Montreal, Canada
hardik.vala@mail.mcgill.ca,
{andrew.piper, derek.ruths}@mcgill.ca

Abstract

Anaphor resolution is an important task in NLP with many applications. Despite much research effort, it remains an open problem. The difficulty of the problem varies substantially across different sub-problems. One sub-problem, in particular, has been largely untouched by prior work despite occurring frequently throughout corpora: the anaphor that has multiple antecedents, which here we call *multi-antecedent anaphors* or m-anaphors. Current coreference resolvers restrict anaphors to at most a single antecedent. As we show in this paper, relaxing this constraint poses serious problems in coreference chain-building, where each chain is intended to refer to a single entity. This work provides a formalization of the new task with preliminary insights into multi-antecedent noun-phrase anaphors, and offers a method for resolving such cases that outperforms a number of baseline methods by a significant margin. Our system uses local agglomerative clustering on candidate antecedents and an existing coreference system to score clusters to determine which cluster of mentions is antecedent for a given anaphor. When we augment an existing coreference system with our proposed method, we observe a substantial increase in performance (0.6 absolute CoNLL F1) on an annotated corpus.

1 Introduction

Anaphor resolution is a very difficult task in Natural Language Understanding, involving the complex interaction of discourse cues, syntactic rules, and semantic phenomena. It is closely related to the task of coreference resolution (Van Deemter and Kibble, 2000), for which a myriad of solutions have been proposed (Clark and Manning, 2015; Peng et al., 2015; Wiseman et al., 2015; Björkelund and Farkas, 2012; Lee et al., 2011; Stoyanov et al., 2010; Ng, 2008; Bergsma and Lin, 2006; Soon et al., 2001). However, given the complexity of the problem, a comprehensive approach remains elusive. The difficulty varies drastically across different cases (proper nouns, pronouns, gerunds, etc.), each of which involves different assumptions about and models of various linguistic phenomena (e.g., vocabulary, syntax, and semantics). As a result, state-of-the-art systems yield varying performance across sub-problems (Mitkov, 2014; Kummerfeld and Klein, 2013; Björkelund and Nugues, 2011; Recasens and Hovy, 2009; Stoyanov et al., 2009; Bengtson and Roth, 2008; Van Deemter and Kibble, 2000; Ng and Cardie, 2002b; Kameyama, 1997).

To avoid the complexity of the overarching resolution task, many current systems — whether learning-based (Clark and Manning, 2015; Peng et al., 2015; Wiseman et al., 2015; Durrett and Klein, 2013; Björkelund and Farkas, 2012) or rule-based (Lee et al., 2011) — focus on a restricted version of the problem, where candidate anaphors are linked to at most one antecedent, from which coreference chains are built by propagating the induced equivalence relation, with each chain corresponding to an entity (Van Deemter and Kibble, 2000).

While this single-antecedent inference task does resolve a very large number of anaphors in any given text, it leaves one quite common sub-problem virtually untouched: anaphors that link to multiple antecedents. These have sometimes been called split-antecedent anaphors; here we use the term *multi-antecedent anaphors* or m-anaphors in

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2287–2296,
Berlin, Germany, August 7-12, 2016.

order to emphasize the existence of more than one (possibly more than two) antecedents for a given anaphor. Consider the following examples:

(1) *[Elizabeth]$_1$ met [Mary]$_2$ at the park and [they]$_{1,2}$ began their stroll to the river.*

(2) *Mrs. Dashwood, having moved to another country, saw her [mother]$_1$ and [sister-in-law]$_2$ demoted to occasional visitors. As such, however, her old [kin]$_{1,2}$ were treated by her new family with quiet civility.*

Such cases present a challenge to state-of-the-art methods: certain features well-suited for the single-antecedent case do not apply (e.g. gender and pluarity) (Recasens and Hovy, 2009; Stoyanov et al., 2009; Bergsma and Lin, 2006), and strong long-distance effects cannot be ignored (Ingria and Stallard, 1989). Moreover, the presence of multiple antecedents for a single anaphor violates the separation between coreference chains.

In this paper, we address the multi-antecedent case of noun-phrase (NP) anaphor resolution in English, the most widely understood and studied form of coreference resolution (Ng, 2010; Ng, 2008). While we frame the general question of multi-antecedent inference, we restrict our analyses to one particular sub-problem: resolving the antecedents of the pronouns *they* and *them*. These pronouns best isolate the characteristics of m-anaphors (see Section 2 for more on the motivation of this choice). We propose a system for resolving *they* and *them* that models grouping compatibility of mentions through a maximum entropy pairwise model, independently from coreference of groupings, which is handled through an existing coreference resolution system leveraging corpus knowledge.

This paper makes four core contributions. First, it provides a generalization of the anaphor resolution problem to permit linking to multiple antecedents. Second, we characterize core properties of m-anaphors and their linguistic environments in a large, annotated corpus. Third, we provide a entity-centric system for specifically resolving multi-antecedent cases that outperforms a number of baselines. And, finally, we show how to pair our system with an existing coreference system and show a gain of 0.6 points (CoNLL F1) on the complete coreference resolution task (resolving all anaphors, single- and multi-antecedent).

The rest of the paper is organized as follows: We introduce the terminology and problem statement for split-antecedent resolution in Section 2. A summary of the data is given in Section 3 and the behaviour of split-antecedent anaphors is analyzed in Section 4. Our approach to antecedent prediction is presented in Section 5 and the results and analysis are reported in Section 6. Finally, we review related work in Section 7 and conclude and discuss future work in Section 8.

2 Problem

This section establishes the terminology used throughout the paper and reformulates the anaphor resolution problem to incorporate linking to multiple antecedents.

2.1 Terminology

We introduce the term m-anaphor for convenience as a special case of anaphor that has to multiple antecedents. For example, *they* and *kin* in Examples (1) and (2), respectively, from the Introduction are m-anaphors. By extension, 1-anaphors are anaphors that have only one antecedent.

Similarly, we define an m-antecedent as one of multiple antecedents of an m-anaphor and we refer to m-antecedents with the same m-anaphor as *siblings*. In Example (1) from the Introduction, *Elizabeth* and *Mary* are sibling m-antecedents of *they*, and in Example (2), *mother* and *sister-in-law* are sibling m-antecedents of *kin*.

Finally, we refer to anaphors with two, three, and four m-antecedents as *2-anaphors*, *3-anaphors*, and *4-anaphors*, respectively. We provide two more examples:

(3) *[Mr. Holmes]$_1$ stared off into the distance. [Watson]$_2$ simply walked off. [Both]$_{1,2}$ were troubled by the news.*

(4) *Virginia found herself alone with her [brother]$_1$, and then the thought of her [sister]$_2$ came to mind. [She]$_3$ remembered the camping trip [they]$_{1,2,3}$ embarked on a few summers ago.*

The anaphor in Example (3) is a 2-anaphor and the anaphor in Example (4) is a 3-anaphor.

2.2 Definition

We define the NP anaphor resolution problem similar to Wiseman et al. (2015), Durrett and Klein

Pronoun	# m-anaphors
they	278
them	165
we	140
you	43
everybody	12

Table 1: Counts of the most frequent m-anaphoric pronouns in P&P.

(2013), and Hirschman (1997): Let $\mathcal{M}$ denote the set of all identified mentions in a document and let $M(x) \subseteq \mathcal{M}$ denote all mentions preceding a mention $x \in \mathcal{M}$. The objective of the task is, for each $x \in \mathcal{M}$, to find $C \subseteq M(x)$ such that all mentions in C are antecedent to x. If $C = \emptyset$, then x is non-anaphoric and if $|C| \geq 1$, then x is 1-anaphoric, and if $|C| > 1$, then x is m-anaphoric. Hence, this formulation generalizes the problem to account for multi-antecedent anaphors.

To constrain the scope of the study, we perform all our analyses on gold mentions, leaving the effect of imperfect mention detection as a problem for future work (this has been studied for the single-antecedent case in Stoyanov et al. (2009)). Moreover, we only consider mentions of *they* and *them* that are known to be m-anaphoric for three reasons. First, non-pronomial m-anaphors, i.e. proper and common nouns, are much more susceptible to long-distance effects and may require external knowledge to resolve. Second, by focusing on this case, we circumvent a host of very involved aspects of the complete m-anaphor resolution problem, i.e. determining whether a mention is m-anaphoric, 1-anaphoric, or not anaphoric at all. For example, *you* may refer to one person or multiple, *who* can be used as an interrogative (non-anaphoric) or reflexive pronoun (anaphoric)), pronouns such as *anyone* and *everyone* introduce many scoping difficulties, and pleonastic pronouns must be removed from the inference task entirely. Third, *they* and *them* are the most prevalent of all pronouns in our dataset (refer to Table 1).

3 Data

Our dataset comprises of the *Pride and Prejudice* novel (P&P) (121440 words) and 36 short stories from the *Scribner Anthology of Contemporary Short Fiction* (Martone et al., 1999) (Scribner) (total of 216901 words), representing an eclectic collection of stories from the modern era. For P&P, all mentions of character have been fully resolved to their antecedents, including mentions referencing multiple characters. For Scribner, all mentions of *they* and *them* are resolved (m-anaphoric, 1-anaphoric, and singleton), including those of non-person entities.

	they		*them*		Total	
	#	%	#	%	#	%
P&P	278	32.10	165	19.05	443	51.15
Scribner	243	12.96	79	4.21	322	17.17
Total	521	19.01	244	8.90	765	27.91

Table 2: Number of m-anaphoric *they* and *them* mentions and % of all *they* and *them* mentions that are m-anaphors.

These stories were annotated by three annotators according to a slightly modified version of the ACE coreference resolution task formulation (Doddington et al., 2004) to allow multiple antecedents. Annotations were conducted through the *brat*[1] annotation tool (Stenetorp et al., 2012)) and the inter-annotator agreement on the shared texts (3 stories from Scribner + 7 chapters from P&P) was 86.5%.

Overall, in P&P, 1289 m-anaphors were discovered, of which 34 (2.6%) were proper nouns, 536 (41.6%) were common nouns, and 719 (55.8%) were pronouns. Table 2 shows the number of gold m-anaphoric *they* and *them* mentions and the percentage of all *they* and *them* mentions that are m-anaphoric.

Literary works were chosen over other textual modalities, e.g. news articles, because they showed a higher density of m-anaphors (a preliminary annotation exercise showed that literary works contained 37% more m-anaphors per word).

The dataset is partitioned according to a roughly, 60/20/20 split into training, validation, and testing sets, where the split is applied to the text of P&P (e.g. the first 60% of story text is used for training), and the collection of Scribner stories (e.g. 60% of the stories were used for training).

4 Behaviour of m-anaphors

m-anaphors present a novel class of anaphor for which very little knowledge exists. To better understand the linguistic behaviour of m-anaphors, we perform the following analyses. First, we examine first and second order statistics of our

[1] `http://brat.nlplab.org`

	First	Second
Avg. distance (# words)	17.08	33.50
Std. distance (# words)	23.80	40.66
Avg. distance (# sent.)	1.19	2.28
Std. distance (# sent.)	3.18	5.10
Avg. # intermediates	1.44	4.21
Std. # intermediates	2.33	4.44

Table 3: Average and standard deviations of the word distance, sentence distance, and number of intermediate mentions between the first and second most recent mentions to an m-anaphor.

Feature	Coefficient	p-value
Sentence position = first	0.16	0.13
Sentence position = last	-0.18	0.006
Dependency = subject	0.27	0.05
Dependency = object	0.08	0.24
Dependency = preposition	-0.22	0.07
Coordinated = true	0.29	0.08
Presence of negation	0.06	0.31
Presence of modality	0.04	0.21

Table 4: Features for m-anaphoricity versus 1-anaphoricity with coefficients estimated from a maximum entropy model, and associated p-values.

dataset to gain insight into the distribution of m-anaphors across a number of dimensions. Second, we fit a maximum entropy model over common coreference features for distinguishing m-anaphoric and anaphoric mentions to evaluate the importance of various features in determining m-anaphoricity versus anaphoricity of mentions.

4.1 m-anaphor Statistics

The distribution of m-anaphors according to the number of referenced $m-$antecedents is as follows: 79.3% are 2-anaphors, 13.2% are 3-anaphors, 3.7% are 4-anaphors, and the remaining 3.8% refer to larger numbers of antecedents. Despite the bias towards 2-anaphors, the simple approach to m-anaphor resolution of taking the previous two mentions as m-antecedent siblings will fail according to Table 3. The usual presence of intermediate mentions between m-anaphors and their m-antecedents makes the resolution task non-trivial. Moreover, the large distances between m-anaphors and their antecedents attenuates any signal for coreference, introducing greater noise to the problem.

4.2 m-anaphoricity Features

The statistics discussed above shed light on the complexity of this problem. Here, we examine whether certain surface-level features of anaphoric phenomena from prior work exhibit any differences for m-anaphoric mentions over anaphoric ones. We construct a maximum entropy model from the training data over the combination of syntactic and semantic features in Table 4, inspired by Wiseman et al. (2015), Durrett and Klein (2013), and Recasens et al. (2013b). The binary classification decision is between m-anaphoric and 1-anaphoric mentions, coded as '1' and '0', respectively. Therefore, the estimated coefficients that are positive favor m-anaphoricity and those that are negative favor 1-anaphoricity.

Except for the feature testing on the last sentence position, none of the results in Table 4 were able to reach statistical significance, suggesting at a surface level, m-anaphoricity and 1-anaphoricity behave very similarly and operate in similar linguistic environments. One possibility is that a deeper set of features is required for distinguishing m-anaphors from 1-anaphors. We identify this as an important topic for future work in this area.

5 m-anaphor Resolution

Our approach to m-anaphor resolution draws inspiration from mention pair models for coreference that make independent binary classification decisions (Ng, 2010). In our method, we employ a maximum entropy model that makes binary decisions on mention pairs as well, but the decision corresponds to "group compatibility" of mentions, i.e. to what degree can a given set of mentions be the sibling m-antecedents to the same m-anaphor. This model is embedded in an agglomerative clustering process, after which a coreference decision is made between clusters and the given m-anaphor. Thus, our model treats the grouping of candidate mentions into sibling sets independently from antecedent-anaphor linking.

5.1 Architecture

Given an m-anaphor g in document $\mathcal{D}$, the steps of our approach are as follows:

1. Mentions preceding g within a k-sentence window are extracted as candidate m-antecedents to g.

2. Perform an agglomerative clustering of the candidate mentions using similarity metric

SIM_1 and average-linkage criteria. Let $\mathcal{C}$ represent the clustering.

3. Each non-singleton cluster $C \in \mathcal{C}$ is scored according to the probability of coreference of the m-anaphor to the cluster. This is done by appealing to an external corpus comprising of sentences containing either *they* or *them*. The grouping of sentences in the document containing all of the mentions in C (and sentences in-between) are compared to each *they* or *them* sentence in the external corpus (depending on the identity of g) using similarity metric SIM_2. The sentence yielding the maximum similarity is selected. The probability of coreference is then calculated by replacing the sentence grouping with the extracted sentence and applying an existing coreference system $COREF$ between g and its counterpart (*they* or *them*) in the extracted sentence.

4. The cluster C_{max} producing the highest probability of coreference is predicted as the group of m-antecedents for g.

Again, inspired by mention-pair models for coreference resolution (Clark and Manning, 2015; Björkelund and Farkas, 2012; Ng and Cardie, 2002a), the SIM_1 similarity metric is defined as $\sigma(\mathbf{w}^\top \mathbf{x})$, where $\mathbf{w}$ is a weight vector and $\mathbf{x}$ is a feature vector defined for a pair of mentions. The parameter vector $\mathbf{w}$ is learned using the standard cross-entropy loss function in a maximum entropy model, where the target variable is a decision on whether the mentions pairs are siblings or not. The learning is conducted over the training set with L2-regularization.

For SIM_2, which is responsible for selecting replacement sentences, we experiment with two different similarity metrics: (1) longest common subsequence normalized by sentence length (LCS) and (2) a subset tree kernel (Collins and Duffy, 2002) with a bag-of-words extension as described in Moschitti (2006), which also describes a simple adaptation to forests (for multiple sentences). The named entity (NE) mentions in sentences are replaced by corresponding NE type placeholders (PERSON, LOCATION, etc. as described in Finkel et al. (2005)) before comparison.

In the experiments to follow, we adopt the classification mention-pair model, a component of the statistical coreference resolution system available in the Stanford CoreNLP suite[2] system, described in Clark and Manning (2015), as $COREF$ for scoring coreference. The external corpus was built from texts comparable to our dataset. 651,108 sentences containing one of *they* or *them* were mined from a larger corpus of 798 literary texts spanning the nineteenth and twentieth centuries (including novels such as *To The Lighthouse*, by Virginia Woolf). Lastly, the candidate m-antecedents are extracted from a 5-sentence pre-window of the given m-anaphor (k = 5) and the regularization parameter in learning is set to 0.20.

5.2 Clustering Features

Table 5 depicts the features we chose to use in the pairwise similarity metric (SIM_1) for agglomerative clustering of candidate m-antecedents. All are common to many coreference resolver systems (Durrett and Klein, 2013; Recasens et al., 2013b; Stoyanov et al., 2010). We distinguish between mention features (Columns 1 & 2), which are defined for each candidate m-antecedent in a pair, and pairwise features (Columns 3-5), which are defined over a pair of candidate m-antecedents.

Three features, in particular, deserve further discussion. Under morphosyntax (Column 3), *[Type Conjunctions]* is a placeholder for a number of conjunctive boolean features derived from the noun type (pronoun/proper/common) of each antecedent in a pairing: e.g., pronoun-pronoun, pronoun-proper, proper-pronoun. Similarly, *[Dependency Conjunctions]* is a placeholder for conjunctive boolean features derived from the grammatical dependency of each antecedent in a pairing: e.g., subject-subject, subject-object, object-subject. The *[# Dependency Pairings]* is an ordinal version of the *Dependency Conjunctions* feature set - a count of the number of occurrences rather than an indicator variable.

The 'Governor = except' feature triggers if one of the mentions in the mention pair is governed by *except* or *exclude*. It represents a form of negation of group membership (e.g. *Everyone except for Mary visited Castlebary*).

Features were extracted using the Stanford CoreNLP system (Manning et al., 2014) and animacy information was specifically obtained through the Stanford deterministic coreference resolution module (Lee et al., 2011).

[2] http://stanfordnlp.github.io/CoreNLP/coref.html

Morphosyntax (Mention)	Grammatical (Mention)	Morphosyntax (Pairwise)	Grammatical (Pairwise)	Semantic (Pairwise)
Type = pronoun	Sentence position = first	Head match	Word distance (max. 30)	Governor = except
Type = proper noun	Sentence position = last	[Type Conjunctions]	Sentence distance	# Conjunctive pairings
Animacy = animate	Dependency = subject		Coordination = and	[# Dependency Pairings]
Animacy = unknown	Dependency = object		[Dependency Conjunctions]	
Person = first	Dependency = preposition			
Person = third				
Singular = true				
Quantified = true				
# Modifiers				

Table 5: Features used in the clustering similarity metric, separated by category. The features *[Type Conjunctions]*, *[Dependency Conjunctions]*, and *[# Dependency Pairings]* are all placeholders for feature sets. See the text for details.

6 Experiments

In order to assess the performance of our method, we conduct two experiments. In the first, we assess performance of our system on the specific *they-them* m-anaphor resolution sub-task. Our system, and its variants, are compared against a number of baseline methods based on performance on the test set.

In the second experiment, we consider how our system improves the performance of a coreference resolution system when *all* anaphors (both 1-anaphors and m-anaphors) are considered.

6.1 Evaluation

Accuracy is measured in terms of the number of mention pairs correctly grouped as m-antecedents for a given m-anaphor — similar to previous works in anaphor resolution (Peng et al., 2015). We use the standard classification metrics for precision, recall, and F1-score. If $n_1, n_2, \ldots, n_N$ represent the number of gold m-antecedents for m-anaphors $g_1, g_2, \ldots, g_N$ in a document, and $m_1, m_2, \ldots, m_N$ are predicted, of which $k_1, k_2, \ldots, k_N$ are correct, then precision is defined as $\sum_i k_i / \sum_i m_i$ and recall as $\sum_i k_i / \sum_i n_i$, where i ranges from 1 to N.

In order to align ourselves with the gold labels, we adjust the predicted mention corresponding to an entity to the closest one preceding the given m-anaphor. Because a given entity may appear multiple times in a candidate mention window, the most recent one, relative to the m-anaphor, is not always the one carrying the strongest signal and hence is not always predicted as an antecedent. For the purposes of evaluation, such cases are considered correct. Automatic handling would involve a separate, single-antecedent coreference resolver, but given the thesis of this work is the multi-antecedent case, this choice is justified.

6.2 System Comparison

We first describe the various baselines and variants of our method we assess and then analyze the performance results.

Systems

- The "most-recent-k" baselines (denoted RECENT-k), which predict the most recent k mentions, relative to the m-anaphor, as the m-antecedents for k = 2, 3, 4.

- The random selection baseline (denoted RANDOM), which randomly predicts mentions in a 5-sentence pre-window as the antecedents according to a binomial with probability 0.5 (imposing the constraint that at least two must be predicted).

- A simple rule-based method (denoted RULE) which proceeds as follows:

  – If the m-anaphor occupies a subject or prepositional position, then predict the most recent mentions in subject positions if they are coordinated, otherwise take them from previous, distinct sentences. If no such mentions can be found take the most recent mentions in subject and object positions governed by the same verb.

  – If the m-anaphor occupies in object position, take the previous mentions in object or prepositional positions if they are coordinated, otherwise take them from previous, distinct sentences. If no such mentions can be found, take the most recent mentions in subject and object positions governed by the same verb.

  – Otherwise, take the two most recent mentions (usually arrive here if there is an error in the dependency parsing).

	Precision	Recall	F1
RECENT-2	21.46	17.68	19.39
RECENT-3	23.73	30.10	26.54
RECENT-4	21.43	38.82	27.62
RANDOM	30.02	29.11	29.56
RULE	39.23	17.45	24.16
LEE	**46.78**	9.91	16.36
M-LCS	41.35	37.81	39.50
M-TREE	41.94	**44.88**	**43.36**

Table 6: Test set performance of each system on the m-anaphor resolution task.

m-anaphor class	Precison	Recall	F1
2-anaphor	48.14	52.90	50.41
3-anaphor	35.92	34.77	35.34
4-anaphor	36.74	12.87	19.06

Table 7: Performance results of the M-TREE system on the different classes of m-anaphors.

- The system described in Lee et al. (2011) (denoted LEE), which performs some light m-anaphor resolution (solely for conjunctive cases).
- The two variants of the developed method, one using the LCS similarity metric (denoted M-LCS) and the other using the subset tree kernel (M-TREE).

Results and Discussion

Accuracy results on the test set for each of the systems are given in Table 6. Both the proposed systems, M-LCS and M-TREE, outperform all other methods by a substantial margin. The Stanford system achieves the highest precision, which is not surprising because it targets conjunctive mentions, which often serve as m-antecedents. Based on the analysis of Section 4, the poor performance of RECENT-2, RECENT-3, and RECENT-4 is expected.

The results for the best-performing system, M-TREE, on the different classes of m-anaphors is given in Table 7. M-TREE outperforms all other systems but exhibits a bias towards 2-anaphors, recent mentions, and mentions coordinated by conjunction. This is not surprising given such cases are the easiest to resolve.

6.3 Full Coreference Resolution

For the complete coreference resolution task, the M-TREE system can be integrated with an existing coreference system. For this experiment, we pair the full coreference resolution system of Clark and Manning (2015) with M-TREE, and we raise the prediction threshold of our model to 0.89, at which point precision on the validation set is 78.9. Moreover, we restrict ourselves to the P&P portion of the test set, given the Scribner stories only have gold labels for instances of *they* and *them*.

	MUC	B^3	CEAF_e	Avg.
CLARK	42.3	39.5	32.4	38.1
CLARK+M-TREE	**43.4**	**40.0**	**31.9**	**38.7**

Table 8: CoNLL metric scores for coreference resolution on the test portion of P&P for the Clark and Manning (2015) system, with (CLARK+M-TREE) and without (CLARK) the pairing with M-TREE.

The Clark and Manning (2015) system is first run over the test set, producing coreference chains which are then filtered for character entities using the approach of Vala et al. (2015). Our adjusted M-TREE system is then applied over all *they* and *them* mentions. Each such mention predicted as m-anaphoric is added to the coreference chains of the entities corresponding to the m-antecedent mentions.

To evaluate the accuracy against the gold mention clusters, each m-anaphoric *they* and *them* is added to each cluster containing a gold m-antecedent. The CoNLL metric scores (Bagga and Baldwin, 1998) of the coreference predictions are shown in Table 8, with the integrated system outperforming the Clark and Manning (2015) system by 0.6 average score (pairing the Clark and Manning (2015) system instead with an oracle m-anaphor resolver yields an average score of 44.8, an increase of 6.7 points).

7 Related Work

The formal problem statement for the noun phrase anaphor resolution we propose is an extension of the standard ACE (Doddington et al., 2004), MUC (Hirschman, 1997), and Ontonotes (Hovy et al., 2006) formulations, as well as the problem settings outlined in Wiseman et al. (2015) and Durrett and Klein (2013), to allow anaphors to link to multiple antecedents. Most previous works impose the constraint that anaphors can be assigned at most one antecedent. Some works cast the coreference resolution problem in an Integer Linear Programming framework, with an explicit constraint for

assigning at most one antecedent to an anaphor (Peng et al., 2015; Denis et al., 2007).

The early work of Ingria and Stallard (1989) proposes the resolution of pronouns without the restriction they be linked to at most one antecedent. The method uses an indexing scheme for parse trees, similar to Hobb's algorithm (Hobbs, 1978), that eliminates candidates antecedents as more information is acquired. Those pronouns with multiple candidates remaining after tree-traversal are predicted as m-anaphors. The method considers each parse tree in isolation, and hence does not permit inter-sentential linking, a severe limitation in corpora such as the one offered in this work.

Other researchers have evaluated noun phrase coreference resolvers along a number of dimensions, including different classes of anaphors (Mitkov, 2014; Kummerfeld and Klein, 2013; Björkelund and Nugues, 2011; Recasens and Hovy, 2009; Stoyanov et al., 2009; Bengtson and Roth, 2008; Van Deemter and Kibble, 2000; Ng and Cardie, 2002b; Kameyama, 1997). This work explores a new class of anaphor, previously unstudied, and evaluates its impact on the coreference resolution problem.

Many state-of-the-art systems for coreference resolution, especially supervised, are constrained to the single-antecedent case (Clark and Manning, 2015; Peng et al., 2015; Wiseman et al., 2015; Björkelund and Farkas, 2012; Ng, 2010; Stoyanov et al., 2010; Ng, 2008; Soon et al., 2001). The most well-known, benchmark datasets for coreference resolution (e.g. Ontonotes and ACE-2005), do not offer gold annotations for multi-antecendet anaphors. Our work presents the first dataset for tackling this problem.

The Lee et al. (2011) is a deterministic system that attempts to resolve the "easy" multi-antecedent cases, namely those in which mentions are joined by some conjunction. Our system goes beyond and attempts to predict more difficult cases as well.

Many of the individual features we employ in our model appear in a variety of other coreference systems, especially those involving mention-pair models (Durrett and Klein, 2013; Recasens et al., 2013b; Stoyanov et al., 2010). Recasens et al. (2013a) attempts to perform coreference resolution under conditions where many standard features for coreference are not suited. Peng et al. (2015) resort to corpus counts of predicates as features, much in the same way we obtain counts of mention pairings according to simple predicates on dependency structures.

The system of Clark and Manning (2015) also makes uses of agglomerative clustering, although it's employed in merging coreference chains, rather than candidate antecedent groupings.

Last, resorting to an external corpus for sentence structures is common practice in the Natural Language Generation literature for producing phrases that are coherent and consistent(Krishnamoorthy et al., 2013; Bangalore and Rambow, 2000; Langkilde and Knight, 1998).

8 Conclusion

We introduced a new class of anaphors to the anaphor resolution problem, m-anaphors, and extended the problem formulation to incorporate them. We offered insights into the linguistic behaviour of m-anaphors, finding that surface-level syntactic and semantic features do not carry enough discriminative power in distinguishing them from 1-anaphors. Furthermore, we developed a system combining a mention-pair model, an existing coreference resolver, and corpus knowledge to resolve m-anaphors that scores higher than a number of baseline methods. Finally, we paired this system with a coreference resolver to solve the general coreference resolution task, showing that m-anaphor prediction can help boost performance.

An important component of the m-anaphor resolution problem that falls outside the scope of this study, but is important for practical application, is the detection of m-anaphoric mentions. Section 4 gives some insight into the problem but a much deeper investigation is necessary to devise a detection method.

Moreover, for simplicity, this study focused solely on m-anaphoric *they* and *them* mentions, but as explained earlier, m-anaphoric mentions can take many forms, each introducing their own particular complexities that warrant special attention.

Regarding the system developed for m-anaphor resolution, resorting to an external corpus to obtain well-formed sentences proved to be very computationally expensive. In future work, we look to incorporate methods that incur less cost, possibly tolerating some error in the formation of

sentences without significantly degrading performance. Also, negation of group membership is a complex linguistic phenomenon that was handled in a crude manner in our system. We look to devote future work to handling such cases.

To promote further research into m-anaphors, we make all our data and software freely available at `http://www.github.com/networkdynamics/manaphor-acl2016`.

References

Amit Bagga and Breck Baldwin. 1998. Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 42–48. Association for Computational Linguistics.

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 294–303. Association for Computational Linguistics.

Shane Bergsma and Dekang Lin. 2006. Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 33–40. Association for Computational Linguistics.

Anders Björkelund and Richárd Farkas. 2012. Data-driven multilingual coreference resolution using resolver stacking. In *Joint Conference on EMNLP and CoNLL-Shared Task*, pages 49–55. Association for Computational Linguistics.

Anders Björkelund and Pierre Nugues. 2011. Exploring lexicalized features for coreference resolution. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 45–50. Association for Computational Linguistics.

Kevin Clark and Christopher D Manning. 2015. Entity-centric coreference resolution with model stacking. In *Association of Computational Linguistics (ACL)*.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Pascal Denis, Jason Baldridge, et al. 2007. Joint determination of anaphoricity and coreference resolution using integer programming. In *HLT-NAACL*, pages 236–243. Citeseer.

George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. 2004. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, volume 2, page 1.

Greg Durrett and Dan Klein. 2013. Easy victories and uphill battles in coreference resolution. In *EMNLP*, pages 1971–1982.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Lynette Hirschman. 1997. {MUC-7 Coreference Task Definition}.

Jerry R Hobbs. 1978. Resolving pronoun references. *Lingua*, 44(4):311–338.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: the 90% solution. In *Proceedings of the human language technology conference of the NAACL, Companion Volume: Short Papers*, pages 57–60. Association for Computational Linguistics.

Robert JP Ingria and David Stallard. 1989. A computational mechanism for pronominal reference. In *Proceedings of the 27th annual meeting on Association for Computational Linguistics*, pages 262–271. Association for Computational Linguistics.

Megumi Kameyama. 1997. Recognizing referential links: An information extraction perspective. In *Proceedings of a Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, pages 46–53. Association for Computational Linguistics.

Niveda Krishnamoorthy, Girish Malkarnenkar, Raymond J Mooney, Kate Saenko, and Sergio Guadarrama. 2013. Generating natural-language video descriptions using text-mined knowledge. In *AAAI*, volume 1, page 2.

Jonathan K Kummerfeld and Dan Klein. 2013. Error-driven analysis of challenges in coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 265–277.

Irene Langkilde and Kevin Knight. 1998. Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Compu-*

tational Linguistics-Volume 1, pages 704–710. Association for Computational Linguistics.

Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2011. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task*, pages 28–34. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Michael Martone, Lex Williford, and Rosellen Brown. 1999. *The Scribner Anthology of Contemporary Short Fiction: Fifty North American Stories Since 1970*. Touchstone.

Ruslan Mitkov. 2014. *Anaphora resolution*. Routledge.

Alessandro Moschitti. 2006. Making tree kernels practical for natural language learning. In *EACL*, volume 113, page 24.

Vincent Ng and Claire Cardie. 2002a. Identifying anaphoric and non-anaphoric noun phrases to improve coreference resolution. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.

Vincent Ng and Claire Cardie. 2002b. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 104–111. Association for Computational Linguistics.

Vincent Ng. 2008. Unsupervised models for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 640–649. Association for Computational Linguistics.

Vincent Ng. 2010. Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the 48th annual meeting of the association for computational linguistics*, pages 1396–1411. Association for Computational Linguistics.

Haoruo Peng, Daniel Khashabi, and Dan Roth. 2015. Solving hard coreference problems. *Urbana*, 51:61801.

Marta Recasens and Eduard Hovy. 2009. A deeper look into features for coreference resolution. In *Anaphora Processing and Applications*, pages 29–42. Springer.

Marta Recasens, Matthew Can, and Daniel Jurafsky. 2013a. Same referent, different words: Unsupervised mining of opaque coreferent mentions. In *HLT-NAACL*, pages 897–906.

Marta Recasens, Marie-Catherine de Marneffe, and Christopher Potts. 2013b. The life and death of discourse entities: Identifying singleton mentions. In *HLT-NAACL*, pages 627–633.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational linguistics*, 27(4):521–544.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations Session at EACL 2012*, Avignon, France, April. Association for Computational Linguistics.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff. 2009. Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 656–664. Association for Computational Linguistics.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom. 2010. Reconcile: A coreference resolution research platform.

Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. 2015. Mr. bennet, his coachman, and the archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 769–774.

Kees Van Deemter and Rodger Kibble. 2000. On coreferring: Coreference in muc and related annotation schemes. *Computational linguistics*, 26(4):629–637.

Sam Wiseman, Alexander M Rush, Stuart M Shieber, Jason Weston, Heather Pon-Barry, Stuart M Shieber, Nicholas Longenbaugh, Sam Wiseman, Stuart M Shieber, Elif Yamangil, et al. 2015. Learning anaphoricity and antecedent ranking features for coreference resolution. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 92–100. Association for Computational Linguistics.

Automatic Labeling of Topic Models Using Text Summaries

Xiaojun Wan and Tianming Wang
Institute of Computer Science and Technology, The MOE Key Laboratory of Computational Linguistics, Peking University, Beijing 100871, China
{wanxiaojun, wangtm}@pku.edu.cn

Abstract

Labeling topics learned by topic models is a challenging problem. Previous studies have used words, phrases and images to label topics. In this paper, we propose to use text summaries for topic labeling. Several sentences are extracted from the most related documents to form the summary for each topic. In order to obtain summaries with both high relevance, coverage and discrimination for all the topics, we propose an algorithm based on submodular optimization. Both automatic and manual analysis have been conducted on two real document collections, and we find 1) the summaries extracted by our proposed algorithm are superior over the summaries extracted by existing popular summarization methods; 2) the use of summaries as labels has obvious advantages over the use of words and phrases.

1 Introduction

Statistical topic modelling plays very important roles in many research areas, such as text mining, natural language processing and information retrieval. Popular topic modeling techniques include Latent Dirichlet Allocation (LDA) (Blei et al., 2003) and Probabilistic Latent Semantic Analysis (pLSA) (Hofmann, 1999). These techniques can automatically discover the abstract "topics" that occur in a collection of documents. They model the documents as a mixture of topics, and each topic is modeled as a probability distribution over words.

Although the discovered topics' word distributions are sometimes intuitively meaningful, a major challenge shared by all such topic models is to accurately interpret the meaning of each topic (Mei et al., 2007). The interpretation of each topic is very important when people want to browse, understand and leverage the topic. However, it is usually very hard for a user to understand the discovered topics based only on the multinomial distribution of words. For example, here are the top terms for a discovered topic: {*fire miles area north southern people coast homes south damage northern river state friday central water rain high california weather*}. It is not easy for a user to fully understand this topic if the user is not very familiar with the document collection. The situation may become worse when the user faces with a number of discovered topics and the sets of top terms of the topics are often overlapping with each other on many practical document collections.

In order to address the above challenge, a few previous studies have proposed to use phrases, concepts and even images for labeling the discovered topics (Mei et al., 2007; Lau et al., 2011; Hulpus et al., 2013; Aletras and Stevenson, 2013). For example, we may automatically extract the phrase "*southern california*" to represent the example topic mentioned earlier. These topic labels can help the user to understand the topics to some extent. However, the use of phrases or concepts as topic labels are not very satisfactory in practice, because the phrases or concepts are still very short, and the information expressed in these short labels is not adequate for user's understanding. The case will become worse when some ambiguous phrase is used or multiple discrete phrases with poor coherence are used for a topic. To address the drawbacks of the above short labels, we need to provide more contextual information and consider using long text descriptions to represent the topics. The long text descriptions can be used independently or used as beneficial complement to the short labels. For example, below is part of the summary label produced by our proposed method and it provides much more contextual information for understanding the topic.

> *Showers and thunderstorms developed in parched areas of the southeast , from western north carolina into south central alabama , north central and northeast texas and the central and southern gulf coast . … The quake was felt over a*

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2297–2305,
Berlin, Germany, August 7-12, 2016.

large area , extending from santa rosa , about 60 miles north of san francisco , to the santa cruz area 70 miles to the south .... Fourteen homes were destroyed in baldwin park 20 miles northeast of downtown los angeles and five were damaged along with five commercial buildings when 75 mph gusts snapped power lines , igniting a fire at allan paper co. , fire officials said . ...

The contributions of this paper are summarized as follows:

1) We are the first to invesitage using text summaries for topic labeling;

2) We propose a summarization algorithm based on submodular optimization to extract summaries with both high relevance, coverage and discrimination for all topics.

3) Automatic and manual analysis reveals the usefulness and advantages of the summaries produced by our algorithm.

2 Related Work

2.1 Topic Labeling

After topics are discovered by topic modeling techniques, these topics are conventionally represented by their top *N* words or terms (Blei et al., 2003; Griffiths and Steyvers, 2004). The words or terms in a topic are ranked based on the conditional probability $p(w_i|t_j)$ in that topic. It is sometimes not easy for users to understand each topic based on the terms. Sometimes topics are presented with manual labeling for exploring research publications (Wang and McCallum, 2006; Mei et al., 2006), and the labeling process is time consuming.

In order to make the topic representations more interpretable and make the topics easier to understand, there are a few studies proposing to automatically find phrases, concepts or even images for topic labeling. Mei et al. (2007) proposed to use phrases (chunks or ngrams) for topic labeling and cast the labeling problem as an optimization problem involving minimizing Kullback-Leibler (KL) divergence between word distributions and maximizing mutual information between a label and a topic model. Lau et al. (2011) also used phrases as topic labels and they proposed to use supervised learning techniques for ranking candidate labels. In their work, candidate labels include the top-5 topic terms and a few noun chunks extracted from related Wikipedia articles. Mao et al. (2012) proposed two effective algorithms that automatically assign concise labels to each topic in a hierarchy by exploiting sibling and parent-child relations among topics. Kou et al. (2015) proposed to map topics and candidate labels (phrases) to word vectors and letter trigram vectors in order to find which candidate label is more semantically related to that topic. Hulpus et al. (2013) took a new approach based on graph centrality measures to topic labelling by making use of structured data exposed by DBpedia. Different from the above works, Aletras and Stevenson (2013) proposed to use images for representing topics, where candidate images for each topic are retrieved from the web and the most suitable image is selected by using a graph-based algorithm. In a very recent study (Aletras et al., 2015), 3 different topic representations (lists of terms, textual phrase labels and images labels) are compared in a document retrieval task, and results show that textual phrase labels are easier for users to interpret than term lists and image labels.

The phrase-based labels in the above works are still very short and are sometimes not adequate for interpreting the topics. Unfortunately, none of previous works has investigated using textual summaries for representing topics yet.

2.2 Document Summarization

The task of document summarization aims to produce a summary with a length limit for a given document or document set. The task has been extensively investigated in the natural language processing and information retrieval fields, and most previous works focus on directly extracting sentences from a news document or collection to form the summary. The summary can be used for helping users quickly browse and understand a document or document collection.

Typical multi-document summarization methods include the centroid-based method (Radev et al., 2004), integer linear programming (ILP) (Gillick et al., 2008), sentence-based LDA (Chang and Chien, 2009), submodular function maximization (Lin and Bilmes, 2010; Lin and Bilmes, 2011), graph based methods (Erkan and Radev, 2004; Wan et al., 2007; Wan and Yang, 2008), and supervised learning based methods (Ouyang et al., 2007; Shen et al., 2007). Though different summarization methods have been proposed in recent years, the submodular function maximization method is still one of the state-of-the-art summarization methods. Moreover, the method is easy to follow and its framework is very flexible. One can design specific submodular functions for addressing special summarization tasks, without altering the overall greedy selection framework.

Though various summarization methods have been proposed, none of existing works has investigated or tried to adapt document summarization techniques for the task of automatic labeling of topic models.

3 Problem Formulation

Given a set of latent topics extracted from a text collection and each topic is represented by a multinomial distribution over words, our goal is to produce understandable text summaries as labels for interpreting all the topics. We now give two useful definitions for later use.

Topic: Each topic θ is a probability distribution of words $\{p_\theta(w)\}_{w\in V}$, where V is the vocabulary set, and we have $\sum_{w\in V} p_\theta(w) = 1$.

Topic Summary: In this study, a summary for each topic θ is a set of sentences extracted from the document collection and it can be used as a label to represent the latent meaning of θ. Typically, the length of the summary is limited to 250 words, as defined in recent DUC and TAC conferences.

Like the criteria for the topic labels in (Mei et al., 2007), the topic summary for each topic needs to meet the following two criteria:

High Relevance: The summary needs to be semantically relevant to the topic, i.e., the summary needs to be closely relevant to all representative documents of the topic. The higher the relevance is, the better the summary is. This criterion is intuitive because we do not expect to obtain a summary unrelated to the topic.

High Coverage: The summary needs to cover as much semantic information of the topic as possible. The summary usually consists of several sentences, and we do not expect all the sentences to focus on the same piece of semantic information. A summary with high coverage will certainly not contain redundant information. This criterion is very similar to the diversity requirement of multi-document summarization.

Since we usually produce a set of summaries for all the topics discovered in a document collection. In order to facilitate users to understand all the topics, the summaries need to meet the following additional criterion:

High Discrimination: The summaries for different topics need to have inter-topic discrimination. If the summaries for two or more topics are very similar with each other, users can hardly understand each topic appropriately. The higher the inter-topic discrimination is, the better the summaries are.

4 Our Method

Our proposed method is based on submodular optimization, and it can extract summaries with both high relevance, coverage and discrimination for all topics. We choose the framework of submodular optimization because the framework is very flexible and different objectives can be easily incorporated into the framework. The overall framework of our method consists of two phases: candidate sentence selection, and topic summary extraction. The two phrases are described in the next two subsections, respectively.

4.1 Candidate Sentence Selection

There are usually many thousands of sentences in a document collection for topic modelling, and all the sentences are more or less correlated with each topic. If we use all the sentences for summary extraction, the summarization efficiency will be very low. Moreover, many sentences are not suitable for summarization because of their low relevance with the topic. Therefore, we filter out the large number of unrelated sentences and treat the remaining sentences as candidates for summary extraction.

For each topic θ, we compute the Kullback-Leibler (KL) divergence between the word distributions of the topic and each sentence s in the whole document collection as follows:

$$KL(\theta, s) = \sum_{w\in TW\cup SW} p_\theta(w) * log \frac{p_\theta(w)}{tf(w,s)/len(s)}$$

where $p_\theta(w)$ is the probability of word w in topic θ. TW denotes the set of top 500 words in topic θ according to the probability distribution. SW denotes the set of words in sentence s after removing stop words. $tf(w,s)$ denotes the frequency of word w in sentence s, and $len(s)$ denotes the length of sentence s after removing stop words. For a word w which does not appear in SW, we set $tf(w,s)/len(s)$ to a very small value (0.00001 in this study).

Then we rank the sentences by an increasing order of the divergence scores and keep the top 500 sentences which are most related to the topic. These 500 sentences are treated as candidate sentences for the subsequent summarization step for each topic. Note that different topics have different candidate sentence sets.

4.2 Topic Summary Extraction

Our method for topic summary extraction is based on submodular optimization. For each topic θ associated with the candidate sentence set V, our

method aims to find an optimal summary $\tilde{E}$ from all possible summaries by maximizing a score function under budget constraint:

$$\tilde{E} = argmax_{E \subseteq V}\{f(E)\} \quad \text{s.t. } len(E) \leq L$$

where $len(E)$ denotes the length of summary *E*. Here *E* is also used to denote the set of sentences in the summary. *L* is a predefined length limit, i.e. 250 words in this study.

$f(E)$ is the score function to evaluate the overall quality of summary *E*. Usually, $f(E)$ is required to be a submodular function, so that we can use a simple greedy algorithm to find the near-optimal summary with theoretical guarantee. Formally, for any $A \subseteq B \subseteq V \backslash v$, we have

$$f(A + v) - f(A) \geq f(B + v) - f(B)$$

which means that the incremental "value" of *v* decreases as the context in which *v* is considered grows from *A* to *B*.

In this study, the score function $f(E)$ is decomposed into three parts and each part evaluates one aspect of the summary:

$$f(E) = REL(E) + COV(E) + DIS(E)$$

where $REL(E)$, $COV(E)$ and $DIS(E)$ evaluate the relevance, coverage and discrimination of summary *E* respectively. We will describe them in details respectively.

4.2.1 Relevance Function

Instead of intuitively measuring relevance between the summary and the topic via the KL divergence between the word distributions of them, we consider to measure the relevance of summary *E* for topic θ by the relevance of the sentences in the summary to all the candidate sentences for the topic as follows:

$$REL(E) = \sum_{s' \in V} \min\{\sum_{s \in E} sim(s', s), \alpha \sum_{s \in V} sim(s', s)\}$$

where *V* represents the candidate sentence set for topic θ, and *E* is used to represent the sentence set of the summary. $sim(s', s)$ is the standard cosine similarity between sentences s' and s. $\alpha \in [0,1]$ is a threshold co-efficient.

The above function is a monotone submodular function because $f(x) = min(x, a)$ where $a \geq 0$ is a concave non-decreasing function. $\sum_{s \in E} sim(s', s)$ measures how similar *E* is to sentence s' and then $\sum_{s \in V} sim(s', s)$ is the largest value that $\sum_{s \in E} sim(s', s)$ can achieve. Therefore, s' is saturated by *E* when $\sum_{s \in E} sim(s', s) \geq \alpha \sum_{s \in V} sim(s', s)$. When $s'$is already saturated by *E* in this way, any new sentence very similar to s' cannot further improve the overall relevance of *E*, and this sentence is less possible to be added to the summary.

4.2.2 Coverage Function

We want the summary to cover as many topic words as possible and contain as many different sentences as possible. The coverage function is thus defined as follows:

$$COV(E) = \beta * \sum_{w \in TW} \left\{ p_\theta(w) * \sqrt{\sum_{s \in E} tf(w, s)} \right\}$$

where $\beta \geq 0$ is a combination co-efficient.

The above function is a monotone submodular function and it encourages the summary *E* to contain many different words, rather than a small set of words. Because $f(x) = \sqrt{x}$ where $x \geq 0$ is a concave non-decreasing function, we have $f(x + y) \leq f(x) + f(y)$. The value of the function will be larger when we use *x* and *y* to represent two frequency values of two different words respectively than that when we use $(x + y)$ to represent the frequency value of a single word. Therefore, the use of this function encourages the coverage of more different words in the summary. In other words, the diversity of the summary is enhanced.

4.2.3 Discrimination Function

The function for measuring the discrimination between the summary *E* of topic θ and all other topics $\{\theta'\}$ is defined as follows:

$$DIS(E) = -\gamma \sum_{\theta'} \sum_{s \in E} \sum_{w \in TW} p_{\theta'}(w) * tf(w, s)$$

where $\gamma \geq 0$ is a combination co-efficient.

The above function is still a monotone submodular function. The negative sign indicates that the summary *E* of topic θ needs to be as irrelevant with any other topic as possible, and thus making different topic summaries have much differences.

4.2.4 Greedy Selection

Since $REL(E)$, $COV(E)$ and $DIS(E)$ are all submodular functions, $f(E)$ is also a submodular function. In order to find a good approximation to the optimal summary, we use a greedy algorithm similar to (Lin and Bilmes, 2010) to select sentence one by one and produce the final summary, as shown in Algorithm 1.

Algorithm 1 Greedy algorithm for summary extraction

1: $E \leftarrow \emptyset$
2: $U \leftarrow V$
3: **while** $U \neq \emptyset$ **do**
4: $\hat{s} \leftarrow argmax_{s \in U} \frac{f(E \cup \{s\}) - f(E)}{len(s)^{\varepsilon}}$
5: $E \leftarrow E \cup \{\hat{s}\}$ if $\sum_{s \in E} len(s) + len(\hat{s}) \leq L$ and $f(E \cup \{s\}) - f(E) \geq 0$
6: $U \leftarrow U \setminus \{\hat{s}\}$
7: **end while**
8: return E

In the algorithm, $len(s)$ denotes the length of sentence *s* and $\varepsilon > 0$ is the scaling factor. At each iteration, the sentence with the largest ratio of objective function gain to scaled cost is found in step 4, and if adding the sentence can increase the objective function value while not violating the length constraint, it is then selected into the summary and otherwise bypassed.

5 Evaluation and Results

5.1 Evaluation Setup

We used two document collections as evaluation datasets, as in (Mei et al. 2007): AP news and SIGMOD proceedings. The AP news dataset contains a set of 2250 AP news articles, which are provided by TREC. There is a total of 43803 sentences in the AP news dataset and the vocabulary size is 37547 (after removing stop words). The SIGMOD proceeding dataset contains a set of 2128 abstracts of SIGMOD proceedings between the year 1976 and 2015, downloaded from the ACM digital library. There is a total of 15211sentences in the SIGMOD proceeding dataset and the vocabulary size is 13688.

For topic modeling, we adopted the most popular LDA to discover topics in the two datasets, respectively. Particularly, we used the LDA module implemented in the MALLET toolkit[1]. Without loss of generality, we extracted 25 topics from the AP news dataset and 25 topics from the SIGMOD proceeding dataset.

The parameter values of our proposed summarization method is either directly borrowed from previous works or empirically set as follows: $\alpha = 0.05$, $\beta = 250$, $\gamma = 300$ and $\varepsilon = 0.15$.

We have two goals in the evaluation: comparison of different summarization methods for topic labeling, and comparison of different kinds of labels (summaries, words, and phrases).

In particular, we compare our proposed summarization method (denoted as **Our Method**) with the following typical summarization methods and all of them extract summaries from the same candidate sentence set for each topic:

MEAD: It uses a heuristic way to obtain each sentence's score by summing the scores based on different features (Radev et al., 2004): centroid-based weight, position and similarity with first sentence.

LexRank: It constructs a graph based on the sentences and their similarity relationships and then applies the PageRank algorithm for sentence ranking (Erkan and Radev, 2004).

TopicLexRank: It is an improved version of LexRank by considering the probability distribution of top 500 words in a topic as a prior vector, and then applies the topic-sensitive PageRank algorithm for sentence ranking, similar to (Wan 2008).

Submodular(REL): It is based on submodular function maximization but only the relevance function is considered.

Submodular(REL+COV): It is based on submodular function maximization and combines two functions: the relevance function and the coverage function.

We also compare the following three different kinds of labels:

Word label: It shows ten topic words as labels for each topic, which is the most intuitive interpretation of the topic.

Phrase label: It uses three phrases as labels for each topic, and the phrase labels are extracted by using the method proposed in (Mei et al., 2007), which is very closely related to our work and considered a strong baseline in this study.

Summary Label: It uses a topic summary with a length of 250 words to label each topic and the summary is produced by our proposed method.

5.2 Evaluation Results

5.2.1 Automatic Comparison of Summarization Methods

In this section, we compare different summarization methods with the following automatic measures:

[1] http://mallet.cs.umass.edu/

KL divergence between word distributions of summary and topic: For each summarization method, we compute the KL divergence between the word distributions of each topic and the summary for the topic, then average the KL divergence across all topics. Table 1 shows the results. We can see that our method and Submodular(REL+COV) have the lowest KL divergence with the topic, which means our method can produce summaries relevant to the topic representation.

Topic word coverage: For each summarization method, we compute the ratio of the words covered by the summary out of top 20 words for each topic, and then average the ratio across all topics. We use top 20 words instead of 500 words because we want to focus on the most important words. The results are shown in Table 2. We can see that our method has almost the best coverage ratio and the produced summary can cover most important words in a topic.

	AP	SIGMOD
MEAD	0.832503	1.470307
LexRank	0.420137	1.153163
TopicLexRank	0.377587	1.112623
Submodular(REL)	0.43264	1.002964
Submodular(REL+COV)	0.349807	0.991071
Our Method	0.360306	0.907193

Table 1. Comparison of KL divergence between word distributions of summary and topic

	AP	SIGMOD
MEAD	0.422246	0.611355
LexRank	0.651217	0.681728
TopicLexRank	0.678515	0.692066
Submodular(REL)	0.62815	0.713159
Submodular(REL+COV)	0.683998	0.723228
Our Method	0.673585	0.74572

Table 2. Comparison of the ratio of the covered words out of top 20 topic words

	AP		SIGMOD	
	average	max	average	max
MEAD	0.026961	0.546618	0.078826	0.580055
LexRank	0.019466	0.252074	0.05635	0.357491
TopicLexRank	0.022548	0.283742	0.062034	0.536886
Submodular(REL)	0.028035	0.47012	0.07522	0.52629
Submodular (REL+COV)	0.023206	0.362795	0.048872	0.524863
Our Method	0.010304	0.093017	0.024551	0.116905

Table 3. Comparison of the average and max similarity between different topic summaries

Similarity between topic summaries: For each summarization method, we compute the cosine similarity between the summaries of any two topics, and then obtain the average similarity and the maximum similarity. Seen from Table 3, the topic summaries produced by our method has the lowest average and maximum similarity with each other, and thus the summaries for different topics have much difference.

5.2.2 Manual Comparison of Summarization Methods

In this section, we compare our summarization method with three typical summarization methods (MEAD, TopicLexRank and Submodular(REL)) manually. We employed three human judges to read and rank the four summaries produced for each topic by the four methods in three aspects: relevance between the summary and the topic with the corresponding sentence set, the content coverage (or diversity) in the summary and the discrimination between different summaries. The human judges were encouraged to read a few closely related documents for better understanding each topic. Note that the judges did not know which summary was generated by our method and which summaries were generated by the baseline methods. The rank k for each summary ranges from 1 to 4 (1 means the best, and 4 means the worst; we allow equal ranks), and the score is thus (4-k). We average the scores across all summaries and all judges and the results on the two datasets are shown in Tables 4 and 5, respectively. In the table, the higher the score is, the better the corresponding summaries are. We can see that our proposed method outperforms all the three baselines over almost all metrics.

	relevance	coverage	discrimination
MEAD	1.03	0.8	1.13
TopicLexRank	1.9	1.6	1.83
Submodular(REL)	2.23	2	2.07
Our Method	2.33	2.4	2.33

Table 4. Manual comparison of different summarization methods on AP news dataset

	relevance	coverage	discrimination
MEAD	1.6	1.4	1.83
TopicLexRank	1.77	2.1	2.1
Submodular(REL)	2.07	2.1	2.03
Our Method	2.43	2.17	2.1

Table 5. Manual comparison of different summarization methods on SIGMOD proceeding dataset

5.2.3 Manual Comparison of Different Kinds of Labels

In this section, we manually compare the three kinds of labels: words, phrases and summary, as

mentioned in Section 5.1. Similarly, the three human judges were asked to read and rank the three kinds of labels in the same three aspects: relevance between the label and the topic with the corresponding sentence set, the content coverage (or diversity) in the label and the discrimination between different labels. The rank *k* for each kind of labels ranges from 1 to 3 (1 means the best, and 3 means the worst; we allow equal ranks), and the score is thus (3-*k*). We average the scores across all labels and all judges and the results on the two datasets are shown in Tables 6 and 7, respectively. It is clear that the summary labels produced by our proposed method have obvious advantages over the conventional word labels and phrase labels. The summary labels have better evaluation results on relevance, coverage and discrimination.

	relevance	coverage	discrimination
Word label	0.67	0.67	1.11
Phrase label	1	0.87	1.4
Summary label	1.83	1.87	1.9

Table 6. Manual comparison of different kinds of labels on AP news dataset

	relevance	coverage	discrimination
Word label	0.87	0.877	1.27
Phrase label	1.4	1.53	1.43
Summary label	1.8	1.97	1.9

Table 7. Manual comparison of different kinds of labels on AP news dataset

5.2.4 Example Analysis

In this section, we demonstrate some running examples on the SIGMOD proceeding dataset. Two topics and the three kinds of labels are shown below. For brevity, we only show the first 100 words of the summaries to users unless they want to see more. We can see that the word labels are very confusing, and the phrase labels for the two topics are totally overlapping with each other and have no discrimination. Therefore, it is hard to understand the two topics by looking at the word or phrase labels. Fortunately, by carefully reading the topic summaries, we can understand what the two topics are really about. In this example, the first topic is about data analysis and data integration, while the second topic is about data privacy. Though the summary labels are much longer than the word labels or phrase labels, users can obtain more reliable information after reading the summary labels and the summaries can help users to better understand each topic and also know the difference between different topics.

In practice, the different kinds of labels can be used together to allow users to browse topic models in a level-wise matter, as described in next section.

Topic 1 on SIGMOD proceeding dataset:

word label: *data analysis scientific set process analyze tool insight interest scenario*

phrase label: *data analysis ; data integration ; data set*

summary label: *The field of data analysis seek to extract value from data for either business or scientific benefit . ... Nowadays data analytic application are accessing more and more data from distributed data store , creating a large amount of data traffic on the network . ...these service will access data from different data source type and potentially need to aggregate data from different data source type with different data format ....Various data model will be discussed , including relational data , xml data , graph-structured data , data stream , and workflow ....*

Topic 2 on SIGMOD proceeding dataset:

word label: *user information attribute model privacy quality record result individual provide*

phrase label: *data set ; data analysis ; data integration*

summary label: *An essential element for privacy metric is the measure of how much adversaries can know about an individual ' sensitive attribute (sa) if they know the individual ' quasi-identifier (qi) ....We present an automated solution that elicit user preference on attribute and value , employing different disambiguation technique ranging from simple keyword matching , to more sophisticated probabilistic model ....Privgene need significantly less perturbation than previous method , and it achieve higher overall result quality , even for model fitting task where ga is not the first choice without privacy consideration ....*

5.2.5 Discussion of Practical Use

Although the summary labels produced by our method have higher relevance, coverage and discrimination than the word labels and the phrase labels, the summary labels have one obvious shortcoming of consuming more reading time of users, because the summaries are much longer than the words and phrases. The feedback from the human judges also reveals the above problem and all the three human judges said they need to take more than five times longer to read the summaries. Therefore, we want to find a better way to make use of the summary label in practice.

In order to consider both the shorter reading time of the phrase labels and the better quality of

the summary labels, we can use both of the two kinds of labels in the following hierarchical way:

For each topic, we first present only the phrase label to users, and if they can easily know about the topic after they read the phrase label, the summary label will not be shown to them. Whereas, if users cannot know well about the topic based on the phrase label, or they need more information about the topic, they may choose to read the summary label for better understanding the topic. Only the first 100 words of the summary label are shown to users, and the rest words will be shown upon request. In this way, the summary label is used as an important complement to the phrase label, and the burden of reading the longer summary label can be greatly alleviated.

6 Conclusions and Future Work

In this study, we addressed the problem of topic labeling by using text summaries. We propose a summarization algorithm based on submodular optimization to extract representative summaries for all the topics. Evaluation results demonstrate that the summaries produced by our proposed algorithm have high relevance, coverage and discrimination, and the use of summaries as labels has obvious advantages over the use of words and phrases.

In future work, we will explore to make use of all the three kinds of labels together to improve the users' experience when they want to browse, understand and leverage the topics.

In this study, we do not consider the coherence of the topic summaries because it is really very challenging to get a coherent summary by extracting different sentences from a large set of different documents. In future work, we will try to make the summary label more coherent by considering the discourse structure of the summary and leveraging sentence ordering techniques.

Acknowledgments

The work was supported by National Natural Science Foundation of China (61331011), National Hi-Tech Research and Development Program (863 Program) of China (2015AA015403) and IBM Global Faculty Award Program. We thank the anonymous reviewers and mentor for their helpful comments.

References

Nikolaos Aletras, and Mark Stevenson. 2013. Representing topics using images. *HLT-NAACL*.

Nikolaos Aletras, Timothy Baldwin, Jey Han Lau, and Mark Stevenson. 2015. Evaluating topic representations for exploring document collections. *Journal of the Association for Information Science and Technology* (2015).

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of machine Learning research* 3: 993-1022.

Ying-Lang Chang and Jen-Tzung Chien. 2009. Latent Dirichlet learning for document summarization. *Proccedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP2009)*.

Güneş Erkan and Dragomir R. Radev. 2004. LexPageRank: Prestige in multi-document text summarization. In *Proceedings of EMNLP*.

Dan Gillick, Benoit Favre, and Dilek Hakkani-Tur. 2008. The ICSI summarization system at TAC 2008. In *Proceedings of the Text Understanding Conference*.

Thomas Hofmann. 1999. Probabilistic latent semantic indexing. *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Ioana Hulpus, Conor Hayes, Marcel Karnstedt, and Derek Greene. 2013. Unsupervised graph-based topic labelling using dbpedia. *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM.

Wanqiu Kou, Fang Li, and Timothy Baldwin. 2015. Automatic labelling of topic models using word vectors and letter trigram vectors. *Information Retrieval Technology*. Springer International Publishing, 253-264.

Jey Han Lau, Karl Grieser, David Newman, and Timothy Baldwin. 2011. Automatic labelling of topic models. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2010. Multi-document summarization via budgeted maximization of submodular functions. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Hui Lin and Jeff Bilmes. 2011. A class of submodular functions for document summarization. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics.

Qiaozhu Mei, Chao Liu, Hang Su, and ChengXiang Zhai. 2006. A probabilistic approach to spatiotemporal theme pattern mining on weblogs. In *Proceedings of the 15th international conference on World Wide Web*, pp. 533-542. ACM.

Qiaozhu Mei, Xuehua Shen, and ChengXiang Zhai. 2007. Automatic labeling of multinomial topic models. *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Xian-Ling Mao, Zhao-Yan Ming, Zheng-Jun Zha, Tat-Seng Chua, Hongfei Yan, and Xiaoming Li. 2012. Automatic labeling hierarchical topics. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 2383-2386. ACM.

You Ouyang, Sujian Li, and Wenjie Li. 2007. Developing learning strategies for topic-based summarization. *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*. ACM.

Dragomir R. Radev, Hongyan Jing, Małgorzata Styś, and Daniel Tam. 2004. Centroid-based summarization of multiple documents. *Information Processing & Management* 40, no. 6: 919-938.

Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. 2007. Document summarization using Conditional Random Fields. In *IJCAI*, vol. 7, pp. 2862-2867.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101.suppl 1: 5228-5235.

Xiaojun Wan. 2008. Using only cross-document relationships for both generic and topic-focused multi-document summarizations. *Information Retrieval* 11.1: 25-49.

Xiaojun Wan, Jianwu Yang, and Jianguo Xiao. 2007. Manifold-ranking based topic-focused multi-document summarization. In *IJCAI*, vol. 7, pp. 2903-2908.

Xiaojun Wan and Jianwu Yang. 2008. Multi-document summarization using cluster-based link analysis. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. ACM.

Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM.

Graph-based Dependency Parsing with Bidirectional LSTM

Wenhui Wang **Baobao Chang**
Key Laboratory of Computational Linguistics, Ministry of Education.
School of Electronics Engineering and Computer Science, Peking University,
No.5 Yiheyuan Road, Haidian District, Beijing, 100871, China
Collaborative Innovation Center for Language Ability, Xuzhou, 221009, China.
{wangwenhui,chbb}@pku.edu.cn

Abstract

In this paper, we propose a neural network model for graph-based dependency parsing which utilizes Bidirectional LSTM (BLSTM) to capture richer contextual information instead of using high-order factorization, and enable our model to use much fewer features than previous work. In addition, we propose an effective way to learn sentence segment embedding on sentence-level based on an extra forward LSTM network. Although our model uses only first-order factorization, experiments on English Peen Treebank and Chinese Penn Treebank show that our model could be competitive with previous higher-order graph-based dependency parsing models and state-of-the-art models.

1 Introduction

Dependency parsing is a fundamental task for language processing which has been investigated for decades. It has been applied in a wide range of applications such as information extraction and machine translation. Among a variety of dependency parsing models, graph-based models are attractive for their ability of scoring the parsing decisions on a whole-tree basis. Typical graph-based models factor the dependency tree into subgraphs, including single arcs (McDonald et al., 2005), sibling or grandparent arcs (McDonald and Pereira, 2006; Carreras, 2007) or higher-order substructures (Koo and Collins, 2010; Ma and Zhao, 2012) and then score the whole tree by summing scores of the subgraphs. In these models, subgraphs are usually represented as high-dimensional feature vectors which are then fed into a linear model to learn the feature weights.

However, conventional graph-based models heavily rely on feature engineering and their performance is restricted by the design of features. In addition, standard decoding algorithm (Eisner, 2000) only works for the first-order model which limits the scope of feature selection. To incorporate high-order features, Eisner algorithm must be somehow extended or modified, which is usually done at high cost in terms of efficiency. The fourth-order graph-based model (Ma and Zhao, 2012), which seems the highest-order model so far to our knowledge, requires $O(n^5)$ time and $O(n^4)$ space. Due to the high computational cost, high-order models are normally restricted to producing only unlabeled parses to avoid extra cost introduced by inclusion of arc-labels into the parse trees.

To alleviate the burden of feature engineering, Pei et al. (2015) presented an effective neural network model for graph-based dependency parsing. They only use atomic features such as word unigrams and POS tag unigrams and leave the model to automatically learn the feature combinations. However, their model requires many atomic features and still relies on high-order factorization strategy to further improve the accuracy.

Different from previous work, we propose an LSTM-based dependency parsing model in this paper and aim to use LSTM network to capture richer contextual information to support parsing decisions, instead of adopting a high-order factorization. The main advantages of our model are as follows:

- By introducing Bidirectional LSTM, our model shows strong ability to capture potential long range contextual information and exhibits improved accuracy in recovering long distance dependencies. It is different to previous work in which a similar effect is usually achieved by high-order factorization. More-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2306–2315,
Berlin, Germany, August 7-12, 2016.

over, our model also eliminates the need for setting feature selection windows and reduces the number of features to a minimum level.

- We propose an LSTM-based sentence segment embedding method named LSTM-Minus, in which distributed representation of sentence segment is learned by using subtraction between LSTM hidden vectors. Experiment shows this further enhances our model's ability to access to sentence-level information.

- Last but important, our model is a first-order model using standard Eisner algorithm for decoding, the computational cost remains at the lowest level among graph-based models. Our model does not trade-off efficiency for accuracy.

We evaluate our model on the English Penn Treebank and Chinese Penn Treebank, experiments show that our model achieves competitive parsing accuracy compared with conventional high-order models, however, with a much lower computational cost.

2 Graph-based dependency parsing

In dependency parsing, syntactic relationships are represented as directed arcs between head words and their modifier words. Each word in a sentence modifies exactly one head, but can have any number of modifiers itself. The whole sentence is rooted at a designated special symbol *ROOT*, thus the dependency graph for a sentence is constrained to be a rooted, directed tree.

For a sentence x, graph-based dependency parsing model searches for the highest-scoring tree of x:

$$y^*(x) = \arg\max_{\hat{y} \in Y(x)} Score(x, \hat{y}; \theta) \quad (1)$$

Here $y^*(x)$ is the tree with the highest score, $Y(x)$ is the set of all valid dependency trees for x and $Score(x, \hat{y}; \theta)$ measures how likely the tree $\hat{y}$ is the correct analysis of the sentence x, θ are the model parameters. However, the size of $Y(x)$ grows exponentially with respect to the length of the sentence, directly solving equation (1) is impractical.

The common strategy adopted in the graph-based model is to factor the dependency tree $\hat{y}$ into

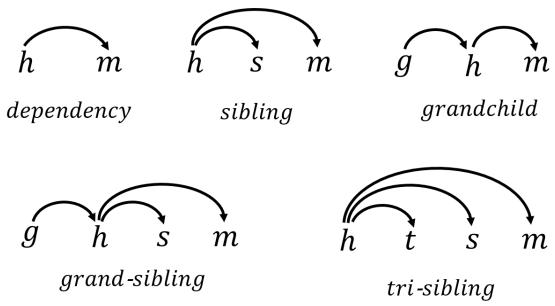


Figure 1: First-order, Second-order and Third-order factorization strategy. Here h stands for head word, m stands for modifier word, s and t stand for the sibling of m. g stands for the grandparent of m.

a set of subgraph c which can be scored in isolation, and score the whole tree $\hat{y}$ by summing score of each subgraph:

$$Score(x, \hat{y}; \theta) = \sum_{c \in \hat{y}} ScoreC(x, c; \theta) \quad (2)$$

Figure 1 shows several factorization strategies. The order of the factorization is defined according to the number of dependencies that subgraph contains. The simplest first-order factorization (McDonald et al., 2005) decomposes a dependency tree into single dependency arcs. Based on the first-order factorization, second-order factorization (McDonald and Pereira, 2006; Carreras, 2007) brings sibling and grandparent information into their model. Third-order factorization (Koo and Collins, 2010) further incorporates richer contextual information by utilizing grand-sibling and tri-sibling parts.

Conventional graph-based models (McDonald et al., 2005; McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010; Ma and Zhao, 2012) score subgraph by a linear model, which heavily depends on feature engineering. The neural network model proposed by Pei et al. (2015) alleviates the dependence on feature engineering to a large extent, but not completely. We follow Pei et al. (2015) to score dependency arcs using neural network model. However, different from their work, we introduce a Bidirectional LSTM to capture long range contextual information and an extra forward LSTM to better represent segments of the sentence separated by the head and modifier. These make our model more accurate in recovering long-distance dependencies and further decrease the number of atomic features.

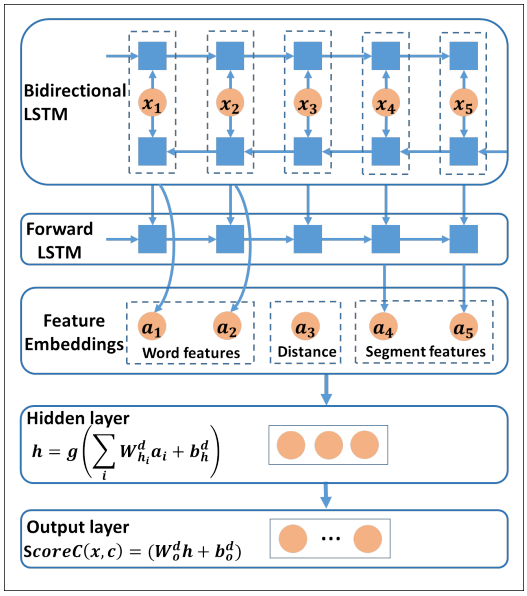


Figure 2: Architecture of the Neural Network. x_1 to x_5 stand for the input token of Bidirectional LSTM. a_1 to a_5 stand for the feature embeddings used in our model.

3 Neural Network Model

In this section, we describe the architecture of our neural network model in detail, which is summarized in Figure 2.

3.1 Input layer

In our neural network model, the words, POS tags are mapped into distributed embeddings. We represent each input token x_i which is the input of Bidirectional LSTM by concatenating POS tag embedding $e_{p_i} \in \mathbb{R}^{d_e}$ and word embedding $e_{w_i} \in \mathbb{R}^{d_e}$, d_e is the the dimensionality of embedding, then a linear transformation w_e is performed and passed though an element-wise activation function g:

$$x_i = g(w_e[e_{w_i}; e_{p_i}] + b_e) \tag{3}$$

where $x_i \in \mathbb{R}^{d_e}$, $w_e \in \mathbb{R}^{d_e \times 2d_e}$ is weight matrix, $b_e \in \mathbb{R}^{d_e}$ is bias term. the dimensionality of input token x_i is equal to the dimensionality of word and POS tag embeddings in our experiment, ReLU is used as our activation function g.

3.2 Bidirectional LSTM

Given an input sequence $x = (x_1, \ldots, x_n)$, where n stands for the number of words in a sentence, a standard LSTM recurrent network computes the hidden vector sequence $h = (h_1, \ldots, h_n)$ in one direction.

Bidirectional LSTM processes the data in both directions with two separate hidden layers, which are then fed to the same output layer. It computes the forward hidden sequence $\overrightarrow{h}$, the backward hidden sequence $\overleftarrow{h}$ and the output sequence v by iterating the forward layer from $t = 1$ to n, the backward layer from $t = n$ to 1 and then updating the output layer:

$$v_t = \overrightarrow{h}_t + \overleftarrow{h}_t \tag{4}$$

where $v_t \in \mathbb{R}^{d_l}$ is the output vector of Bidirectional LSTM for input x_t, $\overrightarrow{h}_t \in \mathbb{R}^{d_l}$, $\overleftarrow{h}_t \in \mathbb{R}^{d_l}$, d_l is the dimensionality of LSTM hidden vector. We simply add the forward hidden vector $\overrightarrow{h}_t$ and the backward hidden vector $\overleftarrow{h}_t$ together, which gets similar experiment result as concatenating them together with a faster speed.

The output vectors of Bidirectional LSTM are used as word feature embeddings. In addition, they are also fed into a forward LSTM network to learn segment embedding.

3.3 Segment Embedding

Contextual information of word pairs[1] has been widely utilized in previous work (McDonald et al., 2005; McDonald and Pereira, 2006; Pei et al., 2015). For a dependency pair (h, m), previous work divides a sentence into three parts (*prefix*, *infix* and *suffix*) by head word h and modifier word m. These parts which we call *segments* in our work make up the context of the dependency pair (h, m).

Due to the problem of data sparseness, conventional graph-based models can only capture contextual information of word pairs by using bigrams or tri-grams features. Unlike conventional models, Pei et al. (2015) use distributed representations obtained by averaging word embeddings in segments to represent contextual information of the word pair, which could capture richer syntactic and semantic information. However, their method is restricted to segment-level since their segment embedding only consider the word information within the segment. Besides, averaging operation simply treats all the words in segment equally. However, some words might carry more

[1] A word pair is limited to the dependency pair (h, m) in our work since we use only first-order factorization. In previous work, word pair could be any pair with particular relation (e.g., sibling pair (s, m) in Figure 1).

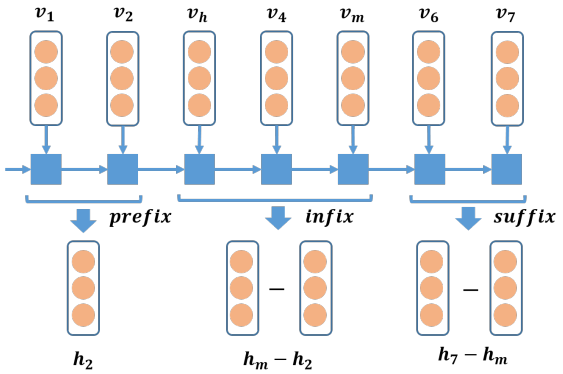


Figure 3: Illustration for learning segment embeddings based on an extra forward LSTM network, v_h, v_m and v_1 to v_7 indicate the output vectors of Bidirectional LSTM for head word h, modifier word m and other words in sentence, h_h, h_m and h_1 to h_7 indicate the hidden vectors of the forward LSTM corresponding to v_h, v_m and v_1 to v_7.

salient syntactic or semantic information and they are expected to be given more attention.

A useful property of forward LSTM is that it could keep previous useful information in their memory cell by exploiting input, output and forget gates to decide how to utilize and update the memory of previous information. Given an input sequence $v = (v_1, \ldots, v_n)$, previous work (Sutskever et al., 2014; Vinyals et al., 2014) often uses the last hidden vector h_n of the forward LSTM to represent the whole sequence. Each hidden vector h_t $(1 \leq t \leq n)$ can capture useful information before and including v_t.

Inspired by this, we propose a method named **LSTM-Minus** to learn segment embedding. We utilize subtraction between LSTM hidden vectors to represent segment's information. As illustrated in Figure 3, the segment *infix* can be described as $h_m - h_2$, h_m and h_2 are hidden vector of the forward LSTM network. The segment embedding of *suffix* can also be obtained by subtraction between the last LSTM hidden vector of the sequence (h_7) and the last LSTM hidden vector in *infix* (h_m). For *prefix*, we directly use the last LSTM hidden vector in *prefix* to represent it, which equals to subtract a zero embedding. When no *prefix* or *suffix* exists, the corresponding embedding is set to zero.

Specifically, we place an extra forward LSTM layer on top of the Bidirectional LSTM layer and learn segment embeddings using LSTM-Minus based on this forward LSTM. LSTM-minus enables our model to learn segment embeddings from information both outside and inside the segments and thus enhances our model's ability to access to sentence-level information.

3.4 Hidden layer and output layer

As illustrated in Figure 2, we map all the feature embeddings to a hidden layer. Following Pei et al. (2015), we use direction-specific transformation to model edge direction and tanh-cube as our activation function:

$$h = g\Big(\sum_i W^d_{h_i} a_i + b^d_h\Big) \tag{5}$$

where $a_i \in \mathbb{R}^{d_{a_i}}$ is the feature embedding, d_{a_i} indicates the dimensionality of feature embedding a_i, $W^d_{h_i} \in \mathbb{R}^{d_h \times d_{a_i}}$ is weight matrices which corresponding to a_i, d_h indicates the dimensionality of hidden layer vector, $b^d_h \in \mathbb{R}^{d_h}$ is bias term. $W^d_{h_i}$ and b^d_h are bound with index $d \in \{0, 1\}$ which indicates the direction between head and modifier.

A output layer is finally added on the top of the hidden layer for scoring dependency arcs:

$$ScoreC(x, c) = W^d_o h + b^d_o \tag{6}$$

Where $W^d_o \in \mathbb{R}^{L \times d_h}$ is weight matrices, $b^d_o \in \mathbb{R}^L$ is bias term, $ScoreC(x, c) \in \mathbb{R}^L$ is the output vector, L is the number of dependency types. Each dimension of the output vector is the score for each kind of dependency type of head-modifier pair.

3.5 Features in our model

Previous neural network models (Pei et al., 2015; Pei et al., 2014; Zheng et al., 2013) normally set context window around a word and extract atomic features within the window to represent the contextual information. However, context window limits their ability in detecting long-distance information. Simply increasing the context window size to get more contextual information puts their model in the risk of overfitting and heavily slows down the speed.

Unlike previous work, we apply Bidirectional LSTM to capture long range contextual information and eliminate the need for context windows, avoiding the limit of the window-based feature selection approach. Compared with Pei et al. (2015), the cancellation of the context window allows our model to use much fewer features. Moreover, by combining a word's atomic features (word form and POS tag) together, our model further decreases the number of features.

Pei et al. (2015)	$h_{-2}.w$, $h_{-1}.w$, $h.w$, $h_1.w$, $h_2.w$ $h_{-2}.p$, $h_{-1}.p$, $h.p$, $h_1.p$, $h_2.p$ $m_{-2}.w$, $m_{-1}.w$, $m.w$, $m_1.w$, $m_2.w$ $m_{-2}.p$, $m_{-1}.p$, $m.p$, $m_1.p$, $m_2.p$ *dis(h, m)*
Our basic model	v_h, v_m *dis(h, m)*

Table 1: Atomic features in our basic model and Pei's 1st-order atomic model. w is short for word and p for POS tag. h indicates head and m indicates modifier. The subscript represents the relative position to the center word. $dis(h, m)$ is the distance between head and modifier. v_h and v_m indicate the outputs of Bidirectional LSTM for head word and modifier word.

Table 1 lists the atomic features used in 1st-order atomic model of Pei et al. (2015) and atomic features used in our basic model. Our basic model only uses the outputs of Bidirectional LSTM for head word and modifier word, and the distance between them as features. Distance features are encoded as randomly initialized embeddings. As we can see, our basic model reduces the number of atomic features to a minimum level, making our model run with a faster speed. Based on our basic model, we incorporate additional segment information (*prefix*, *infix* and *suffix*), which further improves the effect of our model.

4 Neural Training

In this section, we provide details about training the neural network.

4.1 Max-Margin Training

We use the Max-Margin criterion to train our model. Given a training instance $(x^{(i)}, y^{(i)})$, we use $Y(x^{(i)})$ to denote the set of all possible dependency trees and $y^{(i)}$ is the correct dependency tree for sentence $x^{(i)}$. The goal of Max Margin training is to find parameters θ such that the difference in score of the correct tree $y^{(i)}$ from an incorrect tree $\hat{y} \in Y(x^{(i)})$ is at least $\triangle(y^{(i)}, \hat{y})$.

$$Score(x^{(i)},y^{(i)};\theta) \geq Score(x^{(i)},\hat{y};\theta)+\triangle(y^{(i)},\hat{y})$$

The structured margin loss $\triangle(y^{(i)}, \hat{y})$ is defined as:

$$\triangle(y^{(i)}, \hat{y}) = \sum_{j}^{n} \kappa \mathbf{1}\{h(y^{(i)}, x_j^{(i)}) \neq h(\hat{y}, x_j^{(i)})\}$$

where n is the length of sentence x, $h(y^{(i)}, x_j^{(i)})$ is the head (with type) for the j-th word of $x^{(i)}$ in tree $y^{(i)}$ and κ is a discount parameter. The loss is proportional to the number of word with an incorrect head and edge type in the proposed tree.

Given a training set with size m, The regularized objective function is the loss function $J(\theta)$ including a l_2-norm term:

$$\begin{aligned} J(\theta) &= \frac{1}{m}\sum_{i=1}^{m} l_i(\theta) + \frac{\lambda}{2}||\theta||^2 \\ l_i(\theta) &= \max_{\hat{y}\in Y(x^{(i)})} (Score(x^{(i)},\hat{y};\theta)+\triangle(y^{(i)},\hat{y})) \\ &\quad -Score(x^{(i)},y^{(i)};\theta) \end{aligned} \tag{7}$$

By minimizing this objective, the score of the correct tree is increased and score of the highest scoring incorrect tree is decreased.

4.2 Optimization Algorithm

Parameter optimization is performed with the diagonal variant of AdaGrad (Duchi et al., 2011) with minibatchs (batch size = 20) . The parameter update for the i-th parameter $\theta_{t,i}$ at time step t is as follows:

$$\theta_{t,i} = \theta_{t-1,i} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} g_{\tau,i}^2}} g_{t,i} \tag{8}$$

where α is the initial learning rate ($\alpha = 0.2$ in our experiment) and $g_\tau \in \mathbb{R}^{|\theta_i|}$ is the subgradient at time step τ for parameter θ_i.

To mitigate overfitting, dropout (Hinton et al., 2012) is used to regularize our model. we apply dropout on the hidden layer with 0.2 rate.

4.3 Model Initialization&Hyperparameters

The following hyper-parameters are used in all experiments: word embedding size = 100, POS tag embedding size = 100, hidden layer size = 200, LSTM hidden vector size = 100, Bidirectional LSTM layers = 2, regularization parameter $\lambda = 10^{-4}$.

We initialized the parameters using pretrained word embeddings. Following Dyer et al. (2015), we use a variant of the skip n-gram model introduced by Ling et al. (2015) on Gigaword corpus (Graff et al., 2003). We also experimented with randomly initialized embeddings, where embeddings are uniformly sampled from range $[-0.3, 0.3]$. All other parameters are uniformly sampled from range $[-0.05, 0.05]$.

	Models	UAS	LAS	Speed(sent/s)
First-order	MSTParser	91.60	90.39	20
	1st-order atomic (Pei et al., 2015)	92.14	90.92	55
	1st-order phrase (Pei et al., 2015)	92.59	91.37	26
	Our basic model	93.09	92.03	**61**
	Our basic model + segment	**93.51**	**92.45**	26
Second-order	MSTParser	92.30	91.06	14
	2nd-order phrase (Pei et al., 2015)	93.29	92.13	10
Third-order	(Koo and Collins, 2010)	93.04	N/A	N/A
Fourth-order	(Ma and Zhao, 2012)	93.4	N/A	N/A
Unlimited-order	(Zhang and McDonald, 2012)	93.06	91.86	N/A
	(Zhang et al., 2013)	93.50	92.41	N/A
	(Zhang and McDonald, 2014)	**93.57**	**92.48**	N/A

Table 2: Comparison with previous graph-based models on Penn-YM.

5 Experiments

In this section, we present our experimental setup and the main result of our work.

5.1 Experiments Setup

We conduct our experiments on the English Penn Treebank (PTB) and the Chinese Penn Treebank (CTB) datasets.

For English, we follow the standard splits of PTB3. Using section 2-21 for training, section 22 as development set and 23 as test set. We conduct experiments on two different constituency-to-dependency-converted Penn Treebank data sets. The first one, **Penn-YM**, was created by the Penn2Malt tool[2] based on Yamada and Matsumoto (2003) head rules. The second one, **Penn-SD**, use Stanford Basic Dependencies (Marneffe et al., 2006) and was converted by version 3.3.0[3] of the Stanford parser. The Stanford POS Tagger (Toutanova et al., 2003) with ten-way jackknifing of the training data is used for assigning POS tags (accuracy $\approx$ 97.2%).

For Chinese, we adopt the same split of CTB5 as described in (Zhang and Clark, 2008). Following (Zhang and Clark, 2008; Dyer et al., 2015; Chen and Manning, 2014), we use gold segmentation and POS tags for the input.

5.2 Experiments Results

We first make comparisons with previous graph-based models of different orders as shown in Table 2. We use MSTParser [4] for conventional first-order model (McDonald et al., 2005) and second-order model (McDonald and Pereira, 2006). We also include the results of conventional high-order models (Koo and Collins, 2010; Ma and Zhao, 2012; Zhang and McDonald, 2012; Zhang et al., 2013; Zhang and McDonald, 2014) and the neural network model of Pei et al. (2015). Different from typical high-order models (Koo and Collins, 2010; Ma and Zhao, 2012), which need to extend their decoding algorithm to score new types of higher-order dependencies. Zhang and McDonald (2012) generalized the Eisner algorithm to handle arbitrary features over higher-order dependencies and controlled complexity via approximate decoding with cube pruning. They further improve their work by using perceptron update strategies for inexact hypergraph search (Zhang et al., 2013) and forcing inference to maintain both label and structural ambiguity through a secondary beam (Zhang and McDonald, 2014).

Following previous work, UAS (unlabeled attachment scores) and LAS (labeled attachment scores) are calculated by excluding punctuation[5]. The parsing speeds are measured on a workstation with Intel Xeon 3.4GHz CPU and 32GB RAM which is same to Pei et al. (2015). We measure the parsing speeds of Pei et al. (2015) according to their codes[6] and parameters.

On accuracy, as shown in table 2, our

[2] http://stp.lingfil.uu.se/nivre/research/Penn2Malt.html

[3] http://nlp.stanford.edu/software/lex-parser.shtml

[4] http://sourceforge.net/projects/mstparser

[5] Following previous work, a token is a punctuation if its POS tag is {“ ” : , .}

[6] https://github.com/Williammed/DeepParser

Method	Penn-YM		Penn-SD		CTB5	
	UAS	LAS	UAS	LAS	UAS	LAS
(Zhang and Nivre, 2011)	92.9	91.8	-	-	86.0	84.4
(Bernd Bohnet, 2012)	93.39	92.38	-	-	87.5	85.9
(Zhang and McDonald, 2014)	**93.57**	**92.48**	93.01	90.64	**87.96**	**86.34**
(Dyer et al., 2015)	-	-	93.1	90.9	87.2	85.7
(Weiss et al., 2015)	-	-	93.99	**92.05**	-	-
Our basic model + segment	93.51	92.45	**94.08**	91.82	87.55	86.23

Table 3: Comparison with previous state-of-the-art models on Penn-YM, Penn-SD and CTB5.

basic model outperforms previous first-order graph-based models by a substantial margin, even outperforms Zhang and McDonald (2012)'s unlimited-order model. Moreover, incorporating segment information further improves our model's accuracy, which shows that segment embeddings do capture richer contextual information. By using segment embeddings, our improved model could be comparable to high-order graph-based models[7].

With regard to parsing speed, our model also shows advantage of efficiency. Our model uses only first-order factorization and requires $O(n^3)$ time to decode. Third-order model requires $O(n^4)$ time and fourth-order model requires $O(n^5)$ time. By using approximate decoding, the unlimited-order model of Zhang and McDonald (2012) requires $O(k \cdot log(k) \cdot n^3)$ time, where k is the beam size. The computational cost of our model is the lowest among graph-based models. Moreover, although using LSTM requires much computational cost. However, compared with Pei's 1st-order model, our model decreases the number of atomic features from 21 to 3, this allows our model to require a much smaller matrix computation in the scoring model, which cancels out the extra computation cost introduced by the LSTM computation. Our basic model is the fastest among first-order and second-order models. Incorporating segment information slows down the parsing speed while it is still slightly faster than conventional first-order model. To compare with conventional high-order models on practical parsing speed, we can make an indirect comparison according to Zhang and McDonald (2012). Conventional first-order model is about 10 times faster than Zhang and McDonald (2012)'s unlimited-order model and about 40 times faster than conventional third-order model, while our model is faster than conventional first-order model. Our model should be much faster than conventional high-order models.

[7]Note that our model can't be strictly comparable with third-order model (Koo and Collins, 2010) and fourth-order model (Ma and Zhao, 2012) since they are unlabeled model. However, our model is comparable with all the three unlimited-order models presented in (Zhang and McDonald, 2012), (Zhang et al., 2013) and (Zhang and McDonald, 2014), since they all are labeled models as ours.

Method	Peen-YM	Peen-SD	CTB5
Average	93.23	93.83	87.24
LSTM-Minus	**93.51**	**94.08**	**87.55**

Table 4: Model performance of different way to learn segment embeddings.

We further compare our model with previous state-of-the-art systems for English and Chinese. Table 3 lists the performances of our model as well as previous state-of-the-art systems on on Penn-YM, Penn-SD and CTB5. We compare to conventional state-of-the-art graph-based model (Zhang and McDonald, 2014), conventional state-of-the-art transition-based model using beam search (Zhang and Nivre, 2011), transition-based model combining graph-based approach (Bernd Bohnet, 2012) , transition-based neural network model using stack LSTM (Dyer et al., 2015) and transition-based neural network model using beam search (Weiss et al., 2015). Overall, our model achieves competitive accuracy on all three datasets. Although our model is slightly lower in accuarcy than unlimited-order double beam model (Zhang and McDonald, 2014) on Penn-YM and CTB5, our model outperforms their model on Penn-SD. It seems that our model performs better on data sets with larger label sets, given the number of labels used in Penn-SD data set is almost four times more than Penn-YM and CTB5 data sets.

To show the effectiveness of our segment embedding method LSTM-Minus, we compare with averaging method proposed by Pei et al. (2015). We get segment embeddings by averaging the output vectors of Bidirectional LSTM in segments.

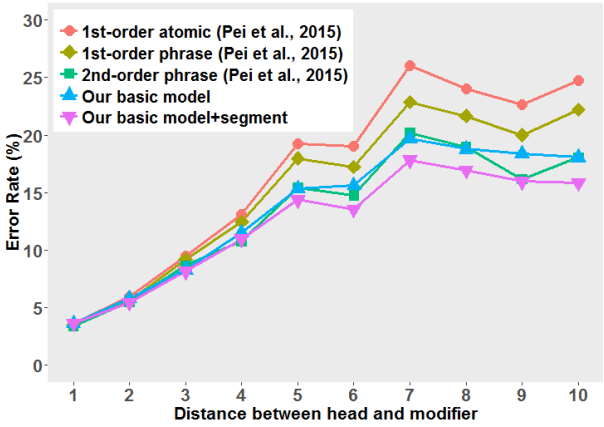


Figure 4: Error rates of different distance between head and modifier on Peen-YM.

To make comparison as fair as possible, we let two models have almost the same number parameters. Table 4 lists the UAS of two methods on test set. As we can see, LSTM-Minus shows better performance because our method further incorporates more sentence-level information into our model.

5.3 Impact of Network Structure

In this part, we investigate the impact of the components of our approach.

LSTM Recurrent Network

To evaluate the impact of LSTM, we make error analysis on Penn-YM. We compare our model with Pei et al. (2015) on error rates of different distance between head and modifier.

As we can see, the five models do not show much difference for short dependencies whose distance less than three. For long dependencies, both our two models show better performance compared with the 1st-order model of Pei et al. (2015), which proves that LSTM can effectively capture long-distance dependencies. Moreover, our models and Pei's 2nd-order phrase model both improve accuracy on long dependencies compared with Pei's 1st-order model, which is in line with our expectations. Using LSTM shows the same effect as high-order factorization strategy. Compared with 2nd-order phrase model of Pei et al. (2015), our basic model occasionally performs worse in recovering long distant dependencies. However, this should not be a surprise since higher order models are also motivated to recover long-distance dependencies. Nevertheless, with the introduction of LSTM-minus segment embeddings, our model consistently outperforms the 2nd-order phrase model of Pei et al. (2015) in accuracies of all long dependencies. We carried out significance test on the difference between our and Pei's models. Our basic model performs significantly better than all 1st-order models of Pei et al. (2015) (t-test with $p<0.001$) and our basic+segment model (still a 1st-order model) performs significantly better than their 2nd-order phrase model (t-test with $p<0.001$) in recovering long-distance dependencies.

Initialization of pre-trained word embeddings

We further analyze the influence of using pre-trained word embeddings for initialization. without using pretrained word embeddings, our improved model achieves 92.94% UAS / 91.83% LAS on Penn-YM, 93.46% UAS / 91.19% LAS on Penn-SD and 86.5% UAS / 85.0% LAS on CTB5. Using pre-trained word embeddings can obtain around 0.5%$\sim$1.0% improvement.

6 Related work

Dependency parsing has gained widespread interest in the computational linguistics community. There are a lot of approaches to solve it. Among them, we will mainly focus on graph-based dependency parsing model here. Dependency tree factorization and decoding algorithm are necessary for graph-based models. McDonald et al. (2005) proposed the first-order model which decomposes a dependency tree into its individual edges and use a effective dynamic programming algorithm (Eisner, 2000) to decode. Based on first-order model, higher-order models(McDonald and Pereira, 2006; Carreras, 2007; Koo and Collins, 2010; Ma and Zhao, 2012) factor a dependency tree into a set of high-order dependencies which bring interactions between head, modifier, siblings and (or) grandparent into their model. However, for above models, scoring new types of higher-order dependencies requires extensions of the underlying decoding algorithm, which also requires higher computational cost. Unlike above models, unlimited-order models (Zhang and McDonald, 2012; Zhang et al., 2013; Zhang and McDonald, 2014) could handle arbitrary features over higher-order dependencies by generalizing the Eisner algorithm.

In contrast to conventional methods, neural network model shows their ability to reduce the effort in feature engineering. Pei et al. (2015) proposed a model to automatically learn high-order feature

combinations via a novel activation function, allowing their model to use a set of atomic features instead of millions of hand-crafted features.

Different from previous work, which is sensitive to local state and accesses to larger context by higher-order factorization. Our model makes parsing decisions on a global perspective with first-order factorization, avoiding the expensive computational cost introduced by high-order factorization.

LSTM network is heavily utilized in our model. LSTM network has already been explored in transition-based dependency parsing. Dyer et al. (2015) presented stack LSTMs with push and pop operations and used them to implement a state-of-the-art transition-based dependency parser. Ballesteros et al. (2015) replaced lookup-based word representations with character-based representations obtained by Bidirectional LSTM in the continuous-state parser of Dyer et al. (2015), which was proved experimentally to be useful for morphologically rich languages.

7 Conclusion

In this paper, we propose an LSTM-based neural network model for graph-based dependency parsing. Utilizing Bidirectional LSTM and segment embeddings learned by LSTM-Minus allows our model access to sentence-level information, making our model more accurate in recovering long-distance dependencies with only first-order factorization. Experiments on PTB and CTB show that our model could be competitive with conventional high-order models with a faster speed.

Acknowledgments

This work is supported by National Key Basic Research Program of China under Grant No.2014CB340504 and National Natural Science Foundation of China under Grant No.61273318. The Corresponding author of this paper is Baobao Chang.

References

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 349–359.

Jonas Kuhn Bernd Bohnet. 2012. The best of both worlds: a graph-based completion model for transition-based parsers. *Conference of the European Chapter of the Association for Computational Linguistics*.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP-CoNLL*, pages 957–961.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 334–343.

Jason Eisner. 2000. *Bilexical Grammars and their Cubic-Time Parsing Algorithms*. Springer Netherlands.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*.

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

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1–11. Association for Computational Linguistics.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Xuezhe Ma and Hai Zhao. 2012. Fourth-order dependency parsing. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Posters, 8-15 December 2012, Mumbai, India*, pages 785–796.

Marie Catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. *Lrec*, pages 449–454.

Ryan T McDonald and Fernando CN Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL*. Citeseer.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 91–98. Association for Computational Linguistics.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2014. Max-margin tensor neural network for chinese word segmentation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 293–303.

Wenzhe Pei, Tao Ge, and Baobao Chang. 2015. An effective neural network model for graph-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 313–322.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing System*, pages 3104–3112.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics.

Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2014. Grammar as a foreign language. *CoRR*, abs/1412.7449.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of IWPT*, volume 3, pages 195–206.

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

Hao Zhang and Ryan T. McDonald. 2012. Generalized higher-order dependency parsing with cube pruning. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 320–331.

Hao Zhang and Ryan T. McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 656–661.

Yue Zhang and Joakim Nivre. 2011. Transition-based dependency parsing with rich non-local features. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 188–193.

Hao Zhang, Liang Huang Kai Zhao, and Ryan Mcdonald. 2013. Online learning for inexact hypergraph search. *Proceedings of Emnlp*.

Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. 2013. Deep learning for Chinese word segmentation and POS tagging. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 647–657, October.

TransG : A Generative Model for Knowledge Graph Embedding

Han Xiao, Minlie Huang*, Xiaoyan Zhu
State Key Lab. of Intelligent Technology and Systems
National Lab. for Information Science and Technology
Dept. of Computer Science and Technology
Tsinghua University, Beijing 100084, PR China
{aihuang, zxy-dcs}@tsinghua.edu.cn

Abstract

Recently, knowledge graph embedding, which projects symbolic entities and relations into continuous vector space, has become a new, hot topic in artificial intelligence. This paper proposes a novel generative model (**TransG**) to address the issue of **multiple relation semantics** that a relation may have multiple meanings revealed by the entity pairs associated with the corresponding triples. The new model can discover latent semantics for a relation and leverage a mixture of relation-specific component vectors to embed a fact triple. To the best of our knowledge, this is the first generative model for knowledge graph embedding, and at the first time, the issue of multiple relation semantics is formally discussed. Extensive experiments show that the proposed model achieves substantial improvements against the state-of-the-art baselines.

1 Introduction

Abstract or real-world knowledge is always a major topic in Artificial Intelligence. Knowledge bases such as Wordnet (Miller, 1995) and Freebase (Bollacker et al., 2008) have been shown very useful to AI tasks including question answering, knowledge inference, and so on. However, traditional knowledge bases are symbolic and logic, thus numerical machine learning methods cannot be leveraged to support the computation over the knowledge bases. To this end, knowledge graph embedding has been proposed to project entities and relations into continuous vector spaces. Among various embedding models, there is a line of translation-based models such as TransE (Bordes et al., 2013), TransH (Wang et al., 2014), TransR (Lin et al., 2015b), and other related models (He et al., 2015) (Lin et al., 2015a).

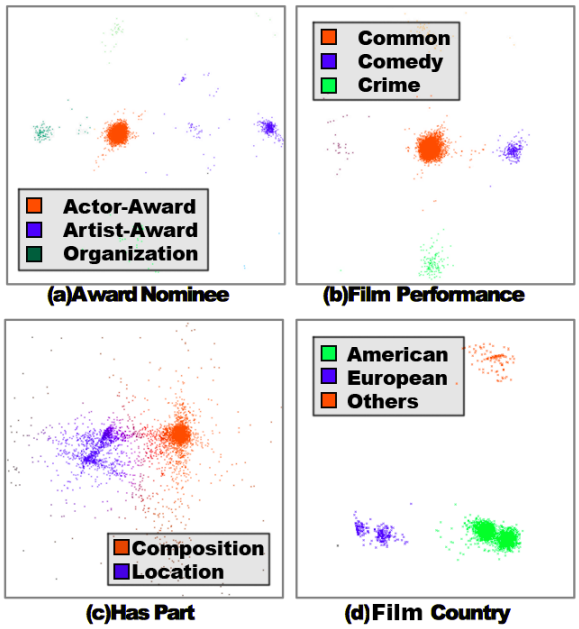


Figure 1: Visualization of TransE embedding vectors with PCA dimension reduction. Four relations (a $\sim$ d) are chosen from Freebase and Wordnet. A dot denotes a triple and its position is decided by the difference vector between tail and head entity ($\mathbf{t} - \mathbf{h}$). Since TransE adopts the principle of $\mathbf{t} - \mathbf{h} \approx \mathbf{r}$, there is supposed to be only one cluster whose centre is the relation vector $\mathbf{r}$. However, results show that there exist multiple clusters, which justifies our multiple relation semantics assumption.

A fact of knowledge base can usually be represented by a triple (h, r, t) where h, r, t indicate a head entity, a relation, and a tail entity, respectively. All translation-based models almost follow the same principle $\mathbf{h_r} + \mathbf{r} \approx \mathbf{t_r}$ where $\mathbf{h_r}, \mathbf{r}, \mathbf{t_r}$ in-

* Correspondence author

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2316–2325,
Berlin, Germany, August 7-12, 2016.

dicate the embedding vectors of triple (h, r, t), with the head and tail entity vector projected with respect to the relation space.

In spite of the success of these models, none of the previous models has formally discussed the issue of ***multiple relation semantics*** that a relation may have multiple meanings revealed by the entity pairs associated with the corresponding triples. As can be seen from Fig. 1, visualization results on embedding vectors obtained from TransE (Bordes et al., 2013) show that, there are different clusters for a specific relation, and different clusters indicate different latent semantics. For example, the relation HasPart has at least two latent semantics: composition-related as (Table, HasPart, Leg) and location-related as (Atlantics, HasPart, NewYorkBay). As one more example, in Freebase, (Jon Snow, birth place, Winter Fall) and (George R. R. Martin, birth place, U.S.) are mapped to schema /fictional_universe/fictional_character/place_of_birth and /people/person/place_of_birth respectively, indicating that *birth place* has different meanings. This phenomenon is quite common in knowledge bases for two reasons: artificial simplification and nature of knowledge. On one hand, knowledge base curators could not involve too many similar relations, so abstracting multiple similar relations into one specific relation is a common trick. On the other hand, both language and knowledge representations often involve ambiguous information. The ambiguity of knowledge means a semantic mixture. For example, when we mention "Expert", we may refer to scientist, businessman or writer, so the concept "Expert" may be ambiguous in a specific situation, or generally a semantic mixture of these cases.

However, since previous translation-based models adopt $\mathbf{h_r} + \mathbf{r} \approx \mathbf{t_r}$, they assign only one translation vector for one relation, and these models are not able to deal with the issue of multiple relation semantics. To illustrate more clearly, as showed in Fig.2, there is only one unique representation for relation $\mathrm{HasPart}$ in traditional models, thus the models made more errors when embedding the triples of the relation. Instead, in our proposed model, we leverage a Bayesian non-parametric infinite mixture model to handle multiple relation semantics by generating multiple translation components for a relation. Thus, different semantics are characterized by different components in our embedding model. For example, we can distinguish the two clusters $\mathrm{HasPart}.1$ or $\mathrm{HasPart}.2$, where the relation semantics are automatically clustered to represent the meaning of associated entity pairs.

To summarize, our contributions are as follows:

- We propose a new issue in knowledge graph embedding, ***multiple relation semantics*** that a relation in knowledge graph may have different meanings revealed by the associated entity pairs, which has never been studied previously.
- To address the above issue, we propose a novel Bayesian non-parametric infinite mixture embedding model, TransG. The model can automatically discover semantic clusters of a relation, and leverage a mixture of multiple relation components for translating an entity pair. Moreover, we present new insights from the generative perspective.
- Extensive experiments show that our proposed model obtains substantial improvements against the state-of-the-art baselines.

2 Related Work

Translation-Based Embedding. Existing translation-based embedding methods share the same translation principle $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ and the score function is designed as:

$$f_r(h, t) = ||\mathbf{h_r} + \mathbf{r} - \mathbf{t_r}||_2^2$$

where $\mathbf{h_r}, \mathbf{t_r}$ are entity embedding vectors projected in the relation-specific space. **TransE** (Bordes et al., 2013), lays the entities in the original entity space: $\mathbf{h_r} = \mathbf{h}, \mathbf{t_r} = \mathbf{t}$. **TransH** (Wang et al., 2014) projects entities into a hyperplane for addressing the issue of complex relation embedding: $\mathbf{h_r} = \mathbf{h} - \mathbf{w_r}^\top \mathbf{h}\mathbf{w_r}, \mathbf{t_r} = \mathbf{t} - \mathbf{w_r}^\top \mathbf{t}\mathbf{w_r}$. To address the same issue, **TransR** (Lin et al., 2015b), transforms the entity embeddings by the same relation-specific matrix: $\mathbf{h_r} = \mathbf{M_r}\mathbf{h}, \mathbf{t_r} = \mathbf{M_r}\mathbf{t}$. TransR also proposes an ad-hoc clustering-based method, **CTransR**, where the entity pairs for a relation are clustered into different groups, and the pairs in the same group share the same relation vector. In comparison, our model is more elegant to address such an issue theoretically, and does not require a pre-process of clustering. Furthermore, our model has much better performance than CTransR, as expected. **TransM** (Fan et al.,

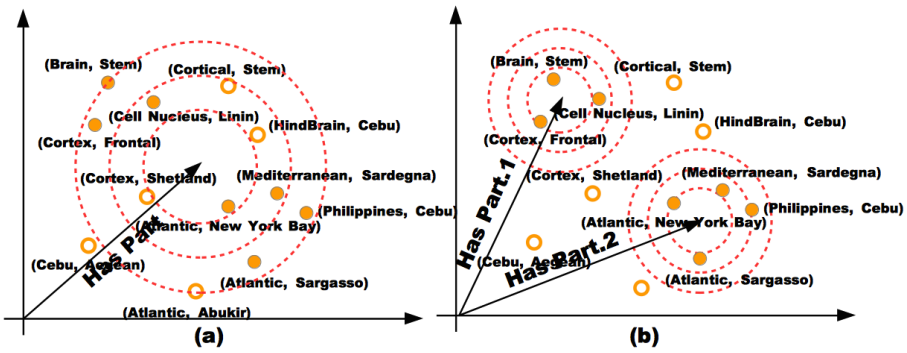


Figure 2: Visualization of multiple relation semantics. The data are selected from Wordnet. The dots are correct triples that belong to HasPart relation, while the circles are incorrect ones. The point coordinate is the difference vector between tail and head entity, which should be near to the centre. (a) The correct triples are hard to be distinguished from the incorrect ones. (b) By applying multiple semantic components, our proposed model could discriminate the correct triples from the wrong ones.

2014) leverages the structure of the knowledge graph via pre-calculating the distinct weight for each training triple to enhance embedding. **KG2E** (He et al., 2015) is a probabilistic embedding method for modeling the uncertainty in knowledge graph.

There are many works to improve translation-based methods by considering other information. For instance, (Guo et al., 2015) aims at discovering the geometric structure of the embedding space to make it semantically smooth. (Wang et al., 2014) focuses on bridging the gap between knowledge and texts, with a loss function for jointly modeling knowledge graph and text resources. (Wang et al., 2015) incorporates the rules that are related with relation types such as 1-N and N-1. **PTransE** (Lin et al., 2015a) takes into account path information in knowledge graph.

Since the previous models are point-wise modeling methods, **ManifoldE** (Xiao et al., 2016) proposes a novel manifold-based approach for knowledge graph embedding. In aid of kernel tricks, manifold-based methods can improve embedding performance substantially.

Structured & Unstructured Embedding. The structured embedding model (Bordes et al., 2011) transforms the entity space with the head-specific and tail-specific matrices. The score function is defined as $f_r(h,t) = ||\mathbf{M_{h,r}h} - \mathbf{M_{t,r}t}||$. According to (Socher et al., 2013), this model cannot capture the relationship between entities. Semantic Matching Energy (SME) (Bordes et al., 2012) (Bordes et al., 2014) can handle the correlations between entities and relations by matrix product and Hadamard product. The unstructured model (Bordes et al., 2012) may be a simplified version of TransE without considering any relation-related information. The score function is directly defined as $f_r(h,t) = ||\mathbf{h} - \mathbf{t}||_2^2$.

Neural Network based Embedding. Single Layer Model (SLM) (Socher et al., 2013) applies neural network to knowledge graph embedding. The score function is defined as $f_r(h,t) = \mathbf{u_r}^\top g(\mathbf{M_{r,1}h} + \mathbf{M_{r,2}t})$ where $\mathbf{M_{r,1}}, \mathbf{M_{r,2}}$ are relation-specific weight matrices. Neural Tensor Network (NTN) (Socher et al., 2013) defines a very expressive score function by applying tensor: $f_r(h,t) = \mathbf{u_r}^\top g(\mathbf{h}^\top \mathbf{W}_{\cdot\cdot\mathbf{r}}\mathbf{t} + \mathbf{M_{r,1}h} + \mathbf{M_{r,2}t} + \mathbf{b_r})$, where $\mathbf{u_r}$ is a relation-specific linear layer, $g(\cdot)$ is the $tanh$ function, $\mathbf{W} \in \mathbb{R}^{d\times d\times k}$ is a 3-way tensor.

Factor Models. The latent factor models (Jenatton et al., 2012) (Sutskever et al., 2009) attempt to capturing the second-order correlations between entities by a quadratic form. The score function is defined as $f_r(h,t) = \mathbf{h}^\top \mathbf{W_r t}$. RESCAL is a collective matrix factorization model which is also a common method in knowledge base embedding (Nickel et al., 2011) (Nickel et al., 2012).

3 Methods

3.1 TransG: A Generative Model for Embedding

As just mentioned, only one single translation vector for a relation may be insufficient to model multiple relation semantics. In this paper, we propose to use Bayesian non-parametric infinite mix-

ture embedding model (Griffiths and Ghahramani, 2011). The generative process of the model is as follows:

1. For an entity $e \in E$:
   (a) Draw each entity embedding mean vector from a standard normal distribution as a prior: $\mathbf{u_e} \backsim \mathcal{N}(\mathbf{0}, \mathbf{1})$.
2. For a triple $(h, r, t) \in \Delta$:
   (a) Draw a semantic component from Chinese Restaurant Process for this relation: $\pi_{r,m} \sim CRP(\beta)$.
   (b) Draw a head entity embedding vector from a normal distribution: $\mathbf{h} \backsim \mathcal{N}(\mathbf{u_h}, \sigma_h^2\mathbf{E})$.
   (c) Draw a tail entity embedding vector from a normal distribution: $\mathbf{t} \backsim \mathcal{N}(\mathbf{u_t}, \sigma_t^2\mathbf{E})$.
   (d) Draw a relation embedding vector for this semantics: $\mathbf{u_{r,m}} = \mathbf{t} - \mathbf{h} \backsim \mathcal{N}(\mathbf{u_t} - \mathbf{u_h}, (\sigma_h^2 + \sigma_t^2)\mathbf{E})$.

where $\mathbf{u_h}$ and $\mathbf{u_t}$ indicate the mean embedding vector for head and tail respectively, σ_h and σ_t indicate the variance of corresponding entity distribution respectively, and $\mathbf{u_{r,m}}$ is the m-th component translation vector of relation r. Chinese Restaurant Process (CRP) is a Dirichlet Process and it can automatically detect semantic components. In this setting, we obtain the score function as below:

$$\begin{aligned}\mathbb{P}\{(h,r,t)\} &\propto \sum_{m=1}^{M_r} \pi_{r,m}\mathbb{P}(\mathbf{u_{r,m}}|h,t) \\ &= \sum_{m=1}^{M_r} \pi_{r,m} e^{-\frac{||\mathbf{u_h}+\mathbf{u_{r,m}}-\mathbf{u_t}||_2^2}{\sigma_h^2+\sigma_t^2}} \quad (1)\end{aligned}$$

where $\pi_{r,m}$ is the mixing factor, indicating the weight of i-th component and M_r is the number of semantic components for the relation r, which is learned from the data automatically by the CRP.

Inspired by Fig.1, TransG leverages a mixture of relation component vectors for a specific relation. Each component represents a specific latent meaning. By this way, TransG could distinguish multiple relation semantics. Notably, the CRP could generate multiple semantic components when it is necessary and the relation semantic component number M_r is learned adaptively from the data.

Table 1: Statistics of datasets

Data	WN18	FB15K	WN11	FB13
#Rel	18	1,345	11	13
#Ent	40,943	14,951	38,696	75,043
#Train	141,442	483,142	112,581	316,232
#Valid	5,000	50,000	2,609	5,908
#Test	5,000	59,071	10,544	23,733

3.2 Explanation from the Geometry Perspective

Similar to previous studies, TransG has geometric explanations. In the previous methods, when the relation r of triple (h, r, t) is given, the geometric representations are fixed, as $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. However, TransG generalizes this geometric principle to:

$$\begin{aligned}m^*_{(h,r,t)} = \underset{m=1...M_r}{\arg\max} &\left(\pi_{r,m} e^{-\frac{||\mathbf{u_h}+\mathbf{u_{r,m}}-\mathbf{u_t}||_2^2}{\sigma_h^2+\sigma_t^2}}\right) \\ &\mathbf{h} + \mathbf{u_{r,m^*_{(h,r,t)}}} \approx \mathbf{t} \quad (2)\end{aligned}$$

where $m^*_{(h,r,t)}$ is the index of primary component. Though all the components contribute to the model, the primary one contributes the most due to the exponential effect ($exp(\cdot)$). When a triple (h, r, t) is given, TransG works out the index of primary component then translates the head entity to the tail one with the primary translation vector.

For most triples, there should be only one component that have significant non-zero value as $\left(\pi_{r,m} e^{-\frac{||\mathbf{u_h}+\mathbf{u_{r,m}}-\mathbf{u_t}||_2^2}{\sigma_h^2+\sigma_t^2}}\right)$ and the others would be small enough, due to the exponential decay. This property reduces the noise from the other semantic components to better characterize multiple relation semantics. In detail, $(\mathbf{t} - \mathbf{h})$ is almost around only one translation vector $\mathbf{u_{r,m^*_{(h,r,t)}}}$ in TransG. Under the condition $m \neq m^*_{(h,r,t)}$, $\left(\frac{||\mathbf{u_h}+\mathbf{u_{r,m}}-\mathbf{u_t}||_2^2}{\sigma_h^2+\sigma_t^2}\right)$ is very large so that the exponential function value is very small. This is why the primary component could represent the corresponding semantics.

To summarize, previous studies make translation identically for all the triples of the same relation, but TransG automatically selects the best translation vector according to the specific semantics of a triple. Therefore, TransG could focus on the specific semantic embedding to avoid much noise from the other unrelated semantic components and result in promising improvements than existing methods. Note that, all the components in

Table 2: Evaluation results on link prediction

Datasets	WN18				FB15K			
Metric	Mean Rank		HITS@10(%)		Mean Rank		HITS@10(%)	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
Unstructured (Bordes et al., 2011)	315	304	35.3	38.2	1,074	979	4.5	6.3
RESCAL (Nickel et al., 2012)	1,180	1,163	37.2	52.8	828	683	28.4	44.1
SE(Bordes et al., 2011)	1,011	985	68.5	80.5	273	162	28.8	39.8
SME(bilinear) (Bordes et al., 2012)	526	509	54.7	61.3	284	158	31.3	41.3
LFM (Jenatton et al., 2012)	469	456	71.4	81.6	283	164	26.0	33.1
TransE (Bordes et al., 2013)	263	251	75.4	89.2	243	125	34.9	47.1
TransH (Wang et al., 2014)	401	388	73.0	82.3	212	87	45.7	64.4
TransR (Lin et al., 2015b)	238	225	79.8	92.0	198	77	48.2	68.7
CTransR (Lin et al., 2015b)	**231**	**218**	79.4	92.3	199	75	48.4	70.2
PTransE (Lin et al., 2015a)	N/A	N/A	N/A	N/A	207	58	51.4	84.6
KG2E (He et al., 2015)	362	348	80.5	93.2	183	69	47.5	71.5
TransG (this paper)	357	345	**84.5**	**94.9**	**152**	**50**	**55.9**	**88.2**

TransG have their own contributions, but the primary one makes the most.

3.3 Training Algorithm

The maximum data likelihood principle is applied for training. As to the non-parametric part, $\pi_{r,m}$ is generated from the CRP with Gibbs Sampling, similar to (He et al., 2015) and (Griffiths and Ghahramani, 2011). A new component is sampled for a triple (h,r,t) with the below probability:

$$\mathbb{P}(m_{r,new}) = \frac{\beta e^{-\frac{||\mathbf{h}-\mathbf{t}||_2^2}{\sigma_h^2+\sigma_t^2+2}}}{\beta e^{-\frac{||\mathbf{h}-\mathbf{t}||_2^2}{\sigma_h^2+\sigma_t^2+2}} + \mathbb{P}\{(h,r,t)\}} \quad (3)$$

where $\mathbb{P}\{(h,r,t)\}$ is the current posterior probability. As to other parts, in order to better distinguish the true triples from the false ones, we maximize the ratio of likelihood of the true triples to that of the false ones. Notably, the embedding vectors are initialized by (Glorot and Bengio, 2010). Putting all the other constraints together, the final objective function is obtained, as follows:

$$\min_{\mathbf{u_h},\mathbf{u_{r,m}},\mathbf{u_t}} \mathcal{L}$$

$$\mathcal{L} = -\sum_{(h,r,t)\in\Delta} ln\left(\sum_{m=1}^{M_r}\pi_{r,m}e^{-\frac{||\mathbf{u_h}+\mathbf{u_{r,m}}-\mathbf{u_t}||_2^2}{\sigma_h^2+\sigma_t^2}}\right)$$

$$+\sum_{(h',r',t')\in\Delta'} ln\left(\sum_{m=1}^{M_r}\pi_{r',m}e^{-\frac{||\mathbf{u_{h'}}+\mathbf{u_{r',m}}-\mathbf{u_{t'}}||_2^2}{\sigma_{h'}^2+\sigma_{t'}^2}}\right)$$

$$+C\left(\sum_{r\in R}\sum_{m=1}^{M_r}||\mathbf{u_{r,m}}||_2^2+\sum_{e\in E}||\mathbf{u_e}||_2^2\right) \quad (4)$$

where Δ is the set of golden triples and Δ' is the set of false triples. C controls the scaling degree. E is the set of entities and R is the set of relations. Noted that the mixing factors π and the variances σ are also learned jointly in the optimization.

SGD is applied to solve this optimization problem. In addition, we apply a trick to control the parameter updating process during training. For those very impossible triples, the update process is skipped. Hence, we introduce a similar condition as TransE (Bordes et al., 2013) adopts: the training algorithm will update the embedding vectors only if the below condition is satisfied:

$$\frac{\mathbb{P}\{(h,r,t)\}}{\mathbb{P}\{(h',r',t')\}} = \frac{\sum_{m=1}^{M_r}\pi_{r,m}e^{-\frac{||\mathbf{u_h}+\mathbf{u_{r,m}}-\mathbf{u_t}||_2^2}{\sigma_h^2+\sigma_t^2}}}{\sum_{m=1}^{M_{r'}}\pi_{r',m}e^{-\frac{||\mathbf{u_{h'}}+\mathbf{u_{r',m}}-\mathbf{u_{t'}}||_2^2}{\sigma_{h'}^2+\sigma_{t'}^2}}} \leq M_r e^{\gamma} \quad (5)$$

where $(h,r,t)\in\Delta$ and $(h',r',t')\in\Delta'$. γ controls the updating condition.

As to the efficiency, in theory, the time complexity of TransG is bounded by a small constant M compared to TransE, that is $O(\text{TransG}) = O(M \times O(\text{TransE}))$ where M is the number of semantic components in the model. Note that TransE is the fastest method among translation-based methods. The experiment of Link Prediction shows that TransG and TransE would converge at around 500 epochs, meaning there is also no significant difference in convergence speed. In experiment, TransG takes 4.8s for one iteration on FB15K while TransR costs 136.8s and PTransE

Table 3: Evaluation results on FB15K by mapping properties of relations(%)

Tasks	Predicting Head(HITS@10)				Predicting Tail(HITS@10)			
Relation Category	1-1	1-N	N-1	N-N	1-1	1-N	N-1	N-N
Unstructured (Bordes et al., 2011)	34.5	2.5	6.1	6.6	34.3	4.2	1.9	6.6
SE(Bordes et al., 2011)	35.6	62.6	17.2	37.5	34.9	14.6	68.3	41.3
SME(bilinear) (Bordes et al., 2012)	30.9	69.6	19.9	38.6	28.2	13.1	76.0	41.8
TransE (Bordes et al., 2013)	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50.0
TransH (Wang et al., 2014)	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
TransR (Lin et al., 2015b)	78.8	89.2	34.1	69.2	79.2	37.4	90.4	72.1
CTransR (Lin et al., 2015b)	81.5	89.0	34.7	71.2	80.8	38.6	90.1	73.8
PTransE (Lin et al., 2015a)	90.1	92.0	58.7	86.1	90.1	**70.7**	87.5	88.7
KG2E (He et al., 2015)	92.3	93.7	**66.0**	69.6	92.6	67.9	94.4	73.4
TransG (this paper)	**93.0**	**96.0**	62.5	**86.8**	**92.8**	68.1	**94.5**	**88.8**

costs 1200.0s on the same computer for the same dataset.

4 Experiments

Our experiments are conducted on four public benchmark datasets that are the subsets of Wordnet and Freebase, respectively. The statistics of these datasets are listed in Tab.1. Experiments are conducted on two tasks : Link Prediction and Triple Classification. To further demonstrate how the proposed model approaches multiple relation semantics, we present semantic component analysis at the end of this section.

4.1 Link Prediction

Link prediction concerns knowledge graph completion: when given an entity and a relation, the embedding models predict the other missing entity. More specifically, in this task, we predict t given $(h, r, *)$, or predict h given $(*, r, t)$. The WN18 and FB15K are two benchmark datasets for this task. Note that many AI tasks could be enhanced by Link Prediction such as relation extraction (Hoffmann et al., 2011).

Evaluation Protocol. We adopt the same protocol used in previous studies. For each testing triple (h, r, t), we corrupt it by replacing the tail t (or the head h) with every entity e in the knowledge graph and calculate a probabilistic score of this corrupted triple (h, r, e) (or (e, r, t)) with the score function $f_r(h, e)$. After ranking these scores in descending order, we obtain the rank of the original triple. There are two metrics for evaluation: the averaged rank (Mean Rank) and the proportion of testing triple whose rank is not larger than 10 (HITS@10). This is called "Raw" setting. When we filter out the corrupted triples that exist in the training, validation, or test datasets, this is the"Filter" setting. If a corrupted triple exists in the knowledge graph, ranking it ahead the original triple is also acceptable. To eliminate this case, the "Filter" setting is preferred. In both settings, a lower Mean Rank and a higher HITS@10 mean better performance.

Implementation. As the datasets are the same, we directly report the experimental results of several baselines from the literature, as in (Bordes et al., 2013), (Wang et al., 2014) and (Lin et al., 2015b). We have attempted several settings on the validation dataset to get the best configuration. For example, we have tried the dimensions of 100, 200, 300, 400. Under the "bern." sampling strategy, the optimal configurations are: learning rate $\alpha = 0.001$, the embedding dimension $k = 100$, $\gamma = 2.5$, $\beta = 0.05$ on WN18; $\alpha = 0.0015$, $k = 400$, $\gamma = 3.0$, $\beta = 0.1$ on FB15K. Note that all the symbols are introduced in "Methods". We train the model until it converges.

Results. Evaluation results on WN18 and FB15K are reported in Tab.2 and Tab.3[1]. We observe that:

1. TransG outperforms all the baselines obviously. Compared to TransR, TransG makes improvements by 2.9% on WN18 and 26.0% on FB15K, and the averaged semantic component number on WN18 is 5.67 and that on FB15K is 8.77. This result demonstrates capturing multiple relation semantics would benefit embedding.

[1]Note that correctly regularized TransE can produce much better performance than what were reported in the ogirinal paper, see (García-Durán et al., 2015).

Table 4: Different clusters in WN11 and FB13 relations.

Relation	Cluster	Triples (Head, Tail)
PartOf	Location	(Capital of Utah, Beehive State), (Hindustan, Bharat) ...
	Composition	(Monitor, Television), (Bush, Adult Body), (Cell Organ, Cell)...
Religion	Catholicism	(Cimabue, Catholicism), (St.Catald, Catholicism) ...
	Others	(Michal Czajkowsk, Islam), (Honinbo Sansa, Buddhism) ...
DomainRegion	Abstract	(Computer Science, Security System), (Computer Science, PL)..
	Specific	(Computer Science, Router), (Computer Science, Disk File) ...
Profession	Scientist	(Michael Woodruf, Surgeon), (El Lissitzky, Architect)...
	Businessman	(Enoch Pratt, Entrepreneur), (Charles Tennant, Magnate)...
	Writer	(Vlad. Gardin, Screen Writer), (John Huston, Screen Writer) ...

2. The model has a bad Mean Rank score on the WN18 dataset. Further analysis shows that there are 24 testing triples (0.5% of the testing set) whose ranks are more than 30,000, and these few cases would lead to about 150 mean rank loss. Among these triples, there are 23 triples whose tail or head entities have never been co-occurring with the corresponding relations in the training set. In one word, there is no sufficient training data for those relations and entities.

3. Compared to CTransR, TransG solves the multiple relation semantics problem much better for two reasons. Firstly, CTransR clusters the entity pairs for a specific relation and then performs embedding for each cluster, but TransG deals with embedding and multiple relation semantics simultaneously, where the two processes can be enhanced by each other. Secondly, CTransR models a triple by only one cluster, but TransG applies a mixture to refine the embedding.

Our model is almost insensitive to the dimension if that is sufficient. For the dimensions of $100, 200, 300, 400$, the HITS@10 of TransG on FB15 are $81.8\%, 84.0\%, 85.8\%, 88.2\%$, while those of TransE are $47.1\%, 48.5\%, 51.3\%, 49.2\%$.

4.2 Triple Classification

In order to testify the discriminative capability between true and false facts, triple classification is conducted. This is a classical task in knowledge base embedding, which aims at predicting whether a given triple (h, r, t) is correct or not. WN11 and FB13 are the benchmark datasets for this task. Note that evaluation of classification needs negative samples, and the datasets have already provided negative triples.

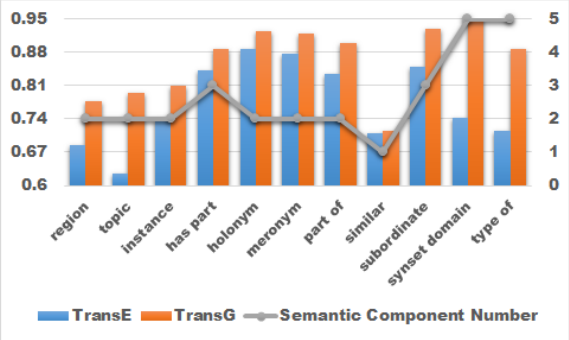


Figure 3: Accuracies of each relations in WN11 for triple classification. The right y-axis is the number of semantic components, corresponding to the lines.

Evaluation Protocol. The decision process is very simple as follows: for a triple (h, r, t), if $f_r(h, t)$ is below a threshold σ_r, then positive; otherwise negative. The thresholds $\{\sigma_r\}$ are determined on the validation dataset.

Table 5: Triple classification: accuracy(%) for different embedding methods.

Methods	WN11	FB13	AVG.
LFM	73.8	84.3	79.0
NTN	70.4	87.1	78.8
TransE	75.9	81.5	78.7
TransH	78.8	83.3	81.1
TransR	85.9	82.5	84.2
CTransR	85.7	N/A	N/A
KG2E	85.4	85.3	85.4
TransG	**87.4**	**87.3**	**87.4**

Implementation. As all methods use the same datasets, we directly re-use the results of different methods from the literature. We have attempted several settings on the validation dataset to find

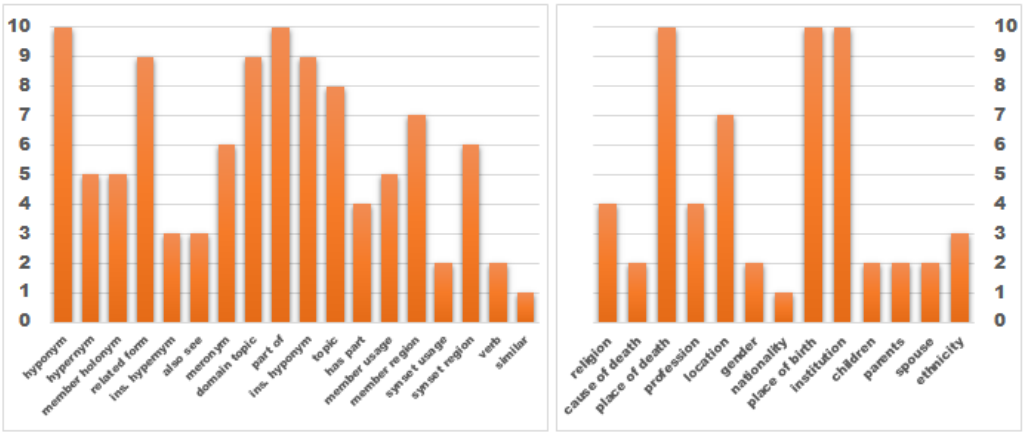


Figure 4: Semantic component number on WN18 (left) and FB13 (right).

the best configuration. The optimal configurations of TransG are as follows: "bern" sampling, learning rate $\alpha = 0.001$, $k = 50$, $\gamma = 6.0$, $\beta = 0.1$ on WN11, and "bern" sampling, $\alpha = 0.002$, $k = 400$, $\gamma = 3.0$, $\beta = 0.1$ on FB13.

Results. Accuracies are reported in Tab.5 and Fig.3. The following are our observations:

1. TransG outperforms all the baselines remarkably. Compared to TransR, TransG improves by 1.7% on WN11 and 5.8% on FB13, and the averaged semantic component number on WN11 is 2.63 and that on FB13 is 4.53. This result shows the benefit of capturing multiple relation semantics for a relation.

2. The relations, such as "Synset Domain" and "Type Of", which hold more semantic components, are improved much more. In comparison, the relation "Similar" holds only one semantic component and is almost not promoted. This further demonstrates that capturing multiple relation semantics can benefit embedding.

4.3 Semantic Component Analysis

In this subsection, we analyse the number of semantic components for different relations and list the component number on the dataset WN18 and FB13 in Fig.4.

Results. As Fig. 4 and Tab. 4 show, we have the following observations:

1. Multiple semantic components are indeed necessary for most relations. Except for relations such as "Also See", "Synset Usage" and "Gender", all other relations have more than one semantic component.

2. Different components indeed correspond to different semantics, justifying the theoretical analysis and effectiveness of TransG. For example, "Profession" has at least three semantics: scientist-related as $(\text{ElLissitzky}, \text{Architect})$, businessman-related as $(\text{EnochPratt}, \text{Entrepreneur})$ and writer-related as $(\text{Vlad.Gardin}, \text{ScreenWriter})$.

3. WN11 and WN18 are different subsets of Wordnet. As we know, the semantic component number is decided on the triples in the dataset. Therefore, It's reasonable that similar relations, such as "Synset Domain" and "Synset Usage" may hold different semantic numbers for WN11 and WN18.

5 Conclusion

In this paper, we propose a generative Bayesian non-parametric infinite mixture embedding model, TransG, to address a new issue, multiple relation semantics, which can be commonly seen in knowledge graph. TransG can discover the latent semantics of a relation automatically and leverage a mixture of relation components for embedding. Extensive experiments show our method achieves substantial improvements against the state-of-the-art baselines.

6 Acknowledgements

This work was partly supported by the National Basic Research Program (973 Program) under grant No. 2012CB316301/2013CB329403, the National Science Foundation of China under grant

No. 61272227/61332007, and the Beijing Higher Education Young Elite Teacher Project.

References

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Jason Weston, Ronan Collobert, Yoshua Bengio, et al. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-fifth AAAI Conference on Artificial Intelligence*.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2012. Joint learning of words and meaning representations for open-text semantic parsing. In *International Conference on Artificial Intelligence and Statistics*, pages 127–135.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259.

Miao Fan, Qiang Zhou, Emily Chang, and Thomas Fang Zheng. 2014. Transition-based knowledge graph embedding with relational mapping properties. In *Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation*, pages 328–337.

Alberto García-Durán, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. 2015. Combining two and three-way embeddings models for link prediction in knowledge bases. *CoRR*, abs/1506.00999.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256.

Thomas L Griffiths and Zoubin Ghahramani. 2011. The indian buffet process: An introduction and review. *The Journal of Machine Learning Research*, 12:1185–1224.

Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically smooth knowledge graph embedding. In *Proceedings of ACL*.

Shizhu He, Kang Liu, Guoliang Ji, and Jun Zhao. 2015. Learning to represent knowledge graphs with gaussian embedding. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 623–632. ACM.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 541–550. Association for Computational Linguistics.

Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. 2012. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015a. Modeling relation paths for representation learning of knowledge bases. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934.

Ilya Sutskever, Joshua B Tenenbaum, and Ruslan Salakhutdinov. 2009. Modelling relational data using bayesian clustered tensor factorization. In *Advances in neural information processing systems*, pages 1821–1828.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*.

Han Xiao, Minlie Huang, and Xiaoyan Zhu. 2016. From one point to a manifold: Knowledge graph embedding for precise link prediction. In *IJCAI*.

Question Answering on Freebase via Relation Extraction and Textual Evidence

Kun Xu[1], **Siva Reddy**[2], **Yansong Feng**[1,*], **Songfang Huang**[3] and **Dongyan Zhao**[1]
[1]Institute of Computer Science & Technology, Peking University, Beijing, China
[2]School of Informatics, University of Edinburgh, UK
[3]IBM China Research Lab, Beijing, China
{xukun, fengyansong, zhaody}@pku.edu.cn
siva.reddy@ed.ac.uk
huangsf@cn.ibm.com

Abstract

Existing knowledge-based question answering systems often rely on small annotated training data. While shallow methods like relation extraction are robust to data scarcity, they are less expressive than the deep meaning representation methods like semantic parsing, thereby failing at answering questions involving multiple constraints. Here we alleviate this problem by empowering a relation extraction method with additional evidence from Wikipedia. We first present a neural network based relation extractor to retrieve the candidate answers from Freebase, and then infer over Wikipedia to validate these answers. Experiments on the WebQuestions question answering dataset show that our method achieves an F_1 of 53.3%, a substantial improvement over the state-of-the-art.

1 Introduction

Since the advent of large structured knowledge bases (KBs) like Freebase (Bollacker et al., 2008), YAGO (Suchanek et al., 2007) and DBpedia (Auer et al., 2007), answering natural language questions using those structured KBs, also known as KB-based question answering (or KB-QA), is attracting increasing research efforts from both natural language processing and information retrieval communities.

The state-of-the-art methods for this task can be roughly categorized into two streams. The first is based on semantic parsing (Berant et al., 2013; Kwiatkowski et al., 2013), which typically learns a grammar that can parse natural language to a sophisticated meaning representation language. But such sophistication requires a lot of annotated training examples that contains compositional structures, a practically impossible solution for large KBs such as Freebase. Furthermore, mismatches between grammar predicted structures and KB structure is also a common problem (Kwiatkowski et al., 2013; Berant and Liang, 2014; Reddy et al., 2014).

On the other hand, instead of building a formal meaning representation, information extraction methods retrieve a set of candidate answers from KB using relation extraction (Yao and Van Durme, 2014; Yih et al., 2014; Yao, 2015; Bast and Haussmann, 2015) or distributed representations (Bordes et al., 2014; Dong et al., 2015). Designing large training datasets for these methods is relatively easy (Yao and Van Durme, 2014; Bordes et al., 2015; Serban et al., 2016). These methods are often good at producing an answer irrespective of their correctness. However, handling compositional questions that involve multiple entities and relations, still remains a challenge. Consider the question *what mountain is the highest in north america*. Relation extraction methods typically answer with all the mountains in *North America* because of the lack of sophisticated representation for the mathematical function *highest*. To select the correct answer, one has to retrieve all the heights of the mountains, and sort them in descending order, and then pick the first entry. We propose a method based on textual evidence which can answer such questions without solving the mathematic functions implicitly.

Knowledge bases like Freebase capture real world facts, and Web resources like Wikipedia provide a large repository of sentences that validate or support these facts. For example, a sentence in Wikipedia says, *Denali (also known as Mount McKinley, its former official name) is the highest mountain peak in North America, with a summit elevation of 20,310 feet (6,190 m) above sea level.* To answer our example question against a KB using a relation extractor, we can use this sentence

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2326–2336,
Berlin, Germany, August 7-12, 2016.

as external evidence, filter out wrong answers and pick the correct one.

Using textual evidence not only mitigates representational issues in relation extraction, but also alleviates the data scarcity problem to some extent. Consider the question, *who was queen isabella's mother*. Answering this question involves predicting two constraints hidden in the word *mother*. One constraint is that the answer should be the *parent* of *Isabella*, and the other is that the answer's *gender* is *female*. Such words with multiple latent constraints have been a pain-in-the-neck for both semantic parsing and relation extraction, and requires larger training data (this phenomenon is coined as sub-lexical compositionality by Wang et al. (2015)). Most systems are good at triggering the *parent* constraint, but fail on the other, i.e., the answer entity should be *female*. Whereas the textual evidence from Wikipedia, *... her mother was Isabella of Barcelos ...*, can act as a further constraint to answer the question correctly.

We present a novel method for question answering which infers on both structured and unstructured resources. Our method consists of two main steps as outlined in §2. In the first step we extract answers for a given question using a structured KB (here Freebase) by jointly performing entity linking and relation extraction (§3). In the next step we validate these answers using an unstructured resource (here Wikipedia) to prune out the wrong answers and select the correct ones (§4). Our evaluation results on a benchmark dataset WebQuestions show that our method outperforms existing state-of-the-art models. Details of our experimental setup and results are presented in §5. Our code, data and results can be downloaded from `https://github.com/syxu828/QuestionAnsweringOverFB`.

2 Our Method

Figure 1 gives an overview of our method for the question "*who did shaq first play for*". We have two main steps: (1) inference on Freebase (*KB-QA* box); and (2) further inference on Wikipedia (*Answer Refinement* box). Let us take a close look into step 1. Here we perform *entity linking* to identify a topic entity in the question and its possible Freebase entities. We employ a *relation extractor* to predict the potential Freebase relations that could exist between the entities in the question and the answer entities. Later we perform a *joint inference* step over the entity linking and relation extraction

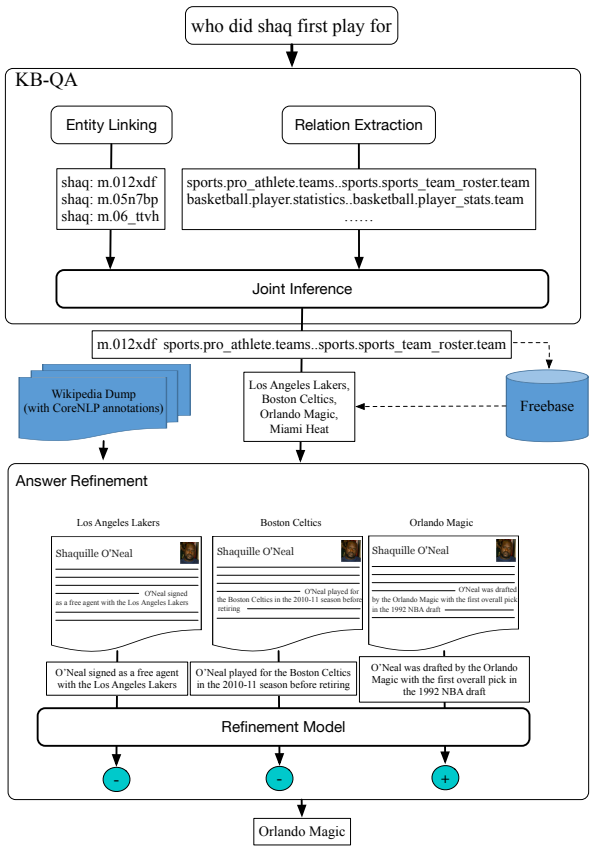


Figure 1: An illustration of our method to find answers for the given question *who did shaq first play for*.

results to find the best entity-relation configuration which will produce a list of candidate answer entities. In the step 2, we refine these candidate answers by applying an *answer refinement* model which takes the Wikipedia page of the topic entity into consideration to filter out the wrong answers and pick the correct ones.

While the overview in Figure 1 works for questions containing single Freebase relation, it also works for questions involving multiple Freebase relations. Consider the question *who plays anakin skywalker in star wars 1*. The actors who are the answers to this question should satisfy the following constraints: (1) the actor played *anakin skywalker*; and (2) the actor played in *star wars 1*. Inspired by Bao et al. (2014), we design a dependency tree-based method to handle such multi-relational questions. We first decompose the original question into a set of *sub-questions* using syntactic patterns which are listed in Appendix. The final answer set of the original question is obtained by intersecting the answer sets of all its sub-questions. For the

example question, the sub-questions are *who plays anakin skywalker* and *who plays in star wars 1*. These sub-questions are answered separately over Freebase and Wikipedia, and the intersection of their answers to these sub-questions is treated as the final answer.

3 Inference on Freebase

Given a sub-question, we assume the question word[1] that represents the answer has a distinct KB relation r with an entity e found in the question, and predict a single KB triple $(e, r, ?)$ for each sub-question (here ? stands for the answer entities). The QA problem is thus formulated as an information extraction problem that involves two sub-tasks, i.e., entity linking and relation extraction. We first introduce these two components, and then present a joint inference procedure which further boosts the overall performance.

3.1 Entity Linking

For each question, we use hand-built sequences of part-of-speech categories to identify all possible named entity mention spans, e.g., the sequence *NN* (*shaq*) may indicate an entity. For each mention span, we use the entity linking tool S-MART[2] (Yang and Chang, 2015) to retrieve the top 5 entities from Freebase. These entities are treated as candidate entities that will eventually be disambiguated in the joint inference step. For a given mention span, S-MART first retrieves all possible entities of Freebase by surface matching, and then ranks them using a statistical model, which is trained on the frequency counts with which the surface form occurs with the entity.

3.2 Relation Extraction

We now proceed to identify the relation between the answer and the entity in the question. Inspired by the recent success of neural network models in KB question-answering (Yih et al., 2015; Dong et al., 2015), and the success of syntactic dependencies for relation extraction (Liu et al., 2015; Xu et al., 2015), we propose a Multi-Channel Convolutional Neural Network (MCCNN) which could exploit both syntactic and sentential information for relation extraction.

[1] who, when, what, where, how, which, why, whom, whose.

[2] S-MART demo can be accessed at http://msre2edemo.azurewebsites.net/

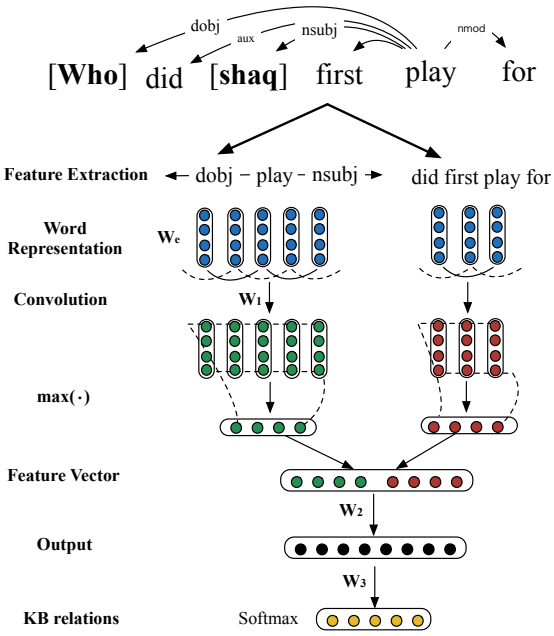


Figure 2: Overview of the multi-channel convolutional neural network for relation extraction. W_e is the word embedding matrix, W_1 is the convolution matrix, W_2 is the activation matrix and W_3 is the classification matrix.

3.2.1 MCCNNs for Relation Classification

In MCCNN, we use two channels, one for syntactic information and the other for sentential information. The network structure is illustrated in Figure 2. Convolution layer tackles an input of varying length returning a fixed length vector (we use max pooling) for each channel. These fixed length vectors are concatenated and then fed into a *softmax* classifier, the output dimension of which is equal to the number of predefined relation types. The value of each dimension indicates the confidence score of the corresponding relation.

Syntactic Features We use the shortest path between an entity mention and the question word in the dependency tree[3] as input to the first channel. Similar to Xu et al. (2015), we treat the path as a concatenation of vectors of words, dependency edge directions and dependency labels, and feed it to the convolution layer. Note that, the entity mention and the question word are excluded from the dependency path so as to learn a more general relation representation in syntactic level. As shown in Figure 2, the dependency path between *who* and *shaq* is $\leftarrow$ *dobj* – *play* – *nsubj* $\rightarrow$.

[3] We use Stanford CoreNLP dependency parser (Manning et al., 2014).

Sentential Features This channel takes the words in the sentence as input excluding the question word and the entity mention. As illustrated in Figure 2, the vectors for *did, first, play* and *for* are fed into this channel.

3.2.2 Objective Function and Learning

The model is learned using pairs of question and its corresponding gold relation from the training data. Given an input question x with an annotated entity mention, the network outputs a vector $o(x)$, where the entry $o_k(x)$ is the probability that there exists the *k*-th relation between the entity and the expected answer. We denote $t(x) \in \mathbb{R}^{K\times 1}$ as the target distribution vector, in which the value for the gold relation is set to 1, and others to 0. We compute the cross entropy error between $t(x)$ and $o(x)$, and further define the objective function over the training data as:

$$J(\theta) = -\sum_{x}\sum_{k=1}^{K} t_k(x)\log o_k(x) + \lambda||\theta||_2^2$$

where θ represents the weights, and λ the $L2$ regularization parameters. The weights θ can be efficiently computed via back-propagation through network structures. To minimize $J(\theta)$, we apply stochastic gradient descent (SGD) with AdaGrad (Duchi et al., 2011).

3.3 Joint Entity Linking & Relation Extraction

A pipeline of entity linking and relation extraction may suffer from error propagations. As we know, entities and relations have strong selectional preferences that certain entities do not appear with certain relations and vice versa. Locally optimized models could not exploit these implicit *bi-directional* preferences. Therefore, we use a joint model to find a globally optimal entity-relation assignment from local predictions. The key idea behind is to leverage various clues from the two local models and the KB to rank a correct entity-relation assignment higher than other combinations. We describe the learning procedure and the features below.

3.3.1 Learning

Suppose the pair (e_{gold}, r_{gold}) represents the gold entity/relation pair for a question q. We take all our entity and relation predictions for q, create a list of entity and relation pairs $\{(e_0, r_0), (e_1, r_1), ..., (e_n, r_n)\}$ from q and rank them using an SVM rank classifier (Joachims, 2006) which is trained to predict a rank for each pair. Ideally higher rank indicates the prediction is closer to the gold prediction. For training, SVM rank classifier requires a ranked or scored list of entity-relation pairs as input. We create the training data containing ranked input pairs as follows: if both $e_{pred} = e_{gold}$ and $r_{pred} = r_{gold}$, we assign it with a score of 3. If only the entity or relation equals to the gold one (i.e., $e_{pred} = e_{gold}$, $r_{pred} \neq r_{gold}$ or $e_{pred} \neq e_{gold}$, $r_{pred} = r_{gold}$), we assign a score of 2 (encouraging partial overlap). When both entity and relation assignments are wrong, we assign a score of 1.

3.3.2 Features

For a given entity-relation pair, we extract the following features which are passed as an input vector to the SVM ranker above:

Entity Clues. We use the score of the predicted entity returned by the entity linking system as a feature. The number of word overlaps between the entity mention and entity's Freebase name is also included as a feature. In Freebase, most entities have a relation *fb:description* which describes the entity. For instance, in the running example, *shaq* is linked to three potential entities *m.06_ttvh (Shaq Vs. Television Show)*, *m.05n7bp (Shaq Fu Video Game)* and *m.012xdf (Shaquille O'Neal)*. Interestingly, the word *play* only appears in the description of *Shaquille O'Neal* and it occurs three times. We count the content word overlap between the given question and the entity's description, and include it as a feature.

Relation Clues. The score of relation returned by the MCCNNs is used as a feature. Furthermore, we view each relation as a *document* which consists of the training questions that this relation is expressed in. For a given question, we use the sum of the *tf-idf* scores of its words with respect to the relation as a feature. A Freebase relation r is a concatenation of a series of fragments $r = r_1.r_2.r_3$. For instance, the three fragments of *people.person.parents* are *people*, *person* and *parents*. The first two fragments indicate the Freebase type of the subject of this relation, and the third fragment indicates the object type, in our case the answer type. We use an indicator feature to denote if the surface form of the third fragment (here *parents*) appears in the question.

Answer Clues. The above two feature classes indicate local features. From the entity-relation (e, r)

pair, we create the query triple $(e, r, ?)$ to retrieve the answers, and further extract features from the answers. These features are non-local since we require both e and r to retrieve the answer. One such feature is using the co-occurrence of the answer type and the question word based on the intuition that question words often indicate the answer type, e.g., the question word *when* usually indicates the answer type *type.datetime*. Another feature is the number of answer entities retrieved.

4 Inference on Wikipedia

We use the best ranked entity-relation pair from the above step to retrieve candidate answers from Freebase. In this step, we validate these answers using Wikipedia as our unstructured knowledge resource where most statements in it are verified for factuality by multiple people.

Our refinement model is inspired by the intuition of how people refine their answers. If you ask someone: *who did shaq first play for*, and give them four candidate answers (*Los Angeles Lakers*, *Boston Celtics*, *Orlando Magic* and *Miami Heat*), as well as access to Wikipedia, that person might first determine that the question is about *Shaquille O'Neal*, then go to *O'Neal*'s Wikipedia page, and search for the sentences that contain the candidate answers as evidence. By analyzing these sentences, one can figure out whether a candidate answer is correct or not.

4.1 Finding Evidence from Wikipedia

As mentioned above, we should first find the Wikipedia page corresponding to the *topic entity* in the given question. We use Freebase API to convert Freebase entity to Wikipedia page. We extract the content from the Wikipedia page and process it with *Wikifier* (Cheng and Roth, 2013) which recognizes Wikipedia entities, which can further be linked to Freebase entities using Freebase API. Additionally we use Stanford CoreNLP (Manning et al., 2014) for tokenization and entity co-reference resolution. We search for the sentences containing the candidate answer entities retrieved from Freebase. For example, the Wikipedia page of *O'Neal* contains a sentence "*O'Neal was drafted by the Orlando Magic with the first overall pick in the 1992 NBA draft*", which is taken into account by the *refinement model* (our inference model on Wikipedia) to discriminate whether *Orlando Magic* is the answer for the given question.

4.2 Refinement Model

We treat the refinement process as a binary classification task over the candidate answers, i.e., correct (positive) and incorrect (negative) answer. We prepare the training data for the refinement model as follows. On the training dataset, we first infer on Freebase to retrieve the candidate answers. Then we use the annotated gold answers of these questions and Wikipedia to create the training data. Specifically, we treat the sentences that contain correct/incorrect answers as positive/negative examples for the refinement model. We use LIBSVM (Chang and Lin, 2011) to learn the weights for classification.

Note that, in the Wikipedia page of the topic entity, we may collect more than one sentence that contain a candidate answer. However, not all sentences are relevant, therefore we consider the candidate answer as correct *if* at least there is one positive evidence. On the other hand, sometimes, we may not find any evidence for the candidate answer. In these cases, we fall back to the results of the KB-based approach.

4.3 Lexical Features

Regarding the features used in LIBSVM, we use the following lexical features extracted from the question and a Wikipedia sentence. Formally, given a question $q = <q_1, ... q_n>$ and an evidence sentence $s = <s_1, ... s_m>$, we denote the tokens of q and s by q_i and s_j, respectively. For each pair (q, s), we identify a set of all possible token pairs (q_i, s_j), the occurrences of which are used as features. As learning proceeds, we hope to learn a higher weight for a feature like (*first*, *drafted*) and a lower weight for (*first*, *played*).

5 Experiments

In this section we introduce the experimental setup, the main results and detailed analysis of our system.

5.1 Training and Evaluation Data

We use the WebQuestions (Berant et al., 2013) dataset, which contains 5,810 questions crawled via Google Suggest service, with answers annotated on Amazon Mechanical Turk. The questions are split into training and test sets, which contain 3,778 questions (65%) and 2,032 questions (35%), respectively. We further split the training questions into 80%/20% for development.

To train the MCCNNs and the joint inference model, we need the gold standard relations of the questions. Since this dataset contains only question-answer pairs and annotated topic entities, instead of relying on gold relations we rely on *surrogate* gold relations which produce answers that have the highest overlap with gold answers. Specifically, for a given question, we first locate the topic entity e in the Freebase graph, then select 1-hop and 2-hop relations connected to the topic entity as relation candidates. The 2-hop relations refer to the n-ary relations of Freebase, i.e., first hop from the subject to a mediator node, and the second from the mediator to the object node. For each relation candidate r, we issue the query (e, r, ?) to the KB, and label the relation that produces the answer with minimal F_1-loss against the gold answer, as the *surrogate* gold relation. From the training set, we collect 461 relations to train the MCCNN, and the target prediction during testing time is over these relations.

5.2 Experimental Settings

We have 6 dependency tree patterns based on Bao et al. (2014) to decompose the question into sub-questions (See Appendix). We initialize the word embeddings with Turian et al. (2010)'s word representations with dimensions set to 50. The hyper parameters in our model are tuned using the development set. The window size of MCCNN is set to 3. The sizes of the hidden layer 1 and the hidden layer 2 of the two MCCNN channels are set to 200 and 100, respectively. We use the Freebase version of Berant et al. (2013), containing 4M entities and 5,323 relations.

5.3 Results and Discussion

We use the average question-wise F_1 as our evaluation metric.[4] To give an idea of the impact of different configurations of our method, we compare the following with existing methods.

Structured. This method involves inference on Freebase only. First the entity linking (EL) system is run to predict the topic entity. Then we run the relation extraction (RE) system and select the best relation that can occur with the topic entity. We choose this entity-relation pair to predict the answer.

[4]We use the evaluation script available at `http://www-nlp.stanford.edu/software/sempre`.

Method	average F_1
Berant et al. (2013)	35.7
Yao and Van Durme (2014)	33.0
Xu et al. (2014)	39.1
Berant and Liang (2014)	39.9
Bao et al. (2014)	37.5
Bordes et al. (2014)	39.2
Dong et al. (2015)	40.8
Yao (2015)	44.3
Bast and Haussmann (2015)	49.4
Berant and Liang (2015)	49.7
Reddy et al. (2016)	50.3
Yih et al. (2015)	52.5
This work	
Structured	44.1
Structured + Joint	47.1
Structured + Unstructured	47.0
Structured + Joint + Unstructured	**53.3**

Table 1: Results on the test set.

Structured + Joint. In this method instead of the above pipeline, we perform joint EL and RE as described in §3.3.

Structured+Unstructured. We use the pipelined EL and RE along with inference on Wikipedia as described in §4.

Structured + Joint + Unstructured. This is our main model. We perform inference on Freebase using joint EL and RE, and then inference on Wikipedia to validate the results. Specifically, we treat the top two predictions of the joint inference model as the candidate subject and relation pairs, and extract the corresponding answers from each pair, take the union, and filter the answer set using Wikipedia.

Table 1 summarizes the results on the test data along with the results from the literature.[5] We can see that joint EL and RE performs better than the default pipelined approach, and outperforms most semantic parsing based models, except (Berant and Liang, 2015) which searches partial logical forms in strategic order by combining imitation learning and agenda-based parsing. In addition, inference on unstructured data helps the default model. The joint EL and RE combined with inference on unstructured data further improves the default pipelined model by 9.2% (from 44.1% to 53.3%), and achieves a new state-of-the-art result beating the previous reported best result of Yih et al. (2015) (with one-tailed t-test significance of $p < 0.05$).

[5]We use development data for all our ablation experiments. Similar trends are observed on both development and test results.

	Entity Linking Accuracy	Relation Extraction Accuracy
Isolated Model	79.8	45.9
Joint Inference	**83.2**	**55.3**

Table 2: Impact of the joint inference on the development set

Method	average F_1
Structured (syntactic)	38.1
Structured (sentential)	38.7
Structured (syntactic + sentential)	40.1
Structured + Joint (syntactic)	43.6
Structured + Joint (sentential)	44.1
Structured + Joint (syntactic + sentential)	**45.8**

Table 3: Impact of different MCCNN channels on the development set.

5.3.1 Impact of Joint EL & RE

From Table 1, we can see that the joint EL & RE gives a performance boost of 3% (from 44.1 to 47.1). We also analyze the impact of joint inference on the individual components of EL & RE.

We first evaluate the EL component using the gold entity annotations on the development set. As shown in Table 2, for 79.8% questions, our entity linker can correctly find the gold standard topic entities. The joint inference improves this result to 83.2%, a 3.4% improvement. Next we use the *surrogate* gold relations to evaluate the performance of the RE component on the development set. As shown in Table 2, the relation prediction accuracy increases by 9.4% (from 45.9% to 55.3%) when using the joint inference.

5.3.2 Impact of the Syntactic and the Sentential Channels

Table 3 presents the results on the impact of individual and joint channels on the end QA performance. When using a single-channel network, we tune the parameters of only one channel while switching off the other channel. As seen, the sentential features are found to be more important than syntactic features. We attribute this to the short and noisy nature of WebQuestions questions due to which syntactic parser wrongly parses or the shortest dependency path does not contain sufficient information to predict a relation. By using both the channels, we see further improvements than using any one of the channels.

Question & Answers
1. what is the largest nation in europe Before: Kazakhstan, Turkey, Russia, ... After: Russia
2. which country in europe has the largest land area Before: Georgia, France, Russia, ... After: Russian Empire, Russia
3. what year did ray allen join the nba Before: 2007, 2003, 1996, 1993, 2012 After: 1996
4. who is emma stone father Before: Jeff Stone, Krista Stone After: Jeff Stone
5. where did john steinbeck go to college Before: Salinas High School, Stanford University After: Stanford University

Table 4: Example questions and corresponding predicted answers before and after using unstructured inference. *Before* uses (*Structured* + *Joint*) model, and *After* uses *Structured* + *Joint* + *Unstructured* model for prediction. The colors *blue* and *red* indicate correct and wrong answers respectively.

5.3.3 Impact of the Inference on Unstructured Data

As shown in Table 1, when structured inference is augmented with the unstructured inference, we see an improvement of 2.9% (from 44.1% to 47.0%). And when *Structured* + *Joint* uses unstructured inference, the performance boosts by 6.2% (from 47.1% to 53.3%) achieving a new state-of-the-art result. For the latter, we manually analyzed the cases in which unstructured inference helps. Table 4 lists some of these questions and the corresponding answers before and after the unstructured inference. We observed the unstructured inference mainly helps for two classes of questions: (1) questions involving aggregation operations (Questions 1-3); (2) questions involving sub-lexical compositionally (Questions 4-5). Questions 1 and 2 contain the predicate $largest$ an aggregation operator. A semantic parsing method should explicitly handle this predicate to trigger $max(.)$ operator. For Question 3, structured inference predicts the Freebase relation *fb:teams..from* retrieving all the years in which *Ray Allen* has played basketball. Note that *Ray Allen* has joined *Connecticut University*'s team in *1993* and *NBA* from *1996*. To answer this question a semantic parsing system would require a *min*(·) operator along with an additional constraint that the year corresponds to the *NBA*'s term. Interestingly, without having to explicitly model these complex predicates, the unstructured inference helps in answering these questions more accurately. Questions 4-5 involve sub-lexical com-

positionally (Wang et al., 2015) predicates *father* and *college*. For example in Question 5, the user queries for the colleges that *John Steinbeck* attended. However, Freebase defines the relation *fb:education..institution* to describe a person's educational information without discriminating the specific periods such as high school or college. Inference using unstructured data helps in alleviating these representational issues.

5.3.4 Error analysis

We analyze the errors of *Structured + Joint + Unstructured* model. Around 15% of the errors are caused by incorrect entity linking, and around 50% of the errors are due to incorrect relation predictions. The errors in relation extraction are due to (i) insufficient context, e.g., in *what is duncan bannatyne*, neither the dependency path nor sentential context provides enough evidence for the MCCNN model; (ii) unbalanced distribution of relations (3022 training examples for 461 relations) heavily influences the performance of MCCNN model towards frequently seen relations. The remaining errors are the failure of unstructured inference due to insufficient evidence in Wikipedia or misclassification.

Entity Linking. In the entity linking component, we had handcrafted POS tag patterns to identify entity mentions, e.g., DT-JJ-NN (noun phrase), NN-IN-NN (prepositional phrase). These patterns are designed to have high recall. Around 80% of entity linking errors are due to incorrect entity prediction even when the correct mention span was found.

Question Decomposition. Around 136 questions (15%) of dev data contains compositional questions, leading to 292 sub-questions (around 2.1 subquestions for a compositional question). Since our question decomposition component is based on manual rules, one question of interest is how these rules perform on other datasets. By human evaluation, we found these rules achieves 95% on a more general but complex QA dataset QALD-5[6].

5.3.5 Limitations

While our unstructured inference alleviates representational issues to some extent, we still fail at modeling compositional questions such as *who is the mother of the father of prince william* involving multi-hop relations and the inter alia. Our current assumption that unstructured data could provide evidence for questions may work only for frequently typed queries or for popular domains like movies, politics and geography. We note these limitations and hope our result will foster further research in this area.

[6] http://qald.sebastianwalter.org/index.php?q=5

6 Related Work

Over time, the QA task has evolved into two main streams – QA on unstructured data, and QA on structured data. TREC QA evaluations (Voorhees and Tice, 1999) were a major boost to unstructured QA leading to richer datasets and sophisticated methods (Wang et al., 2007; Heilman and Smith, 2010; Yao et al., 2013; Yih et al., 2013; Yu et al., 2014; Yang et al., 2015; Hermann et al., 2015). While initial progress on structured QA started with small toy domains like GeoQuery (Zelle and Mooney, 1996), recent focus has shifted to large scale structured KBs like Freebase, DBPedia (Unger et al., 2012; Cai and Yates, 2013; Berant et al., 2013; Kwiatkowski et al., 2013; Xu et al., 2014), and on noisy KBs (Banko et al., 2007; Carlson et al., 2010; Krishnamurthy and Mitchell, 2012; Fader et al., 2013; Parikh et al., 2015). An exciting development in structured QA is to exploit multiple KBs (with different schemas) at the same time to answer questions jointly (Yahya et al., 2012; Fader et al., 2014; Zhang et al., 2016). QALD tasks and linked data initiatives are contributing to this trend.

Our model combines the best of both worlds by inferring over structured and unstructured data. Though earlier methods exploited unstructured data for KB-QA (Krishnamurthy and Mitchell, 2012; Berant et al., 2013; Yao and Van Durme, 2014; Reddy et al., 2014; Yih et al., 2015), these methods do not rely on unstructured data at test time. Our work is closely related to Joshi et al. (2014) who aim to answer noisy telegraphic queries using both structured and unstructured data. Their work is limited in answering single relation queries. Our work also has similarities to Sun et al. (2015) who does question answering on unstructured data but enrich it with Freebase, a reversal of our pipeline. Other line of very recent related work include Yahya et al. (2016) and Savenkov and Agichtein (2016).

Our work also intersects with relation extraction methods. While these methods aim to predict a relation between two entities in order to pop-

ulate KBs (Mintz et al., 2009; Hoffmann et al., 2011; Riedel et al., 2013), we work with sentence level relation extraction for question answering. Krishnamurthy and Mitchell (2012) and Fader et al. (2014) adopt open relation extraction methods for QA but they require hand-coded grammar for parsing queries. Closest to our extraction method is Yao and Van Durme (2014) and Yao (2015) who also uses sentence level relation extraction for QA. Unlike them, we can predict multiple relations per question, and our MCCNN architecture is more robust to unseen contexts compared to their logistic regression models.

Dong et al. (2015) were the first to use MCCNN for question answering. Yet our approach is very different in spirit to theirs. Dong et al. aim to maximize the similarity between the distributed representation of a question and its answer entities, whereas our network aims to predict Freebase relations. Our search space is several times smaller than theirs since we do not require potential answer entities beforehand (the number of relations is much smaller than the number of entities in Freebase). In addition, our method can explicitly handle compositional questions involving multiple relations, whereas Dong et al. learn latent representation of relation joins which is difficult to comprehend. Moreover, we outperform their method by 7 points even without unstructured inference.

7 Conclusion and Future Work

We have presented a method that could infer both on structured and unstructured data to answer natural language questions. Our experiments reveal that unstructured inference helps in mitigating representational issues in structured inference. We have also introduced a relation extraction method using MCCNN which is capable of exploiting syntax in addition to sentential features. Our main model which uses joint entity linking and relation extraction along with unstructured inference achieves the state-of-the-art results on WebQuestions dataset. A potential application of our method is to improve KB-question answering using the documents retrieved by a search engine.

Since we pipeline structured inference first and then unstructured inference, our method is limited by the coverage of Freebase. Our future work involves exploring other alternatives such as treating structured and unstructured data as two independent resources in order to overcome the knowledge gaps in either of the two resources.

Acknowledgments

We would like to thank Weiwei Sun, Liwei Chen, and the anonymous reviewers for their helpful feedback. This work is supported by National High Technology R&D Program of China (Grant No. 2015AA015403, 2014AA015102), Natural Science Foundation of China (Grant No. 61202233, 61272344, 61370055) and the joint project with IBM Research. For any correspondence, please contact Yansong Feng.

Appendix

The syntax-based patterns for question decomposition are shown in Figure 3. The first four patterns are designed to extract sub-questions from simple questions, while the latter two are designed for complex questions involving clauses.

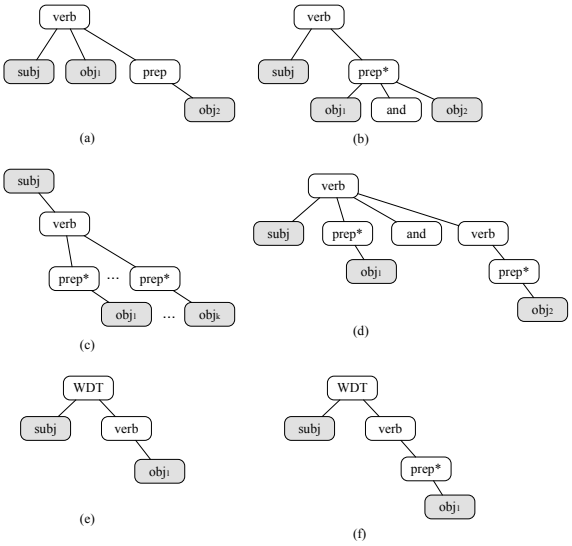


Figure 3: Syntax-based patterns for question decomposition.

References

Sren Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. 2007. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC*.

Michele Banko, Michael J Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. 2007. Open information extraction for the web. In *IJCAI*.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *ACL*.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *CIKM*.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *ACL*.

Jonathan Berant and Percy Liang. 2015. Imitation learning of agenda-based semantic parsers. *Transactions of the Association for Computational Linguistics*, 3:545–558.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*.

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *EMNLP*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Qingqing Cai and Alexander Yates. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM TIST*, 2(3):27.

Xiao Cheng and Dan Roth. 2013. Relational inference for wikification. In *ACL*.

Li Dong, Furu Wei, Ming Zhou, and Ke Xu. 2015. Question answering over freebase with multi-column convolutional neural networks. In *ACL-IJCNLP*.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Anthony Fader, Luke S. Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *ACL*.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *SIGKDD*.

Michael Heilman and Noah A Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Thorsten Joachims. 2006. Training linear svms in linear time. In *SIGKDD*.

Mandar Joshi, Uma Sawant, and Soumen Chakrabarti. 2014. Knowledge graph and corpus driven segmentation and answer inference for telegraphic entity-seeking queries. In *EMNLP*.

Jayant Krishnamurthy and Tom M Mitchell. 2012. Weakly supervised training of semantic parsers. In *EMNLP-CoNLL*.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *EMNLP*.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng WANG. 2015. A dependency-based neural network for relation classification. In *ACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL*.

Ankur P. Parikh, Hoifung Poon, and Kristina Toutanova. 2015. Grounded semantic parsing for complex knowledge extraction. In *NAACL*.

Siva Reddy, Mirella Lapata, and Mark Steedman. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association of Computational Linguistics*, pages 377–392.

Siva Reddy, Oscar Täckström, Michael Collins, Tom Kwiatkowski, Dipanjan Das, Mark Steedman, and Mirella Lapata. 2016. Transforming Dependency Structures to Logical Forms for Semantic Parsing. *Transactions of the Association for Computational Linguistics*, 4.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

Denis Savenkov and Eugene Agichtein. 2016. When a knowledge base is not enough: Question answering over knowledge bases with external text data. In *SIGIR*.

Iulian Vlad Serban, Alberto García-Durán, Çaglar Gülçehre, Sungjin Ahn, Sarath Chandar, Aaron C. Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. In *ACL*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*.

Huan Sun, Hao Ma, Wen-tau Yih, Chen-Tse Tsai, Jingjing Liu, and Ming-Wei Chang. 2015. Open domain question answering via semantic enrichment. In *WWW*.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 384–394.

Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. 2012. Template-based question answering over rdf data. In *WWW*.

Ellen M Voorhees and Dawn M. Tice. 1999. The trec-8 question answering track report. In *TREC*.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP-CoNLL*.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *ACL*.

Kun Xu, Sheng Zhang, Yansong Feng, and Dongyan Zhao. 2014. Answering natural language questions via phrasal semantic parsing. In *Natural Language Processing and Chinese Computing - Third CCF Conference, NLPCC 2014, Shenzhen, China, December 5-9, 2014. Proceedings*, pages 333–344.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2015. Semantic relation classification via convolutional neural networks with simple negative sampling. In *EMNLP*.

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. 2012. Natural language questions for the web of data. In *EMNLP*.

Mohamed Yahya, Denilson Barbosa, Klaus Berberich, Qiuyue Wang, and Gerhard Weikum. 2016. Relationship queries on extended knowledge graphs. In *WSDM*.

Yi Yang and Ming-Wei Chang. 2015. S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking. In *ACL-IJNLP*.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *EMNLP*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. In *ACL*.

Xuchen Yao, Benjamin Van Durme, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *NAACL*.

Xuchen Yao. 2015. Lean question answering over freebase from scratch. In *NAACL*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. In *ACL*.

Wen-tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In *ACL*.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *ACL-IJCNLP*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI*.

Yuanzhe Zhang, Shizhu He, Kang Liu, and Jun Zhao. 2016. A joint model for question answering over multiple knowledge bases. In *AAAI*.

Vector-space topic models for detecting Alzheimer's disease

Maria Yancheva
Department of Computer Science,
University of Toronto
Toronto, Ontario, Canada
yancheva@cs.toronto.edu

Frank Rudzicz
Toronto Rehabilitation Institute; and
Department of Computer Science,
University of Toronto
Toronto, Ontario, Canada
frank@cs.toronto.edu

Abstract

Semantic deficit is a symptom of language impairment in Alzheimer's disease (AD). We present a generalizable method for automatic generation of information content units (ICUs) for a picture used in a standard clinical task, achieving high recall, 96.8%, of human-supplied ICUs. We use the automatically generated topic model to extract semantic features, and train a random forest classifier to achieve an F-score of 0.74 in binary classification of controls versus people with AD using a set of only 12 features. This is comparable to results (0.72 F-score) with a set of 85 *manual* features. Adding semantic information to a set of standard lexicosyntactic and acoustic features improves F-score to 0.80. While control and dementia subjects discuss the same topics in the same contexts, controls are more informative per second of speech.

1 Introduction

Alzheimer's disease (AD) is the most common cause of neurodegenerative dementia, and affects more than 24.3 million people worldwide (Ballard et al., 2011). Importantly, early detection enables some therapeutic intervention and disease-modifying treatment (Sperling et al., 2011). Longitudinal studies of people with autopsy-confirmed AD indicate that linguistic changes are detectable in the prodromal stages of the disease; these include a decline in grammatical complexity, word-finding difficulties, and semantic content deficiencies, such as low idea density (i.e., the ratio of semantic units to the total number of words in a speech sample), and low efficiency (i.e., the rate of semantic units over the duration of the speech sample) (Bayles and Kaszniak, 1987; Snowdon et al., 1996; Le et al., 2011; Ahmed et al., 2013b). In the present study, we investigate methods of automatically assessing the semantic content of speech, and use it to distinguish people with AD from healthy older adults.

A standard clinical task for eliciting spontaneous speech, with high sensitivity to language in early AD, is picture description. In it, a participant is asked to provide a free-form verbal description of a visual stimulus (Goodglass and Kaplan, 1983; Bayles and Kaszniak, 1987). The picture is associated with a set of human-supplied information content units (**hsICUs**) representing components of the image, such as subjects, objects, locations, and actions (Croisile et al., 1996). The semantic content of the elicited speech can then be scored by counting the hsICUs present in the description. Previous studies found that, even in the earliest stages, descriptions by those with AD are less informative compared to those of healthy older adults, producing fewer information units out of a pre-defined list of units, and having less relevant content and lower efficiency (Hier et al., 1985; Croisile et al., 1996; Giles et al., 1996; Ahmed et al., 2013a).

Using a pre-defined list of annotated hsICUs is subject to several limitations: (i) it is *subjective* — different authors use a different number of hsICUs for the same picture (e.g., from 7 to 25 for *Cookie Theft* in the Boston Diagnostic Aphasia Examination (BDAE)) (Hier et al., 1985; Croisile et al., 1996; Forbes-McKay and Venneri, 2005; Lai et al., 2009); (ii) it *may not be optimal* for detecting linguistic impairment — the manually-annotated hsICUs are neither exhaustive of all details present in the picture, nor necessarily reflective of the content units which differ most across groups; (iii) it is *not generalizable* — hsICUs are specific to a particular picture, and new visual stimuli (e.g.,

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2337–2346,
Berlin, Germany, August 7-12, 2016.

required for longitudinal assessments) need to be annotated manually. In addition to requiring time and effort, this may result in inconsistencies, since the methodology for identifying hsICUs was never clearly defined in previous work.

Automatic scoring of semantic content in speech to detect cognitive impairment has so far required manual hsICUs. Hakkani-Tür et al. (2010) used unigram recall among hsICUs in the Western Aphasia Battery's *Picnic* picture (Kertesz, 1982) and obtained a correlation of 0.93 with manual hsICU counts. Pakhomov et al. (2010) counted N-grams ($N = 1, 2, 3, 4$) extracted from a list of hsICUs for the *Cookie Theft* picture to assess semantic content in the speech of patients with frontotemporal lobar degeneration. Fraser et al. (2016) counted instances of lexical tokens extracted from a list of hsICUs, using dependency parses of *Cookie Theft* picture descriptions, and combined them with other lexicosyntactic and acoustic features to obtain classification accuracy of 81.9% in identifying people with AD from controls. While those automated methods for scoring the information content in speech used manual hsICUs, we have found none that attempted to produce ICUs automatically.

In this paper, we present a generalizable method for automatically generating information content units for any given picture (or spontaneous speech task), using reference speech. Since clinical data can be sparse, we present a method for building word vector representations using a large general corpus, then augment it with local context windows from a smaller clinical corpus. We evaluate the generated ICUs by computing recall of hsICUs and use the constructed topic models to compare the speech of participants with and without dementia, and compute topic alignment. Second, we automatically score new picture descriptions by learning semantic features extracted from these generated ICU models, using a random forest classifier; we assess performance with recall, precision, and F-score. Third, we propose a set of clinically-relevant features for identifying AD based on differences in topic, topic context, idea density and idea efficiency.

2 Methodology

2.1 Data

DementiaBank is one of the largest public, longitudinal datasets of spontaneous speech from individuals with and without dementia. It was collected at the University of Pittsburgh (Becker et al., 1994) and contains verbal descriptions of the standard *Cookie Theft* picture (Goodglass and Kaplan, 1983), along with manual transcriptions.

In our study, we use 255 speech samples from participants diagnosed with probable or possible AD (collectively referred to as the 'AD' class), and 241 samples from healthy controls (collectively referred to as the 'CT' class), see Table 1. We remove all CHAT-format annotations (MacWhinney, 2015), filled pauses (e.g., '*ah*' and '*um*'), phonological fragments (e.g., '*b b boy*' becomes '*boy*'), repairs (e.g., '*in the in the kitchen*' becomes '*in the kitchen*'), non-standard forms (e.g., '*gonna*' becomes '*going to*'), and punctuation (e.g., commas are removed). These corrections are all provided in the database. We ignore transcripts of the investigator's speech, as irrelevant. Subject data were randomly partitioned into training, validation, and test sets using a 60-20-20 split.

Table 1: Distribution of dataset transcriptions.

Class	Subjects	Samples	Tokens
AD	168	255	24,753
CT	98	241	26,654
Total	266	496	51,407

2.2 Human-supplied ICUs (hsICUs)

We combine all hsICUs in previous work for the *Cookie Theft* picture (Hier et al., 1985; Croisile et al., 1996; Forbes-McKay and Venneri, 2005; Lai et al., 2009) with hsICUs obtained from a speech language pathologist (SLP) at the Toronto Rehabilitation Institute (TRI). The annotations of the SLP overlap completely with previously identified hsICUs, except for one (*apron*). The first three columns of Table 2 summarize these manually-produced hsICUs.

2.3 Automatic generation of ICUs

Our novel method of identifying ICUs is based on simple topic modelling using clusters of global word-vector representations from picture descriptions. First, we train a word-vector model on a large normative general-purpose corpus, allowing us to avoid sparsity in the clinical data's word-word co-occurrence matrix. Then, we extract the vector representations of words in the Dementia-

Table 2: Information units above the double line are human-supplied ICUs (hsICUs) found in previous work, except those marked with † which were annotated by an SLP for this study; those below are additionally analyzed. Over 1,000 clustering configurations based on word vectors extracted from *Control* and *Dementia* reference transcriptions, μ is the mean of the scaled distance (Eq. 1) of each hsICU to its closest cluster centroid, σ is the standard deviation, and $\delta = (\mu_{dementia} - \mu_{control})$. Statistical significance of δ was tested using an independent two-sample, two-tailed t-test; *** $= p < .001$, ** $= p < .01$, * $= p < .05$, ns = not significant.

			Control		**Dementia**			
Type	**ID**	**hsICU**	μ	σ	μ	σ	δ	p
Subject	S1	boy	-0.510	0.102	-0.860	0.204	**-0.350**	***
Subject	S2	girl	-0.357	0.203	-0.545	0.284	**-0.187**	***
Subject	S3	woman	0.171	0.468	0.140	0.433	-0.031	ns
Subject	S4	mother	-0.533	0.206	-0.187	0.300	**0.345**	***
Place	P1	kitchen	0.667	0.650	0.901	0.710	**0.234**	***
Place	P2	exterior	1.985	0.601	1.947	0.530	-0.039	ns
Object	O1	cookie	-1.057	0.221	-0.943	0.230	**0.114**	***
Object	O2	jar	0.243	0.486	0.146	0.453	**-0.097**	***
Object	O3	stool	-0.034	0.674	-0.162	0.623	**-0.128**	***
Object	O4	sink	-0.839	0.433	-0.600	0.631	**0.239**	***
Object	O5	plate	0.564	0.593	0.639	0.608	**0.076**	**
Object	O6	dishcloth	*4.509*	1.432	*3.989*	1.154	**-0.521**	***
Object	O7	water	-0.418	0.582	-0.567	0.530	**-0.149**	***
Object	O8	cupboard	0.368	0.613	0.453	0.637	**0.085**	**
Object	O9	window	-0.809	0.425	-0.298	0.452	**0.511**	***
Object	O10	cabinet	2.118	0.556	2.154	0.496	0.036	ns
Object	O11	dishes	0.037	0.503	-0.083	0.406	**-0.120**	***
Object	O12	curtains	-0.596	0.594	0.121	0.707	**0.717**	***
Object	O13	faucet	1.147	0.567	1.016	0.547	**-0.131**	***
Object	O14	floor	-0.466	0.384	-0.932	0.451	**-0.466**	***
Object	O15	counter	0.202	0.427	0.449	0.323	**0.247**	***
Object	O16	apron†	-0.140	0.433	0.181	0.688	**0.321**	***
Action	A1	boy *stealing* cookies	1.219	0.373	0.746	0.462	**-0.473**	***
Action	A2	boy/stool *falling* over	-0.064	0.465	-0.304	0.409	**-0.240**	***
Action	A3	woman *washing* dishes	-0.058	0.539	0.009	0.611	**0.068**	**
Action	A4	woman *drying* dishes	-0.453	0.469	-0.385	0.541	**0.068**	**
Action	A5	water *overflowing* in sink	0.147	0.804	0.282	0.791	**0.135**	***
Action	A6	girl's actions towards boy, girl *asking* for a cookie	0.800	0.555	0.620	0.861	**-0.179**	***
Action	A7	woman *daydreaming*, unaware or unconcerned about overflow	0.049	0.774	0.092	0.561	0.043	ns
Action	A8	dishes already washed *sitting* on worktop	-0.224	0.535	-0.597	0.426	**-0.373**	***
Action	A9	woman being *indifferent* to the children	0.781	0.795	0.881	0.585	**0.100**	**
Relation		brother	2.297	0.510	1.916	0.344	**-0.380**	***
Relation		sister	0.862	0.273	0.737	0.349	**-0.125**	***
Relation		son	2.140	0.443	1.818	0.312	**-0.322**	***
Relation		daughter	0.916	0.356	0.904	0.421	-0.012	ns

Bank corpus, and optionally augment them with local context windows from the clinical dataset.

We use GloVe v1.2 (Pennington et al., 2014) to obtain embedded word representations and train on a combined corpus of Wikipedia 2014[1] + Gigaword 5[2]. The trained model consists of 400,000 word vectors, in 50 dimensions.

Transcriptions in DementiaBank are lowercased and tokenized using NLTK v3.1, and each word token is converted to its vector space representation using the trained GloVe model. There are a total of 26,654 word vectors (1,087 unique vectors) in the control data, and 24,753 (1,131 unique) in the dementia data. Since we aim to construct a model of semantic content, only nouns and verbs are retained prior to clustering. The resulting dataset consists of 9,330 word vectors (801 unique vectors) in the control data, and 8,021 (843 unique) in the dementia data.

We use k-means clustering with whitening, initialization with the Forgy method, and a distortion threshold of 10^{-5} as the stopping condition, where distortion is defined as the sum of the distances between each vector and its corresponding centroid. We train a *control cluster model* on the control training set (see Fig. 1 for a 2D projection of cluster vectors using principal component analysis), and a *dementia cluster model* on the dementia training set. Clusters represent topics, or groups of semantically related word vectors, discussed by the respective group of subjects. While prior work is based on hsICUs that are *expected* to be discussed by *healthy* speakers, we construct a separate cluster model for the control and dementia groups since it is unclear whether the topics discussed by both groups overlap. We vary k $(= 1, 5, 10, 15, 20, 30, 40, 50)$, completing 1,000 runs for each value, and use the Elbow method to select the optimal number of clusters on the respective validation set. The optimal setting, $k = 10$, optimizes the tradeoff between the percentage of variance explained by the clusters, and their total number. The resulting clusters represent topics that can be compared against hsICUs.

3 Experiments

3.1 Recall of hsICUs

In order to assess (i) how well the automatically generated clusters match clinical hsICUs for this

[1]http://dumps.wikimedia.org/enwiki/20140102/

[2]https://catalog.ldc.upenn.edu/LDC2011T07

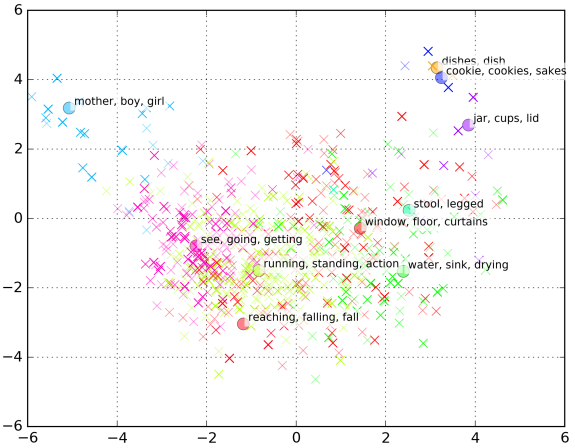


Figure 1: *Control* cluster model. The word vectors belonging to a given cluster are shown in the same colour. The most frequent words in each cluster are displayed.

image, and (ii) how much the two generated topic models differ, we analyze the vector space distance between each hsICU and its closest cluster centroid ($d_{Euclidean}$) in each of the *control* and *dementia* models. Since some clusters are more dispersed than others, we need to scale the distance appropriately. To do so, for each cluster in each model, we compute the mean distortion, μ_{cl}, of the vectors in the cluster, and the associated standard deviation σ_{cl}. For each hsICU vector, we compute the scaled distance between the vector and its closest cluster centroid in each generated model as follows:

$$d_{scaled} = \frac{(d_{Euclidean} - \mu_{cl})}{\sigma_{cl}} \quad (1)$$

The scaled distance is equivalent to the number of standard deviations above the mean — a value below zero indicates hsICUs which are very close to an automatically generated cluster centroid, while a large positive value indicates hsICUs that are far from a cluster centroid. To account for the fact that k-means is a stochastic algorithm, we perform clustering multiple times and average the results. Table 2 shows the mean, μ, and standard deviation, σ, of d_{scaled}, for each hsICU, over 1,000 cluster configurations for each model.

To quantify the recall of hsICUs using each generated cluster model, we consider hsICUs with $\mu \leq 3.0$ to be *recalled* (i.e., the distance to the assigned cluster centroid is not greater than those of 99.7% of the datapoints in the cluster, given a Gaussian distribution of distortion). The recall of

hsICUs, for both the control and dementia models, is 96.8%. Since the optimal number of generated clusters is $k = 10$, while the number of hsICUs is 31, multiple hsICUs can be grouped in related *themes* (e.g., one automatically generated cluster corresponds to the description of animate subjects in the picture, capturing four hsICUs: S1–S4). Both the control and dementia models do not recall hsICU O6, *dishcloth*, which suggests that it is a topic that neither study group discusses. All remaining hsICUs are recalled by both the control and dementia models, indicating that the hsICU topics are discussed by both groups.

However, to assess whether they are discussed to the same *extent*, i.e. to evaluate whether the two topic models differ, we conducted an independent two-sample two-tailed t-test to compare the mean scaled distance, μ, of each hsICU to its closest cluster centroid, in each cluster model (see δ in Table 2). As anticipated, since they involve inference of attention, the control model is better at accounting for the topics of the overflowing sink and the mother's indifference: *overflowing* ($t(1998) = -3.78$, $p < .001$); *sink* ($t(1998) = -9.85$, $p < .001$); *indifferent* ($t(1998) = -3.20$, $p < .01$). While there is no significant difference in the term *woman* between the two groups, the control model predicts the term *mother* better than the dementia model ($t(1998) = -30.05$, $p < .001$). To investigate whether healthy participants are more likely to identify relations between the subjects than participants with cognitive impairment, we repeated the recall experiment with the following new hsICUs: *brother*, *sister*, *son*, *daughter*. Interestingly, the dementia cluster model contains a cluster which aligns significantly more closely, than any in the control model, with all four of these relation words: *brother* ($t(1998) = 19.53$, $p < .001$); *sister* ($t(1998) = 8.93$, $p < .001$); *son* ($t(1998) = 18.78$, $p < .001$). While the control participants mention relation words as often as the participants with dementia[3], the generated cluster models show that the ratio of relation words to non-relation words is higher for the dementia group[4].

The new hsICU, *apron*, which was not identified in previous literature but was labelled by an SLP for this study, is significantly more likely to be discussed by the control population ($t(1998) = -12.46$, $p < .001$), suggesting at the importance of details for distinguishing cognitively impaired individuals. In a similar vein, control participants are significantly more likely to identify objects in the background of the scene, such as the *window* ($t(1998) = -26.04$, $p < .001$), *curtains* ($t(1998) = -24.54$, $p < .001$), *cupboard* ($t(1998) = -3.03$, $p < .01$), or *counter* ($t(1998) = -14.59$, $p < .001$).

3.2 Cluster model alignment

While prior work counted the *frequency* with which fixed topics are mentioned, our data-driven cluster models allow greater exploration of differences between the set of topics discussed by each subject group, and the alignment between them. Since prior work has found that subjects with cognitive impairment produce more irrelevant content, we quantify the amount of dispersion within each cluster through the standard deviation of its distortion and its type-to-token ratio (TTR), as shown in Table 3. Further, we compute directional alignment between pairs of clusters in each model. For each cluster in one model, alignment is determined by computing the closest cluster in the other model for each vector, and taking the majority assignment label (see a in Table 3). To quantify the alignment, the Euclidean distance of each vector to the assigned cluster in the other model is computed, scaled by the mean and standard deviation of the cluster distortion; the mean of the scaled distance, μ_a, is reported in Table 3.

To quantify the alignment of clusters in each model, we consider clusters to be *recalled* if their distance to the closest cluster in the other model is $\mu_a \leq 3$. Notably, all control clusters (C0-C9) are recalled by the dementia model, while one dementia cluster, D7, is not recalled by the control model. This exemplifies the fact that while the dementia group mentions all topics discussed by controls, they also mention a sufficient number of extraneous terms which constitute a new heterogeneous topic cluster, having the highest TTR.

[3] An independent two-sample two-tailed t-test of the effect of group on the *number of occurrences* of each relation word shows no statistical significance: *son* ($t(494) = 0.65$, $p > .05$), *daughter* ($t(494) = 0.63$, $p > .05$), *brother* ($t(494) = 0.97$, $p > .05$), *sister* ($t(494) = 1.65$, $p > .05$).

[4] An independent two-sample two-tailed t-test of the effect of group on this ratio shows a significant difference in the ratio of *sister* to *mother*, with the control group having a lower ratio ($t(494) = -4.10$, $p < .001$).

Table 3: Cluster statistics for control (C*) and dementia (D*) models, with computed cluster alignment. *Cluster words* are the 5 most frequently occurring words. f_{vec} is the fraction of all vectors which belong to the given cluster. μ_{cl} and σ_{cl} are the mean and standard deviation of the cluster distortion. f_n is the fraction of nouns among cluster vectors; $(1 - f_n)$ is the fraction of verbs. *TTR* is the type-to-token ratio. a is the ID of the aligned cluster, and μ_a is the mean scaled distance to the aligned cluster centroid.

	ID	Cluster words	f_{vec}	μ_{cl}	σ_{cl}	f_n	TTR	a	μ_a
Control	C0	window, floor, curtains, plate, kitchen	0.14	5.42	1.18	0.94	0.14	D4	0.69
	C1	dishes, dish	0.04	1.62	1.11	1.00	0.01	D1	0.01
	C2	running, standing, action, hand, counter	0.18	4.97	1.25	0.57	0.22	D8	0.16
	C3	water, sink, drying, overflowing, washing	0.17	5.18	1.13	0.66	0.09	D6	0.04
	C4	stool, legged	0.03	0.53	1.26	0.96	0.01	D4	-0.28
	C5	mother, boy, girl, sister, children	0.11	3.49	1.08	1.00	0.04	D2	-0.08
	C6	cookie, cookies, sakes, cream	0.06	2.00	1.15	1.00	0.01	D0	-0.08
	C7	jar, cups, lid, dried, bowl	0.04	3.88	2.30	0.97	0.04	D5	0.63
	C8	see, going, getting, looks, know	0.18	3.84	1.16	0.38	0.13	D3	0.18
	C9	reaching, falling, fall, summer, growing	0.05	4.18	1.41	0.38	0.16	D8	0.21
Dementia	D0	cookie, cookies, cake, baking, apples	0.07	2.18	0.74	1.00	0.02	C6	0.09
	D1	dishes, dish, eating, bowls, dinner	0.05	1.42	1.72	0.98	0.03	C1	0.05
	D2	boy, girl, mother, sister, lady	0.11	3.63	1.25	0.99	0.05	C5	0.20
	D3	going, see, getting, get, know	0.24	3.67	1.06	0.38	0.11	C8	-0.11
	D4	stool, floor, window, chair, curtains	0.10	5.10	1.00	0.97	0.13	C0	0.08
	D5	jar, cups, jars, dried, honey	0.04	2.00	2.26	0.98	0.03	C7	-0.44
	D6	sink, drying, washing, spilling, overflowing	0.14	5.36	1.20	0.52	0.19	C3	0.36
	D7	mama, huh, alright, johnny, ai	0.01	6.24	1.34	0.95	0.55	C8	**4.13**
	D8	running, fall, falling, reaching, hand	0.18	4.97	1.29	0.47	0.25	C2	0.15
	D9	water, dry, food	0.05	0.39	1.13	1.00	0.01	C3	-0.59

3.3 Local context weighted vectors

Since there is significant overlap in the topics discussed between the control and dementia groups, we proceed by investigating whether the overlapping topics are discussed in the same contexts. To this end, we augment the word vector representations with local context windows from DementiaBank. Each word vector is constructed using a linear combination of its global vector from the trained GloVe model, and the vectors of the $\pm N$ surrounding context words, where each context word is weighted inversely to its distance from the central word:

$$\phi_w = v_w + \sum_{i=-N}^{-1} \alpha_i \times v_i + \sum_{i=1}^{N} \alpha_i \times v_i \qquad (2)$$

Here, ϕ_w is the local-context-weighted vector for word w, v_w is the GloVe vector for word w, v_i is the GloVe vector for word i within the context of w, and α_i is the weighting of word i, inversely and linearly proportional to the distance between context and central word. Following previous work (Fraser and Hirst, 2016), we use a context window of size $N = 3$. We extract local-context-weighted vectors for all control and dementia transcripts, and construct two topic models as before.

To quantify whether the dementia contexts differ significantly from the control contexts for the *same word*, we extract all word usages as local-context-weighted vectors, and find the centroid of the control usages, along with the mean and standard deviation of the control vectors from their centroids. Then, we compute the average scaled Euclidean distance, d_{scaled}, of the dementia vectors from the control centroid, as in Eq. 1. Words with $d_{scaled} > 3$ (i.e., where the dementia context vectors are further from the control centroid than the majority of control context vectors) are considered to have different context usage across the control and dementia groups.

Interestingly, all of the control cluster words are used in the *same contexts* by both healthy participants and those with dementia. However, the *average number of times* these words are used per transcript is significantly higher in the control group (1.07, $s.d. = 0.12$) than in the dementia group (0.77, $s.d. = 0.14$; $t(18) = 1.87$, $p < .05$).

While the two groups discuss the same topics generally and use the same words in the same contexts, not all participants in the dementia group identify all of the control topics or discuss them with the same frequency. A contextual analysis reveals that certain words are discussed in a distinct number of limited contexts, while others are discussed in more varied contexts. For in-

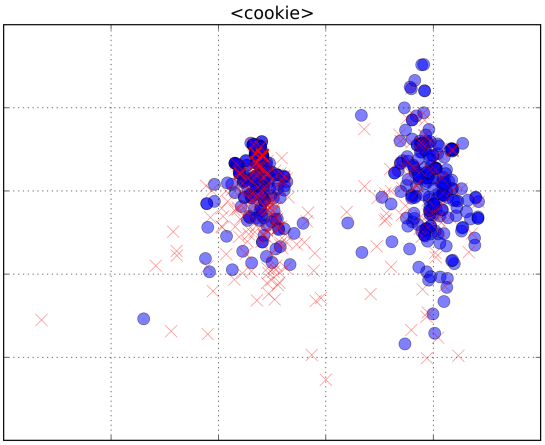


Figure 2: All usages of the word *cookie* in DementiaBank. Control usages are represented with blue circles; dementia with red crosses.

stance, while we identified a control cluster associated with the topic of the *cookie* in Section 3.2, there are two clearly distinct contexts in which this word is used, by both groups, as illustrated in Fig. 2. The two clusters in context space correspond to: (i) the usage of *cookie* in the compound noun phrase *cookie jar*, and (ii) referring to a single *cookie*, e.g. *reaching for a cookie*, *hand her a cookie*, *getting a cookie*.

3.4 Classification

To classify speakers as having AD or not, we extract the following types of features from our automatically-generated cluster models: (i) distance-based metrics for each of the control model clusters, C0–C9, (ii) distance-based metrics for each of the dementia model clusters, D0–D9, (iii) idea density, and (iv) idea efficiency. Given the vectors associated with a transcript's nouns and verbs, feature C_i (and equivalently, D_i) is computed by finding the average scaled distance, d_{scaled} (Eq. 1), of all vectors assigned to cluster C_i. A feature value below zero indicates that the transcript words assigned to the cluster are very well predicted by it (i.e., their distance from the cluster centroid is less than the average cluster distortion). Conversely, clusters which represent topics not discussed in the transcript have large positive feature values. We chose these distance-based metrics to evaluate topic recall in the transcript since a continuous measure is more appropriate for modelling the non-discrete nature of language and semantic similarity. We compute *idea density* as the number of expected topics mentioned[5] divided by the total number of words in the transcript, and *idea efficiency* as the number of expected topics mentioned divided by the total duration of the recording (in seconds). The expected topics used for computation of idea density and idea efficiency are the ICUs from the automatically-produced cluster models.

We perform classification using a random forest, whose parameters are optimized on the validation set, and performance reported on the test set. We vary the following experimental settings: *cluster model* (control; dementia; combined), *feature set* (distance-based; distance-based + idea density + idea efficiency), and *context* (no context; context with $N = 3$). A three-way ANOVA is conducted to examine the effects of these settings on average test F-score. There is a significant interaction between *feature set* and *context*, $F(1, 110) = 9.07$, $p < 0.01$. Simple main effect analysis shows that when using the extended feature set, vectors constructed without local context windows from the clinical dataset yield significantly better results than those with context ($p < 0.001$), but there is no effect when using only distance-based features ($p = 0.87$). There is no main effect of *cluster model* on test performance, $F(2, 117) = 2.30$, $p = 0.11$, which is expected since cluster alignment revealed significant overlap between the topics discussed by the control and dementia groups (Section 3.2). Notably, there is a significant effect of *feature set* on test performance, whereby adding the idea density and idea efficiency features results in significantly higher F-scores, both when using local context for vector construction ($p < 0.05$), and otherwise ($p < 0.001$).

As a baseline, we use a list of hsICUs extracted by Fraser et al. (2016) in a state-of-the-art automated method for separating AD and control speakers in DementiaBank. These features consist of (i) counts of lexical tokens representing hsICUs (e.g., *boy*, *son*, and *brother* are used to identify whether hsICU S1 (Table 2) was discussed, and (ii) Boolean values which indicate whether each hsICU was mentioned or not. Overall, this constitutes 85 features. Additionally, Fraser et al. (2016) identified a list of lexicosyntactic and acoustic (LS&A) features which are indicative of cognitive impairment. We compute the performance of each set of features independently, and then com-

[5] I.e., the number of word vectors in the transcript whose scaled distance is within 3 s.d.'s from the mean cluster distortion of at least one cluster.

Table 4: Binary classification (AD:CT) using a random forest classifier, with 10-fold cross-validation. All cluster models are trained on vectors with no local context. LS&A are lexicosyntactic and acoustic features as described by Fraser et al. (2016). The reported precision, recall, and F-score are a weighted average over the two classes.

Model	Features	Accuracy	Precision	Recall	F-score
Baseline	hsICUs	0.73	0.74	0.73	0.72
Baseline	LS&A	0.76	0.77	0.76	0.76
Baseline	hsICUs + LS&A	0.80	0.80	0.80	**0.80**
control	distance-based	0.68	0.69	0.68	0.68
dementia	distance-based	0.66	0.67	0.66	0.66
combined	distance-based	0.68	0.69	0.68	0.68
control	distance-based + idea density + idea efficiency	0.74	0.76	0.74	0.74
dementia	distance-based + idea density + idea efficiency	0.74	0.75	0.74	0.74
combined	distance-based + idea density + idea efficiency	0.74	0.75	0.74	0.74
control	distance-based + idea density + idea efficiency + LS&A	0.79	0.79	0.79	0.79
dementia	distance-based + idea density + idea efficiency + LS&A	0.77	0.78	0.77	0.77
combined	distance-based + idea density + idea efficiency + LS&A	0.80	0.80	0.80	**0.80**

bine them. Table 4 summarizes the results; the first column indicates the cluster model (e.g., *control* indicates a cluster model trained on the control transcriptions), and the second column specifies the feature set. Our 12 automatically generated features (i.e., the combined set of distance-based measures, idea density, and idea efficiency) result in higher F-scores (0.74) than using 85 manually generated hsICUs (0.72); a two-sample paired t-test shows no difference (using control cluster model: $t(9) = 1.10$, $p = 0.30$; using dementia cluster model: $t(9) = 0.74$, $p = 0.48$) indicating the similarity of our method to the manual gold standard. Furthermore, we match state-of-the-art results (F-score of 0.80) when we augment the set of LS&A features with our automatically generated semantic features.

4 Discussion

We demonstrated a method for generating topic models automatically within the context of clinical assessment, and confirmed that low idea density and low idea efficiency are salient indicators of cognitive impairment. In our data, we also found that speakers with and without Alzheimer's disease generally discuss the *same topics* and in the *same contexts*, although those with AD give more spurious descriptions, as exemplified by the irrelevant topic cluster D7 (Table 3).

Using a fully automated topic generation and feature extraction pipeline, we found a small set of features which perform as well as a large set of manually constructed hsICUs in binary classification experiments, achieving an F-score of 0.80 in 10-fold cross-validation on DementiaBank. The features which correlate most highly with class include: idea efficiency (Pearson's $r = -0.41$), which means that healthy individuals discuss more topics per unit time; distance from cluster C4 ($r = 0.34$), which indicates that speakers with AD focus less on the topic of the *three-legged stool*; and idea density ($r = -0.26$), which shows that healthy speakers need fewer words to express the same number of topics.

While we anticipated that combining a large normative corpus with local context windows from a clinical corpus would produce optimal vectors, using the former exclusively actually performs better. This phenomenon is being investigated. This implies that word-vector representations do not *need* to be adapted with context windows in specific clinical data in order to be effective.

A limitation of the current work is its requirement of high-quality transcriptions of speech, since high word-error rates (WERs) could compromise semantic information. We are therefore generating automatic transcriptions of the DementiaBank audio using the Kaldi speech recognition toolkit[6]. So far, a triphone model with the standard insertion penalty (0) and language model scale (20) on DementiaBank gives the best average WER of $36.7 \pm 3.6\%$ with 10-fold cross-validation. Continued optimization is the subject of ongoing research but preliminary experiments with these transcriptions indicate significantly lower perfor-

[6] http://kaldi.sourceforge.net/

mance of the baseline model (0.68 F-score; $t(9) = 3.52$, $p < 0.01$). While the eventual aim is a completely automatic system, our methodology overcomes several major challenges in the manual semantic annotation of clinical images for cognitive assessment, even with manual transcriptions. Specifically, our methodology is fully objective, sensitive to differences between groups, and generalizable to new stimuli which is especially important if longitudinal analysis is to avoid the so-called 'practice effect' by using multiple stimuli.

Across many domains, to extract useful semantic features (such as idea density and idea efficiency), one needs to first identify information content units in speech or text. Our method can be applied to any picture or contentful stimuli, given a sufficient amount of normative data, with no modification. Although we apply this generalizable method to a single (albeit important) image used in clinical practice in this work, we note that we obtain better accuracies with this completely automated method than a completely manual alternative.

Acknowledgments

The authors would like to thank Selvana Morcos, a speech language pathologist at the Toronto Rehabilitation Institute, for her generous help with providing professional annotations of information content units for the BDAE *Cookie Theft* picture.

References

S. Ahmed, C. A. de Jager, A. F. Haigh, and P. Garrard. 2013a. Semantic processing in connected speech at a uniformly early stage of autopsy-confirmed Alzheimer's disease. *Neuropsychology*, 27(1):79–85.

S. Ahmed, A. F. Haigh, C. A. de Jager, and P. Garrard. 2013b. Connected speech as a marker of disease progression in autopsy-proven Alzheimer's disease. *Brain*, 136(12):3727–3737.

C. Ballard, S. Gauthier, A. Corbett, C. Brayne, D. Aarsland, and E. Jones. 2011. Alzheimer's disease. *The Lancet*, 377(9770):1019–1031.

K. A. Bayles and A. W. Kaszniak. 1987. *Communication and cognition in normal aging and dementia*. Little, Brown, Boston.

J. T. Becker, F. Boller, O. L. Lopez, J. Saxton, and K. L. McGonigle. 1994. The natural history of Alzheimer's disease. *Archives of Neurology*, 51:585–594.

B. Croisile, B. Ska, M. J. Brabant, A. Duchene, Y. Lepage, G. Aimard, and M. Trillet. 1996. Comparative study of oral and written picture description in patients with Alzheimer's disease. *Brain and Language*, 53(1):1–19.

K. E. Forbes-McKay and A. Venneri. 2005. Detecting subtle spontaneous language decline in early Alzheimer's disease with a picture description task. *Neurological Sciences*, 26(4):243–254.

K. C. Fraser and G. Hirst. 2016. Detecting semantic changes in Alzheimer's disease with vector space models. In Dimitrios Kokkinakis, editor, *Proceedings of LREC 2016 Workshop: Resources and Processing of Linguistic and Extra-Linguistic Data from People with Various Forms of Cognitive/Psychiatric Impairments (RaPID-2016)*, pages 1–8, Portorož, Slovenia. Linköping University Electronic Press.

K. C. Fraser, J. A. Meltzer, and F. Rudzicz. 2016. Linguistic features identify Alzheimer's disease in narrative speech. *Journal of Alzheimer's Disease*, 49(2):407–422.

E. Giles, K. Patterson, and J. R. Hodges. 1996. Performance on the Boston Cookie Theft picture description task in patients with early dementia of the Alzheimer's type: missing information. *Aphasiology*, 10(4):395–408.

H. Goodglass and E. Kaplan. 1983. *The assessment of aphasia and related disorders*. Lea and Febiger, Philadelphia.

D. Hakkani-Tür, D. Vergyri, and G. Tur. 2010. Speech-based automated cognitive status assessment. In *11th Annual Conference of the International Speech Communication Association*, pages 258–261.

D. B. Hier, K. Hagenlocker, and A. G. Shindler. 1985. Language disintegration in dementia: effects of etiology and severity. *Brain and Language*, 25(1):117–133.

A. Kertesz. 1982. *The Western aphasia battery*. Grune and Stratton, New York.

Y. H. Lai, H. H. Pai, and Y. T. Lin. 2009. To be semantically-impaired or to be syntactically-impaired: linguistic patterns in Chinese-speaking persons with or without dementia. *Journal of Neurolinguistics*, 22(5):465–475.

X. Le, I. Lancashire, G. Hirst, and R. Jokel. 2011. Longitudinal detection of dementia through lexical and syntactic changes in writing: a case study of three British novelists. *Literary and Linguistic Computing*, 26(4):435–461, may.

B. MacWhinney. 2015. *The CHILDES Project: Tools for analyzing talk*. Lawrence Erlbaum Associates, Mahwah, NJ, 3rd edition.

S. V. S. Pakhomov, G. E. Smith, D. Chacon, Y. Feliciano, N. Graff-Radford, R. Caselli, and D. S. Knopman. 2010. Computerized analysis of speech and language to identify psycholinguistic correlates of frontotemporal lobar degeneration. *Cognitive and Behavioral Neurology*, 23(3):165–177.

J. Pennington, R. Socher, and C. D. Manning. 2014. GloVe: Global vectors for word representation. In *Conference on Empirical Methods on Natural Language Processing (EMNLP)*, pages 1532–1543.

D. A. Snowdon, S. J. Kemper, J. A. Mortimer, L. H. Greiner, D. R. Wekstein, and W. R. Markesbery. 1996. Linguistic ability in early life and cognitive function and Alzheimer's disease in late life. Findings from the Nun Study. *JAMA: the Journal of the American Medical Association*, 275(7):528–532.

R. A. Sperling, P. S. Aisen, L. A. Beckett, D. A. Bennett, S. Craft, A. M. Fagan, T. Iwatsubo, C. R. Jack, J. Kaye, T. J. Montine, D. C. Park, E. M. Reiman, C. C. Rowe, E. Siemers, Y. Stern, K. Yaffe, M. C. Carrillo, B. Thies, M. Morrison-Bogorad, M. V. Wagster, and C. H. Phelps. 2011. Toward defining the preclinical stages of Alzheimer's disease: recommendations from the National Institute on Aging-Alzheimer's Association workgroups on diagnostic guidelines for Alzheimer's disease. *Alzheimer's & Dementia: the Journal of the Alzheimer's Association*, 7(3):280–292.

Chinese Couplet Generation with Neural Network Structures

Rui Yan[1,2], **Cheng-Te Li**[3], **Xiaohua Hu**[4], **and Ming Zhang**[5]
[1]Institute of Computer Science and Technology, Peking University, Beijing 100871, China
[2]Natural Language Processing Department, Baidu Inc., Beijing 100193, China
[3]Academia Sinica, Taipei 11529, Taiwan
[4]College of Computing and Informatics, Drexel University, Philadelphia, PA 19104, USA
[5]Department of Computer Science, Peking University, Beijing 100871, China
yanrui02@baidu.com, ctli@citi.sinica.edu.tw
xh29@drexel.edu, mzhang_cs@pku.edu.cn

Abstract

Part of the unique cultural heritage of China is the Chinese couplet. Given a sentence (namely an *antecedent clause*), people reply with another sentence (namely a *subsequent clause*) equal in length. Moreover, a special phenomenon is that corresponding characters from the same position in the two clauses match each other by following certain constraints on semantic and/or syntactic relatedness. Automatic couplet generation by computer is viewed as a difficult problem and has not been fully explored. In this paper, we formulate the task as a natural language generation problem using neural network structures. Given the issued antecedent clause, the system generates the subsequent clause via sequential language modeling. To satisfy special characteristics of couplets, we incorporate the attention mechanism and polishing schema into the encoding-decoding process. The couplet is generated incrementally and iteratively. A comprehensive evaluation, using perplexity and BLEU measurements as well as human judgments, has demonstrated the effectiveness of our proposed approach.

1 Introduction

Chinese antithetical couplets, (namely "对联"), form a special type of poetry composed of two clauses (i.e., sentences). The popularity of the game of Chinese couplet challenge manifests itself in many aspects of people's life, e.g., as a means of expressing personal emotion, political views, or communicating messages at festive occasions. Hence, Chinese couplets are considered an important cultural heritage. A couplet is often written in calligraphy on red banners during special occasions such as wedding ceremonies and the Chinese New Year. People also use couplets to celebrate birthdays, mark the openings of a business, and commemorate historical events. We illustrate a real couplet for Chinese New Year celebration in Figure 1, and translate the couplet into English character-by-character.

Usually in the couplet generation game, one person challenges the other person with a sentence (namely an *antecedent clause*). The other person then replies with another sentence (namely a *subsequent clause*) equal in length and term segmentation, in a way that corresponding characters from the same position in the two clauses match each other by obeying certain constraints on semantic and/or syntactic relatedness. We also illustrate the special phenomenon of Chinese couplet in Figure 1: "one" is paired with "two", "term" is associated with "character", "hundred" is mapped into "thousand", and "happiness" is coupled with "treasures". As opposed to free languages, couplets have unique poetic elegance, e.g., aestheticism and conciseness etc. Filling in the couplet is considered as a challenging task with a set of structural and semantic requirements. Only few best scholars are able to master the skill to manipulate and to organize terms.

The Chinese couplet generation given the antecedent clause can be viewed as a big challenge in the joint area of Artificial Intelligence and Natural Language Processing. With the fast development of computing techniques, we realize that computers might play an important role in helping people to create couplets: 1) it is rather convenient for computers to sort out appropriate term combinations from a large corpus, and 2) computer programs can take great advantages to recognize, to learn, and even to remember patterns or rules given the corpus. Although computers are no sub-

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2347–2357,
Berlin, Germany, August 7-12, 2016.

Figure 1: An example of a Chinese couplet for Chinese New Year. We mark the character-wise translation under each Chinese character of the couplet so as to illustrate that each character from the same position of the two clauses has the constraint of certain relatedness. Overall, the couplet can be translated as: the term of "peaceful and lucky" (i.e., 和顺) indicates countless happiness; the two characters "safe and sound" (a.k.a., 平 and 安) worth innumerable treasures.

stitute for human creativity, they can process very large text repositories of couplets. Furthermore, it is relatively straightforward for the machine to check whether a generated couplet conforms to constraint requirements. The above observations motivate automatic couplet generation using computational intelligence. Beyond the long-term goal of building an autonomous intelligent system capable of creating meaningful couplets eventually, there are potential short-term applications for augmented human expertise/experience to create couplets for entertainment or educational purposes.

To design the automatic couplet generator, we first need to empirically study the generation criteria. We discuss some of the general generation standards here. For example, the couplet generally have rigid formats with the same length for both clauses. Such a syntactic constraint is strict: both clauses have exactly the same length while the length is measured in Chinese characters. Each character from the same position of the two clauses have certain constraints. This constraint is less strict. Since Chinese language is flexible sometimes, synonyms and antonyms both indicate semantic relatedness. Also, semantic coherence is a critical feature in couplets. A well-written couplet is supposed to be semantically coherent among both clauses.

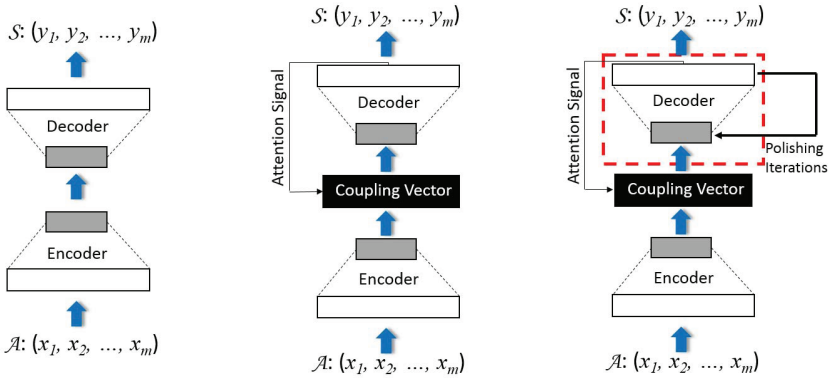
In this paper we are concerned with automatic couplet generation. We propose a neural couplet machine (NCM) based on neural network structures. Given a large collection of texts, we learn representations of individual characters, and their combinations within clauses as well as how they mutually reinforce and constrain each other. Given any specified antecedent clause, the system could generate a subsequent clause via sequential language modeling using encoding and decoding. To satisfy special characteristics of couplets, we incorporate the attention mechanism and polishing schema into the generation process. The couplet is generated incrementally and iteratively to refine wordings. Unlike the single-pass generation process, the hidden representations of the draft subsequent clause will be fed into the neural network structure to *polish* the next version of clause in our proposed system. In contrast to previous approaches, our generator makes utilizations of neighboring characters within the clause through an iterative polishing schema, which is novel.

To sum up, our contributions are as follows. For the first time, we propose a series of neural network-based couplet generation models. We formulate a new system framework to take in the antecedent clauses and to output the subsequent clauses in the couplet pairs. We tackle the special characteristics of couplets, such as corresponding characters paired in the two clauses, by incorporating the attention mechanism into the generation process. For the 1st time, we propose a novel polishing schema to iteratively refine the generated couplet using local pattern of neighboring characters. The draft subsequent clause from the last iteration will be used as additional information to generate a revised version of the subsequent clause.

The rest of the paper is organized as follows. In Section 2, we briefly summarize related work of couplet generation. Then Sections 3 and 4 show the overview of our approach paradigm and then detail the neural models. The experimental results and evaluation are reported in Section 5 and we draw conclusions Section 6.

2 Related Work

There are very few studies focused on Chinese couplet generation, based on templates (Zhang and Sun, 2009) or statistic translations (Jiang and Zhou, 2008). The Chinese couplet generation task can be viewed as a reduced form of 2-sentence poem generation (Jiang and Zhou, 2008). Given the first line of the poem, the generator ought to generate the second line accordingly, which is a similar process as couplet generation. We consider automatic Chinese poetry generation to be a closely re-



(a). Sequential couplet generation. (b). Couplet generation with attention (c). Couplet generation with polishing schema.

Figure 2: Three neural models for couplet generation. More details will be introduced in Section 4.

lated research area. Note that there are still some differences between couplet generation and poetry generation. The task of generating the subsequent clause to match the given antecedent clause is more well-defined than generating all sentences of a poem. Moreover, not all of the sentences in the poems need to follow couplet constraints.

There are some formal researches into the area of computer-assisted poetry generation. Scientists from different countries have studied the automatic poem composition in their own languages through different ways: 1) Genetic Algorithms. Manurung *et al.* (2004; 2011) propose to create poetic texts in English based on state search; 2) Statistical Machine Translation (SMT). Greene *et al.* (2010) propose a translation model to generation cross-lingual poetry, from Italian to English; 3) Rule-based Templates. Oliveira (2009; 2012) has proposed a system of poem generation platform based on semantic and grammar templates in Spanish. An interactive system has been proposed to reproduce the traditional Japanese poem named *Haiku* based on rule-based phrase search related to user queries (Tosa et al., 2008; Wu et al., 2009). Netzer *et al.* (2009) propose another way of Haiku generation using word association rules.

As to computer-assisted Chinese poetry generation. There are now several Chinese poetry generators available. The system named *Daoxiang*[1] basically relies on manual pattern selection. The system maintains a list of manually created terms related to pre-defined keywords, and inserts terms *randomly* into the selected template as a poem. The system is simple but random term selection leads to unnatural sentences.

[1] http://www.poeming.com/web/index.htm

Zhou *et al.* (2010) use a genetic algorithm for Chinese poetry generation by tonal codings and state search. He *et al.* (2012) extend the couplet machine translation paradigm (Jiang and Zhou, 2008) from a 2-line couplet to a 4-line poem by giving previous sentences sequentially, considering structural templates. Yan *et al.* (2013; 2016) proposed a summarization framework to generate poems. Recently, along with the prosperity of neural networks, a recurrent neural network based language generation is proposed (Zhang and Lapata, 2014): the generation is more or less a translation process. Given previous sentences, the system generates the next sentence of the poem.

We also briefly introduce deep neural networks, which contribute great improvements in NLP. A series of neural models are proposed, such as convolutional neural networks (CNN) (Kalchbrenner et al., 2014) and recurrent neural networks (RNN) (Mikolov et al., 2010) with or without gated recurrent units (GRU) (Cho et al., 2014) and long-short term memory (LSTM) units (Hochreiter and Schmidhuber, 1997). We conduct a pilot study to design neural network structures for couplet generation problems. For the first time, we propose a polishing schema for the couplet generation process, and combine it with the attention mechanism to satisfy the couplet constraints, which is novel.

3 Overview

The basic idea of the Chinese couplet generation is to build a hidden representation of the antecedent clause, and then generate the subsequent clause accordingly, shown in Figure 2. In this way, our system works in an encoding-decoding manner. The units of couplet generation are characters.

Problem formulation. We define the following

formulations:

• *Input.* Given the antecedent clause $\mathcal{A} = \{x_1, x_2, \ldots, x_m\}$, $x_i \in \mathcal{V}$, where x_i is a character and $\mathcal{V}$ is the vocabulary, we then learn an abstractive representation of the antecedent clause $\mathcal{A}$.

• *Output.* We generate a subsequent clause $\mathcal{S} = \{y_1, y_2, \ldots, y_m\}$ according to $\mathcal{A}$, which indicates semantic coherence. We have $y_i \in \mathcal{V}$. To be more specific, each character y_i in $\mathcal{S}$ is coordinated with the corresponding character x_i in $\mathcal{A}$, which is determined by the couplet constraint.

As mentioned, we encode the input clause as a hidden vector, and then decode the vector into an output clause so that the two clauses are actually a pair of couplets. Since we have special characteristics for couplet generation, we propose different neural models for different concerns. The proposed models are extended incrementally so that the final model would be able to tackle complicated issues for couplet generation. We first introduce these neural models from a high level description, and then elaborate them in more details.

Sequential Couplet Generation. The model accepts the input clause. We use a recurrent neural network (RNN) over characters to capture the meaning of the clause. Thus we obtain a single vector which represents the antecedent clause. We then use another RNN to decode the input vector into the subsequent clause by the character-wise generation. Basically, the process is a sequence-to-sequence generation via encoding and decoding, which is based on the global level of the clause. We show the diagram of sequential couplet generation in Figure 2(a).

Couplet Generation with Attention. There is a special phenomenon within a pair of couplets: the characters from the same position in the antecedent clause and subsequent clause, i.e., x_i and y_i, generally have some sort of relationships such as "coupling" or "pairing". Hence we ought to model such one-to-one correlation between x_i and y_i in the neural model for couplet generation. Recently, the attention mechanism is proposed to allow the decoder to dynamically select and linearly combine different parts of the input sequence with different weights. Basically, the attention mechanism models the alignment between positions between inputs and outputs, so it can be viewed as a local matching model. Moreover, the tonal coding issue can also be addressed by the pairwise attention mechanism. The extension of attention mechanism to the sequential couplet generation model is shown in Figure 2(b).

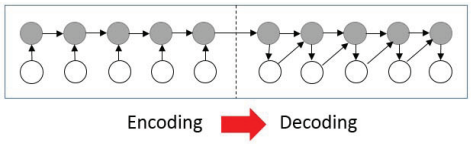


Figure 3: Couplet generation via sequential language modeling: plain neural couplet machine.

Polishing Schema for Generation. Couplet generation is a form of art, and art usually requires polishing. Unlike the traditional single-pass generation in previous neural models, our proposed couplet generator will be able to polish the generated couplets for one or more iterations to refine the wordings. The model is essentially the same as the sequential generation with attention except that the information representation of the previous generated clause draft will be again utilized as an input, serving as *additional* information for semantic coherence. The principle is illustrated in Figure 2(c): the generated draft from the previous iteration will be incorporated into the hidden state which generates the polished couplet pair in the next iteration.

To sum up, we introduce three neural models for Chinese couplet generation. Each revised model targets at tackling an issue for couplet generation so that the system could try to imitate a human couplet generator. We further elaborate these neural models incrementally in details in Section 4.

4 Neural Generation Models

4.1 Sequential Couplet Generation

The sequential couplet generation model is basically a sequence-to-sequence generation fashion (Sutskever et al., 2014) using encoding and decoding shown in Figure 3. We use a recurrent neural network (RNN) to iteratively pick up information over the character sequence $x_1, x_2, \ldots, x_m$ of the input antecedent clause $\mathcal{A}$. All characters are vectorized using their embeddings (Mikolov et al., 2013). For each character, the RNN allocates a hidden state $\boldsymbol{s_i}$, which is dependent on the current character's embedding $\boldsymbol{x}_i$ and the previous state $\boldsymbol{s}_{i-1}$. Since usually each clause in the couplet pair would not be quite long, it is sufficient to use a vanilla RNN with basic interactions.

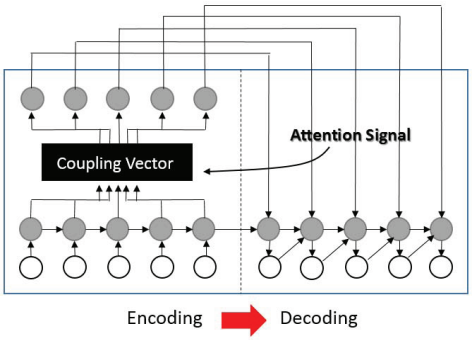


Figure 4: Couplet generation with attention mechanism, namely attention neural couplet machine. Attention signal is generated by both encoder and decoder, and then fed into the coupling vector. Calculation details are elaborated in Section 4.2.

The equation for encoding is as follows:

$$\boldsymbol{s}_i = f(W_h \boldsymbol{s}_{i-1} + W_x \boldsymbol{x}_i + b) \quad (1)$$

$\boldsymbol{x}$ is the vector representation (i.e., embedding) of the character. W and b are parameters for weights and bias. $f(\cdot)$ is the non-linear activation function and we use ReLU (Nair and Hinton, 2010) in this paper. As for the hidden state $\boldsymbol{h}_i$ in the decoding RNN, we have:

$$\boldsymbol{h}_i = f(W_x \boldsymbol{x}_{i-1} + W_h \boldsymbol{h}_{i-1}) \quad (2)$$

4.2 Couplet Generation with Attention

As mentioned, there is special phenomenon in the couplet pair that the characters from the same position in the antecedent clause and the subsequent clause comply with certain relatedness, so that two clauses may, to some extent, look "symmetric".

Hence we introduce the attention mechanism into the couplet generation model. The attention mechanism coordinates, either statically or dynamically, different positions of the input sequence (Shang et al., 2015). To this end, we introduce a hidden coupling vector $\boldsymbol{c}_i = \sum_{j=1}^{m} \alpha_{ij} \boldsymbol{s}_j$. The coupling vectors linearly combine all parts from the antecedent clause, and determine which part should be utilized to generate the characters in the subsequent clause. The attention signal α_{ij} can be calculated as $\alpha_{ij} = \sigma_{att}(\boldsymbol{s}_j, \boldsymbol{h}_{i-1})$ after a softmax function. The score is based on how well the inputs from position j and the output at position i match. $\sigma_{att}(\cdot)$ is parametrized as a neural network which is jointly trained with all the other components (Bahdanau et al., 2015; Hermann et al., 2015). This mechanism enjoys the advantage of adaptively focusing on the corresponding characters of the input text according to the generated characters in the subsequent clause. The mechanism is pictorially shown in Figure 4.

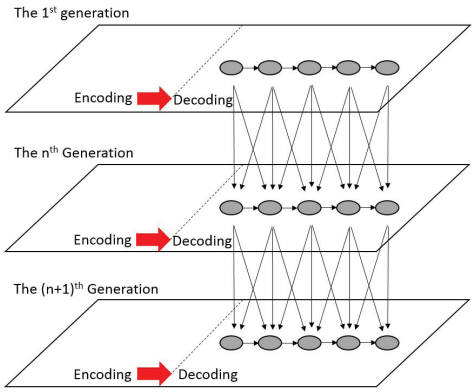


Figure 5: Couplet generation with the polishing schema, i.e., full neural couplet machine. Note that for conciseness, we only show the gist of this schema across polishing iterations. The shaded circles are the hidden vectors to generate characters in the subsequent clause. We omit the duplicated sequential and attention dependencies within each iteration as we have shown in Figures 3 & 4.

With the coupling vectors generated, we have the following equation for the decoding process with attention mechanism:

$$\boldsymbol{h}_i = f(W_x \boldsymbol{x}_{i-1} + W_h \boldsymbol{h}_{i-1} + W_c \boldsymbol{c}_i) \quad (3)$$

4.3 Polishing Schema for Generation

Inspired by the observation that a human couplet generator might recompose the clause for several times, we propose a polishing schema for the couplet generation. Specifically, after a single-pass generation, the couplet generator itself shall be aware of the generated clause as a *draft*, so that polishing each and every character of the clause becomes possible.

We hereby propose a convolutionary neural network (CNN) based polishing schema shown in Figure 5. The intuition for convolutionary structure is that this polishing schema guarantees better coherence: with the batch of neighboring characters, the couplet generator knows which character to generate during the revision process.

A convolutional neural network applies a fixed-size window to extract local (neighboring) patterns of successive characters. Suppose the window is of size t, the detected features at a certain position

$x_i, \cdots, x_{i+t-1}$ is given by

$$\boldsymbol{o}_i^{(n)} = f(W[\boldsymbol{h}_i^{(n)}; \cdots; \boldsymbol{h}_{i+t-1}^{(n)}] + \boldsymbol{b}) \quad (4)$$

Here $\boldsymbol{h}^{(n)}$ with the superscript n is the hidden vector representation from the n-th iteration. W and $\boldsymbol{b}$ are parameters for convolution. Semicolons refer to column vector concatenation. Also, $f(\cdot)$ is the non-linear activation function and we use ReLU (Nair and Hinton, 2010) as well. Note that we pad zero at the end of the term if a character does not have enough following characters to fill the slots in the convolution window. In this way, we obtain a set of detected features. Then a max-pooling layer aggregates information over different characters into a fixed-size vector.

Now the couplet generation with both attention mechanism and polishing schema becomes:

$$\begin{aligned}\boldsymbol{h}_i^{(n+1)} = f(&W_x \boldsymbol{x}_{i-1} + W_h \boldsymbol{h}_{i-1}^{(n+1)} \\ &+ W_c \boldsymbol{c}_i^{(n+1)} + W_o \boldsymbol{o}_i^{(n)})\end{aligned} \quad (5)$$

Note that in this way ,we feed the information from the n-th generation iteration into the (n+1)-th polishing iteration. For the iterations, we have the stopping criteria as follows.

• After each iteration process, we have the subsequent clause generated; we encode the clause as $\boldsymbol{h}$ using the RNN encoder using the calculation shown in Equation (1). We stop the algorithm iteration when the cosine similarity between the two $\boldsymbol{h}^{(n+1)}$ and $\boldsymbol{h}^{(n)}$ from two successive iterations exceeds a threshold Δ (Δ = 0.5 in this study).

• Ideally, we shall let the algorithm converge by itself. There will always be some long-tail cases. To be practical, it is necessary to apply a termination schedule when the generator polishes for many times. We stop the couplet generator after a fixed number of recomposition. Here we empirically set the threshold as 5 times of polishing, which means 6 iterations in all.

5 Experiments and Evaluations

5.1 Experimental Setups

Datasets. A large Chinese couplet corpus is necessary to learn the model for couplet generation. There is, however, no large-sized pure couplet collection available (Jiang and Zhou, 2008). As mentioned, generally people regard Chinese couplets as a reduced form of Chinese poetry and there are several large Chinese poem datasets publicly available, such as *Poems of Tang Dynasty* (i.e., Tang Poem) and *Poems of Song Dynasty* (i.e., Song Poem). It becomes a widely acceptable approximation to mine couplets out of existing poems, even though poems are not specifically intended for couplets[2] (Jiang and Zhou, 2008; Yan et al., 2013; He et al., 2012). We are able to mine such sentence pairs out of the poems and filtering those do not conform to couplet constraints, which is a similar process mentioned in (Jiang and Zhou, 2008). Moreover, we also crawl couplets from *couplet forums* where couplet fans discuss, practice and show couplet works. We performed standard Chinese segmentation into characters.

Table 1: Detailed information of the datasets. Each pair of couplets consist of two clauses.

	#Pairs	#Character
TANG Poem	26,833	6,358
SONG Poem	11,324	3,629
Couplet Forum	46,959	8,826

In all, we collect 85,116 couplets. We randomly choose 2,000 couplets for validation and 1,000 couplets for testing, other non-overlap ones for training. The details are shown in Table 1.

Hyperparameters and Setups. Word embeddings (Mikolov et al., 2013) are a standard apparatus in neural network-based text processing. A word is mapped to a low dimensional, real-valued vector. This process, known as vectorization, captures some underlying meanings. Given enough data, usage, and context, word embeddings can make highly accurate guesses about the meaning of a particular word. Embeddings can equivalently be viewed that a word is first represented as a one-hot vector and multiplied by a look-up table (Mikolov et al., 2013). In our model, we first vectorize all words using their embeddings. Here we used 128-dimensional word embeddings through vectorization, and they were initialized randomly and learned during training. We set the width of convolution filters as 3. The above parameters were chosen empirically.

Training. The objective for training is the cross entropy errors of the predicted character distribution and the actual character distribution in our

[2]For instance, in the 4-sentence poetry (namely *quatrain*, i.e., 绝句 in Chinese), the 3rd and 4th sentences are usually paired; in the 8-sentence poetry (namely *regulated verse*, i.e., 律诗 in Chinese), the 3rd-4th and 5th-6th sentences are generally form pairs which satisfy couplet constraints.

corpus. An ℓ_2 regularization term is also added to the objective. The model is trained with back propagation through time with the length being the time step. The objective is minimized by stochastic gradient descent with shuffled mini-batches (with a mini-batch size of 100) for optimization. During training, the cross entropy error of the output is back-propagated through all hidden layers. Initial learning rate was set to 0.8, and a multiplicative learning rate decay was applied. We used the validation set for early stopping. In practice, the training converges after a few epochs.

5.2 Evaluation Metrics

It is generally difficult to judge the effect of couplets generated by computers. We propose to evaluate results from 3 different evaluation metrics.

Perplexity. For most of the language generation research, language perplexity is a sanity check. Our first set of experiments involved intrinsic evaluation of the "perplexity" evaluation for the generated couplets. Perplexity is actually an entropy based evaluation. In this sense, the lower perplexity for the couplets generated, the better performance in purity for the generations, and the couplets are likely to be good. m denotes the length.

$$\text{pow}\left[2, -\frac{1}{m}\sum_{i=1}^{m}\log p(y_i)\right]$$

BLEU. The Bilingual Evaluation Understudy (BLEU) score-based evaluation is usually used for machine translation (Papineni et al., 2002): given the reference translation(s), the algorithm evaluates the quality of text which has been machine-translated from the reference translation as ground truth. We adapt the BLEU evaluation under the couplet generation scenario. Take a couplet from the dataset, we generate the computer authored subsequent clause given the antecedent clause, and compare it with the original subsequent clause written by humans. There is a concern for such an evaluation metric is that BLEU score can only reflect the partial capability of the models; there is (for most cases) only one ground truth for the generated couplet but actually there are more than one appropriate ways to generate a well-written couplet. The merit of BLEU evaluation is to examine how likely to approximate the computer generated couplet towards human authored ones.

Human Evaluation. We also include human judgments from 13 evaluators who are graduate students majoring in Chinese literature. Evaluators are requested to express an opinion over the automatically generated couplets. A clear criterion is necessary for human evaluation. We use the evaluation standards discussed in (Wang, 2002; Jiang and Zhou, 2008; He et al., 2012; Yan et al., 2013; Zhang and Lapata, 2014): "syntactic" and "semantic" satisfaction. For the syntactic side, evaluators consider whether the subsequent clauses conform the length restriction and word pairing between the two clauses. For a higher level of semantic side, evaluators then consider whether the two clauses are semantically meaningful and coherent. Evaluators assign 0-1 scores for both syntactic and semantic criteria ('0'-no, '1'- yes). The evaluation process is conducted as a blind-review[3]

5.3 Algorithms for Comparisons

We implemented several generation methods as baselines. For fairness, we conduct the same pre-generation process to all algorithms.

Standard SMT. We adapt the standard phrase-based statistical machine translation method (Koehn et al., 2003) for the couplet task, which regards the antecedent clause as the source language and the subsequent clause as the target language.

Couplet SMT. Based on SMT techniques, a phrase-based SMT system for Chinese couplet generation is proposed in (Jiang and Zhou, 2008), which incorporates extensive couplet-specific character filtering and re-rankings.

LSTM-RNN. We also include a sequence-to-sequence LSTM-RNN (Sutskever et al., 2014). LSTM-RNN is basically a RNN using the LSTM units, which consists of memory cells in order to store information for extended periods of time. For generation, we first use an LSTM-RNN to encode the given antecedent sequence to a vector space, and then use another LSTM-RNN to decode the vector into the output sequence.

Since Chinese couplet generation can be viewed as a reduced form of Chinese poetry generation, we also include some approaches designed for poetry generation as baselines.

iPoet. Given the antecedent clause, the *iPoet* method first retrieves relevant couplets from the

[3]We understand that acceptability is a gradable concept, especially for the less subjective tasks. Here from our experience, to grade the "yes"-"no" acceptability is more feasible for the human evaluators to judge (with good agreement). As to couplet evaluation, it might be more difficult for the evaluators to say "very acceptable" or "less acceptable". We will try to make scale-based evaluation as the future work.

Table 2: Overall performance comparison against baselines.

Algorithm	Perplexity	BLEU	Human Evaluation		
			Syntactic	Semantic	Overall
Standard SMT (Koehn et al., 2003)	128	21.68	0.563	0.248	0.811
Couplet SMT (Jiang and Zhou, 2008)	97	28.71	0.916	0.503	1.419
LSTM-RNN (Sutskever et al., 2014)	85	24.23	0.648	0.233	0.881
iPoet (Yan et al., 2013)	143	13.77	0.228	0.435	0.663
Poetry SMT (He et al., 2012)	121	23.11	0.802	0.516	1.318
RNNPG (Zhang and Lapata, 2014)	99	25.83	0.853	0.600	1.453
Neural Couplet Machine (NCM)	**68**	**32.62**	**0.925**	**0.631**	**1.556**

corpus, and then summarizes the retrieved couplets into a single clause based on a generative summarization framework (Yan et al., 2013).

Poetry SMT. He et al. (2012) extend the couplet SMT method into a poetry-oriented SMT approach, with different focus and different filtering for different applications from Couplet SMT.

RNNPG. The RNN-based poem generator (RNNPG) is proposed to generate a poem (Zhang and Lapata, 2014), and we adapt it into the couplet generation scenario. Given the antecedent clause, the subsequent clause is generated through the standard RNN process with contextual convolutions of the given antecedent clause.

Neural Couplet Machine (NCM). We propose the neural generation model particularly for couplets. Basically we have the RNN based encoding-decoding process with attention mechanism and polishing schema. We demonstrate with the best performance of all NCM variants proposed here.

5.4 Performance

In Table 2 we show the overall performance of our proposed NCM system compared with strong competing methods as described above. We see that, for perplexity, BLEU and human judgments, our system outperforms other baseline models.

The standard SMT method manipulates characters according to the dataset by standard translation but ignores all couplet characteristics in the model. The Couplet SMT especially established for couplet generation performs much better than the general SMT method since it incorporates several filtering with couplet constraints. As a strongly competitive baseline of the neural model LSTM-RNN, the perplexity performance gets boosted in the generation process, which indicates that neural models show strong ability for language generation. However, there is a major drawback that LSTM-RNN does not explicitly model the couplet constraints such as length restrictions and so on for couplet pairs. LSTM-RNN is not really a couplet-driven generation method and might not capture the corresponding patterns between the antecedent clause and subsequent clause well enough to get a high BLEU score.

For the group of algorithms originally proposed for poetry generation, we have summarization-based poetry method *iPoet*, translation-based poetry method *Poetry SMT* and a neural network based method *RNNPG*. In general, the summarization based poetry method iPoet does not perform well in either perplexity or BLEU evaluation: summarization is not an intuitive way to model and capture the pairwise relationship between the antecedent and subsequent clause within the couplet pair. Poetry SMT performs better, indicating the translation-based solution makes more sense for couplet generation than summarization methods. RNNPG is a strong baseline which applies both neural network structures, while the insufficiency lies in the lack of couplet-oriented constraints during the generation process. Note that all poetry-oriented methods show worse performance than the couplet SMT method, indicating that couplet constraints should be specially addressed.

We hence introduce the neural couplet machine based on neural network structures specially designed for couplet generation. We incorporate attention mechanism and polishing schema into the generation process. The attention mechanism strengthens the coupling characteristics between the antecedent and subsequent clause and the polishing schema enables the system to revise and refine the generated couplets, which leads to better performance in experimental evaluations.

For evaluations, the perplexity scores and BLEU scores show some consistency. Besides, we

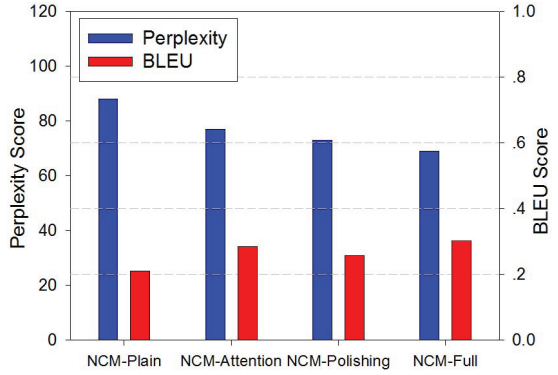


Figure 6: Performance comparison of all variants in the neural couplet machine family.

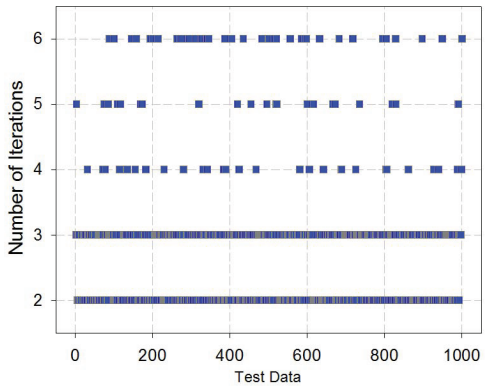


Figure 7: The distribution of stopping iteration counts for all test data. Note that 6 iterations of generation means 5 times of polishing.

observe that the BLEU scores are quite low for almost all methods. It is not surprising that these methods are not likely to generate the exactly same couplets as the ground truth, since that is not how the objective function works. BLEU can only partially calibrate the capability of couplet generation because there are many ways to create couplets which do not look like the ground truth but also make sense to people. Although quite subjective, the *human evaluations* in Table 2 can to some extent show the potentials of all couplet generators.

5.5 Analysis and Discussions

There are two special strategies in the proposed neural model for couplet generation: 1) attention mechanism and 2) polishing schema. We hence analyze the separate contributions of the two components in all the neural couplet machine variants. We have the *NCM-Plain* model with no attention or polishing strategy. We incrementally add the attention mechanism as *NCM-Attention*, and then add the polishing schema as *NCM-Full*. The three NCM variants correspond to the three models proposed in this paper. Besides, for a complete comparison, we also include the plain NCM integrated with polishing schema but without attention mechanism, namely *NCM-Polishing*.

The results are shown in Figure 6. We can see that NCM-Plain shows the weakest performance, with no strategy tailored for couplet generation. An interesting phenomenon is that *NCM-Attention* has better performance in BLEU score while *NCM-Polishing* performs better in terms of perplexity. We conclude that attention mechanism captures the pairing patterns between the two clauses, and the polishing schema enables better wordings of semantic coherence in the couplet after several revisions. The two strategies address different concerns for couplet generation, hence *NCM-Full* performs best.

We also take a closer look at the polishing schema proposed in this paper, which enables a multi-pass generation. The couplet generator can generate a subsequent clause utilizing additional information from the generated subsequent clause from the last iteration. It is a novel insight against previous methods. The effect and benefits of the polishing schema is demonstrated in Figure 6. We also examine the stopping criteria, shown in Figure 7. In general, most of the polishing process stops after 2-3 iterations.

6 Conclusions

The Chinese couplet generation is a difficult task in the field of natural language generation. We propose a novel neural couplet machine to tackle this problem based on neural network structures. Given an antecedent clause, we generate a subsequent clause to create a couplet pair using a sequential generation process. The two innovative insights are that 1) we adapt the attention mechanism for the couplet coupling constraint, and 2) we propose a novel polishing schema to refine the generated couplets using additional information.

We compare our approach with several baselines. We apply perplexity and BLEU to evaluate the performance of couplet generation as well as human judgments. We demonstrate that the neural couplet machine can generate rather good couplets and outperform baselines. Besides, both attention mechanism and polishing schema contribute to the better performance of the proposed approach.

Acknowledgments

We thank all the anonymous reviewers for their valuable and constructive comments. This paper is partially supported by the National Natural Science Foundation of China (NSFC Grant Numbers 61272343, 61472006), the Doctoral Program of Higher Education of China (Grant No. 20130001110032) as well as the National Basic Research Program (973 Program No. 2014CB340405).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Erica Greene, Tugba Bodrumlu, and Kevin Knight. 2010. Automatic analysis of rhythmic poetry with applications to generation and translation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP'10, pages 524–533.

Jing He, Ming Zhou, and Long Jiang. 2012. Generating chinese classical poems with statistical machine translation models. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 1650–1656.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1684–1692.

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

Long Jiang and Ming Zhou. 2008. Generating chinese couplets using a statistical mt approach. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 377–384.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

R. Manurung, G. Ritchie, and H. Thompson. 2011. Using genetic algorithms to create meaningful poetic text. *Journal of Experimental & Theoretical Artificial Intelligence*, 24(1):43–64.

H. Manurung. 2004. An evolutionary algorithm approach to poetry generation. *University of Edinburgh. College of Science and Engineering. School of Informatics.*

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*, volume 2, page 3.

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

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Yael Netzer, David Gabay, Yoav Goldberg, and Michael Elhadad. 2009. Gaiku: generating haiku with word associations norms. In *Proceedings of the Workshop on Computational Approaches to Linguistic Creativity*, CALC '09, pages 32–39.

H. Oliveira. 2009. Automatic generation of poetry: an overview. *Universidade de Coimbra*.

H.G. Oliveira. 2012. Poetryme: a versatile platform for poetry generation. *Computational Creativity, Concept Invention, and General Intelligence*, 1:21.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th Interational Joint Conference on Natural Language Processing*, ACL-IJCNLP'15, pages 1577–1586.

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

N. Tosa, H. Obara, and M. Minoh. 2008. Hitch haiku: An interactive supporting system for composing haiku poem. *Entertainment Computing-ICEC 2008*, pages 209–216.

Li Wang. 2002. A summary of rhyming constraints of chinese poems. Beijing Press.

X. Wu, N. Tosa, and R. Nakatsu. 2009. New hitch haiku: An interactive renku poem composition supporting tool applied for sightseeing navigation system. *Entertainment Computing–ICEC 2009*, pages 191–196.

Rui Yan, Han Jiang, Mirella Lapata, Shou-De Lin, Xueqiang Lv, and Xiaoming Li. 2013. i, poet: Automatic chinese poetry composition through a generative summarization framework under constrained optimization. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, IJCAI'13, pages 2197–2203.

Rui Yan. 2016. i, poet: Automatic poetry composition through recurrent neural networks with iterative polishing schema. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, IJCAI'16.

Xingxing Zhang and Mirella Lapata. 2014. Chinese poetry generation with recurrent neural networks. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 670–680.

Kai-Xu Zhang and Mao-Song Sun. 2009. An chinese couplet generation model based on statistics and rules. *Journal of Chinese Information Processing*, 1:017.

Cheng-Le Zhou, Wei You, and Xiaojun Ding. 2010. Genetic algorithm and its implementation of automatic generation of chinese songci. *Journal of Software*, 21(3):427–437.

A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task

Danqi Chen and **Jason Bolton** and **Christopher D. Manning**
Computer Science Stanford University
Stanford, CA 94305-9020, USA
{danqi,jebolton,manning}@cs.stanford.edu

Abstract

Enabling a computer to understand a document so that it can answer comprehension questions is a central, yet unsolved goal of NLP. A key factor impeding its solution by machine learned systems is the limited availability of human-annotated data. Hermann et al. (2015) seek to solve this problem by creating over a million training examples by pairing *CNN* and *Daily Mail* news articles with their summarized bullet points, and show that a neural network can then be trained to give good performance on this task. In this paper, we conduct a thorough examination of this new reading comprehension task. Our primary aim is to understand what depth of language understanding is required to do well on this task. We approach this from one side by doing a careful hand-analysis of a small subset of the problems and from the other by showing that simple, carefully designed systems can obtain accuracies of 72.4% and 75.8% on these two datasets, exceeding current state-of-the-art results by over 5% and approaching what we believe is the ceiling for performance on this task.[1]

1 Introduction

Reading comprehension (RC) is the ability to read text, process it, and understand its meaning.[2] How to endow computers with this capacity has been an elusive challenge and a long-standing goal of Artificial Intelligence (e.g., (Norvig, 1978)). Genuine reading comprehension involves interpretation of the text and making complex inferences. Human reading comprehension is often tested by asking questions that require interpretive understanding of a passage, and the same approach has been suggested for testing computers (Burges, 2013).

In recent years, there have been several strands of work which attempt to collect human-labeled data for this task – in the form of document, question and answer triples – and to learn machine learning models directly from it (Richardson et al., 2013; Berant et al., 2014; Wang et al., 2015). However, these datasets consist of only hundreds of documents, as the labeled examples usually require considerable expertise and neat design, making the annotation process quite expensive. The subsequent scarcity of labeled examples prevents us from training powerful statistical models, such as deep learning models, and would seem to prevent a system from learning complex textual reasoning capacities.

Recently, researchers at *DeepMind* (Hermann et al., 2015) had the appealing, original idea of exploiting the fact that the abundant news articles of *CNN* and *Daily Mail* are accompanied by bullet point summaries in order to heuristically create large-scale supervised training data for the reading comprehension task. Figure 1 gives an example. Their idea is that a bullet point usually summarizes one or several aspects of the article. If the computer understands the content of the article, it should be able to infer the missing entity in the bullet point.

This is a clever way of creating supervised data cheaply and holds promise for making progress on training RC models; however, it is unclear what level of reading comprehension is actually needed to solve this somewhat artificial task and, indeed, what statistical models that do reasonably well on this task have actually learned.

In this paper, our aim is to provide an in-depth and thoughtful analysis of this dataset and what level of natural language understanding is needed to

[1] Our code is available at https://github.com/danqi/rc-cnn-dailymail.

[2] https://en.wikipedia.org/wiki/Reading_comprehension

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2358–2367,
Berlin, Germany, August 7-12, 2016.

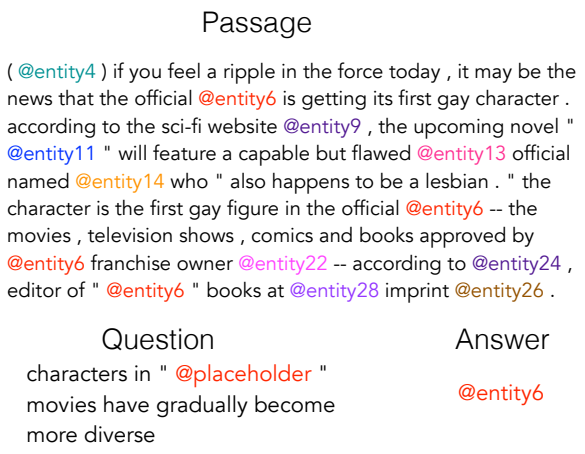


Figure 1: An example item from dataset *CNN*.

do well on it. We demonstrate that simple, carefully designed systems can obtain high, state-of-the-art accuracies of 72.4% and 75.8% on *CNN* and *Daily Mail* respectively. We do a careful hand-analysis of a small subset of the problems to provide data on their difficulty and what kinds of language understanding are needed to be successful and we try to diagnose what is learned by the systems that we have built. We conclude that: (i) this dataset is easier than previously realized, (ii) straightforward, conventional NLP systems can do much better on it than previously suggested, (iii) the distributed representations of deep learning systems are very effective at recognizing paraphrases, (iv) partly because of the nature of the questions, current systems much more have the nature of single-sentence relation extraction systems than larger-discourse-context text understanding systems, (v) the systems that we present here are close to the ceiling of performance for single-sentence and unambiguous cases of this dataset, and (vi) the prospects for getting the final 20% of questions correct appear poor, since most of them involve issues in the data preparation which undermine the chances of answering the question (coreference errors or anonymization of entities making understanding too difficult).

2 The Reading Comprehension Task

The RC datasets introduced in (Hermann et al., 2015) are made from articles on the news websites *CNN* and *Daily Mail*, utilizing articles and their bullet point summaries.[3] Figure 1 demonstrates an example[4]: it consists of a passage p, a question q and an answer a, where the passage is a news article, the question is a cloze-style task, in which one of the article's bullet points has had one entity replaced by a placeholder, and the answer is this questioned entity. The goal is to infer the missing entity (answer a) from all the possible entities which appear in the passage. A news article is usually associated with a few (e.g., 3–5) bullet points and each of them highlights one aspect of its content.

The text has been run through a Google NLP pipeline. It it tokenized, lowercased, and named entity recognition and coreference resolution have been run. For each coreference chain containing at least one named entity, all items in the chain are replaced by an @entity$n$ marker, for a distinct index n. Hermann et al. (2015) argue convincingly that such a strategy is necessary to ensure that systems approach this task by understanding the passage in front of them, rather than by using world knowledge or a language model to answer questions without needing to understand the passage. However, this also gives the task a somewhat artificial character. On the one hand, systems are greatly helped by entity recognition and coreference having already been performed; on the other, they suffer when either of these modules fail, as they do (in Figure 1, "the character" should probably be coreferent with @entity14; clearer examples of failure appear later on in our data analysis). Moreover, this inability to use world knowledge also makes it much more difficult for a human to do this task – occasionally it is very difficult or impossible for a human to determine the correct answer when presented with an item anonymized in this way.

The creation of the datasets benefits from the sheer volume of news articles available online, so they offer a large and realistic testing ground for statistical models. Table 1 provides some statistics on the two datasets: there are 380k and 879k training examples for *CNN* and *Daily Mail* respectively. The passages are around 30 sentences and 800 tokens on average, while each question contains around 12–14 tokens.

In the following sections, we seek to more deeply understand the nature of this dataset. We first build some straightforward systems in order to get a better idea of a lower-bound for the performance of current NLP systems. Then we turn to data analysis of a

[3]The datasets are available at `https://github.com/deepmind/rc-data`.

[4]The original article can be found at `http://www.cnn.com/2015/03/10/entertainment/feat-star-wars-gay-character/`.

	CNN	Daily Mail
# Train	380,298	879,450
# Dev	3,924	64,835
# Test	3,198	53,182
Passage: avg. tokens	761.8	813.1
Passage: avg. sentences	32.3	28.9
Question: avg. tokens	12.5	14.3
Avg. # entities	26.2	26.2

Table 1: Data statistics of the *CNN* and *Daily Mail* datasets. The avg. tokens and sentences in the passage, the avg. tokens in the query, and the number of entities are based on statistics from the training set, but they are similar on the development and test sets.

sample of the items to examine their nature and an upper bound on performance.

3 Our Systems

In this section, we describe two systems we implemented – a conventional entity-centric classifier and an end-to-end neural network. While Hermann et al. (2015) do provide several baselines for performance on the RC task, we suspect that their baselines are not that strong. They attempt to use a frame-semantic parser, and we feel that the poor coverage of that parser undermines the results, and is not representative of what a straightforward NLP system – based on standard approaches to factoid question answering and relation extraction developed over the last 15 years – can achieve. Indeed, their frame-semantic model is markedly inferior to another baseline they provide, a heuristic word distance model. At present just two papers are available presenting results on this RC task, both presenting neural network approaches: (Hermann et al., 2015) and (Hill et al., 2016). While the latter is wrapped in the language of end-to-end memory networks, it actually presents a fairly simple window-based neural network classifier running on the CNN data. Its success again raises questions about the true nature and complexity of the RC task provided by this dataset, which we seek to clarify by building a simple attention-based neural net classifier.

Given the (passage, question, answer) triple (p, q, a), $p = \{p_1, \ldots, p_m\}$ and $q = \{q_1, \ldots, q_l\}$ are sequences of tokens for the passage and question sentence, with q containing exactly one "@placeholder" token. The goal is to infer the correct entity $a \in p \cap E$ that the placeholder corresponds to, where E is the set of all abstract entity markers. Note that the correct answer entity must appear in the passage p.

3.1 Entity-Centric Classifier

We first build a conventional feature-based classifier, aiming to explore what features are effective for this task. This is similar in spirit to (Wang et al., 2015), which at present has very competitive performance on the MCTest RC dataset (Richardson et al., 2013). The setup of this system is to design a feature vector $f_{p,q}(e)$ for each candidate entity e, and to learn a weight vector θ such that the correct answer a is expected to rank higher than all other candidate entities:

$$\theta^\intercal f_{p,q}(a) > \theta^\intercal f_{p,q}(e), \forall e \in E \cap p \setminus \{a\} \quad (1)$$

We employ the following feature templates:

1. Whether entity e occurs in the passage.
2. Whether entity e occurs in the question.
3. The frequency of entity e in the passage.
4. The first position of occurence of entity e in the passage.
5. n-gram exact match: whether there is an exact match between the text surrounding the placeholder and the text surrounding entity e. We have features for all combinations of matching left and/or right one or two words.
6. Word distance: we align the placeholder with each occurrence of entity e, and compute the average minimum distance of each non-stop question word from the entity in the passage.
7. Sentence co-occurrence: whether entity e co-occurs with another entity or verb that appears in the question, in some sentence of the passage.
8. Dependency parse match: we dependency parse both the question and all the sentences in the passage, and extract an indicator feature of whether $w \xrightarrow{r}$ @placeholder and $w \xrightarrow{r} e$ are both found; similar features are constructed for @placeholder $\xrightarrow{r} w$ and $e \xrightarrow{r} w$.

3.2 End-to-end Neural Network

Our neural network system is based on the *Attentive-Reader* model proposed by (Hermann et al., 2015). The framework can be described in the following three steps (see Figure 2):

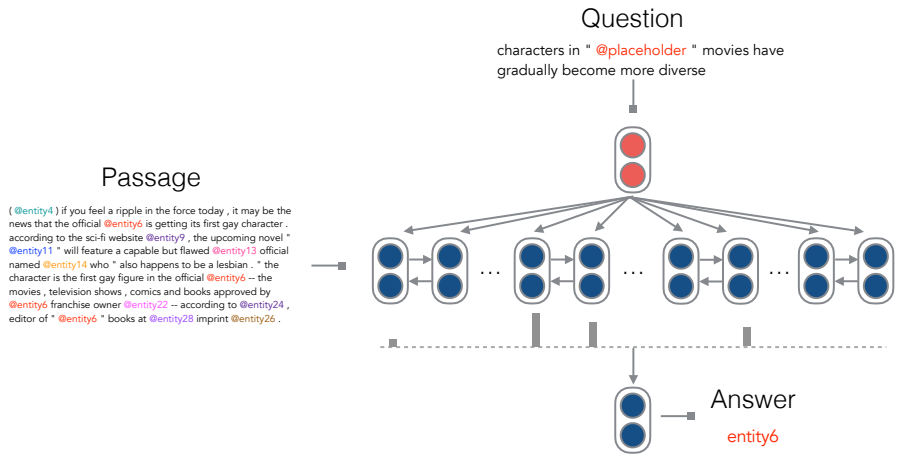


Figure 2: Our neural network architecture for the reading comprehension task.

Encoding: First, all the words are mapped to d-dimensional vectors via an embedding matrix $E \in \mathbb{R}^{d \times |\mathcal{V}|}$; therefore we have p: $\mathbf{p}_1, \ldots, \mathbf{p}_m \in R^d$ and $q : \mathbf{q}_1, \ldots, \mathbf{q}_l \in R^d$.

Next we use a shallow bi-directional LSTM with hidden size $\tilde{h}$ to encode contextual embeddings $\tilde{\mathbf{p}}_i$ of each word in the passage,

$$\overrightarrow{\mathbf{h}}_i = \text{LSTM}(\overrightarrow{\mathbf{h}}_{i-1}, \mathbf{p}_i), i = 1, \ldots, m$$
$$\overleftarrow{\mathbf{h}}_i = \text{LSTM}(\overleftarrow{\mathbf{h}}_{i+1}, \mathbf{p}_i), i = m, \ldots, 1$$

and $\tilde{\mathbf{p}}_i = \text{concat}(\overrightarrow{\mathbf{h}}_i, \overleftarrow{\mathbf{h}}_i) \in \mathbb{R}^h$, where $h = 2\tilde{h}$. Meanwhile, we use another bi-directional LSTM to map the question $\mathbf{q}_1, \ldots, \mathbf{q}_l$ to an embedding $\mathbf{q} \in \mathbb{R}^h$.

Attention: In this step, the goal is to compare the question embedding and all the contextual embeddings, and *select* the pieces of information that are relevant to the question. We compute a probability distribution α depending on the degree of relevance between word p_i (in its context) and the question q and then produce an output vector $\mathbf{o}$ which is a weighted combination of all contextual embeddings $\{\tilde{\mathbf{p}}_i\}$:

$$\alpha_i = \text{softmax}_i \, \mathbf{q}^\intercal \mathbf{W}_s \tilde{\mathbf{p}}_i \quad (2)$$
$$\mathbf{o} = \sum_i \alpha_i \tilde{\mathbf{p}}_i \quad (3)$$

$\mathbf{W_s} \in \mathbb{R}^{h \times h}$ is used in a bilinear term, which allows us to compute a similarity between $\mathbf{q}$ and $\tilde{\mathbf{p}}_i$ more flexibly than with just a dot product.

Prediction: Using the *output* vector $\mathbf{o}$, the system outputs the most likely answer using:

$$a = \arg\max_{a \in p \cap E} W_a^\intercal \mathbf{o} \quad (4)$$

Finally, the system adds a softmax function on top of $W_a^\intercal \mathbf{o}$ and adopts a negative log-likelihood objective for training.

Differences from (Hermann et al., 2015). Our model basically follows the *AttentiveReader*. However, to our surprise, our experiments observed nearly **8–10**% improvement over the original *AttentiveReader* results on *CNN* and *Daily Mail* datasets (discussed in Sec. 4). Concretely, our model has the following differences:

- We use a bilinear term, instead of a tanh layer to compute the relevance (attention) between question and contextual embeddings. The effectiveness of the simple bilinear attention function has been shown previously for neural machine translation by (Luong et al., 2015).
- After obtaining the weighted contextual embeddings $\mathbf{o}$, we use $\mathbf{o}$ for direct prediction. In contrast, the original model in (Hermann et al., 2015) combined $\mathbf{o}$ and the question embedding $\mathbf{q}$ via another non-linear layer before making final predictions. We found that we could remove this layer without harming performance. We believe it is sufficient for the model to learn to return the entity to which it maximally gives attention.
- The original model considers all the words from the vocabulary $\mathcal{V}$ in making predictions. We think this is unnecessary, and only predict among entities which appear in the passage.

Of these changes, only the first seems important; the other two just aim at keeping the model simple.

Window-based MemN2Ns (Hill et al., 2016). Another recent neural network approach proposed by (Hill et al., 2016) is based on a memory network architecture (Weston et al., 2015). We think it is highly similar in spirit. The biggest difference is their way of encoding passages: they demonstrate that it is most effective to only use a 5-word context window when evaluating a candidate entity and they use a positional unigram approach to encode the contextual embeddings: if a window consists of 5 words $x_1, \ldots, x_5$, then it is encoded as $\sum_{i=1}^{5} E_i(x_i)$, resulting in 5 separate embedding matrices to learn. They encode the 5-word window surrounding the placeholder in a similar way and all other words in the question text are ignored. In addition, they simply use a dot product to compute the "relevance" between the question and a contextual embedding. This simple model nevertheless works well, showing the extent to which this RC task can be done by very local context matching.

4 Experiments

4.1 Training Details

For training our conventional classifier, we use the implementation of *LambdaMART* (Wu et al., 2010) in the RankLib package.[5] We use this ranking algorithm since our problem is naturally a ranking problem and forests of boosted decision trees have been very successful lately (as seen, e.g., in many recent Kaggle competitions). We do not use all the features of *LambdaMART* since we are only scoring 1/0 loss on the first ranked proposal, rather than using an IR-style metric to score ranked results. We use Stanford's neural network dependency parser (Chen and Manning, 2014) to parse all our document and question text, and all other features can be extracted without additional tools.

For training our neural networks, we only keep the most frequent $|\mathcal{V}| = 50$k words (including entity and placeholder markers), and map all other words to an <*unk*> token. We choose word embedding size $d = 100$, and use the 100-dimensional pre-trained *GloVe* word embeddings (Pennington et al., 2014) for initialization. The attention and output parameters are initialized from a uniform distribution between $(-0.01, 0.01)$, and the LSTM weights are initialized from a Gaussian distribution $\mathcal{N}(0, 0.1)$.

We use hidden size $h = 128$ for *CNN* and 256 for *Daily Mail*. Optimization is carried out using vanilla stochastic gradient descent (SGD), with a fixed learning rate of 0.1. We sort all the examples by the length of its passage, and randomly sample a mini-batch of size 32 for each update. We also apply dropout with probability 0.2 to the embedding layer and gradient clipping when the norm of gradients exceeds 10.

All of our models are run on a single GPU (GeForce GTX TITAN X), with roughly a runtime of 6 hours per epoch for *CNN*, and 15 hours per epoch for *Daily Mail*. We run all the models up to 30 epochs and select the model that achieves the best accuracy on the development set.

4.2 Main Results

Table 2 presents our main results. The conventional feature-based classifier obtains 67.9% accuracy on the *CNN* test set. Not only does this significantly outperform any of the symbolic approaches reported in (Hermann et al., 2015), it also outperforms all the neural network systems from their paper and the best single-system result reported so far from (Hill et al., 2016). This suggests that the task might not be as difficult as suggested, and a simple feature set can cover many of the cases. Table 3 presents a feature ablation analysis of our entity-centric classifier on the development portion of the *CNN* dataset. It shows that n-gram match and frequency of entities are the two most important classes of features.

More dramatically, our single-model neural network surpasses the previous results by a large margin (over 5%), pushing up the state-of-the-art accuracies to 72.4% and 75.8% respectively. Due to resource constraints, we have not had a chance to investigate ensembles of models, which generally can bring further gains, as demonstrated in (Hill et al., 2016) and many other papers.

Concurrently with our paper, Kadlec et al. (2016) and Kobayashi et al. (2016) also experiment on these two datasets and report competitive results. However, our single model not only still outperforms theirs, but also appears to be structurally simpler. All these recent efforts converge to similar numbers, and we believe that they are approaching the ceiling performance of this task, as we will indicate in the next section.

5 Data Analysis

So far, we have good results via either of our systems. In this section, we aim to conduct an in-depth analysis and answer the following questions: (i) Since the

[5] https://sourceforge.net/p/lemur/wiki/RankLib/.

Model	CNN		Daily Mail	
	Dev	Test	Dev	Test
Frame-semantic model †	36.3	40.2	35.5	35.5
Word distance model †	50.5	50.9	56.4	55.5
Deep LSTM Reader †	55.0	57.0	63.3	62.2
Attentive Reader †	61.6	63.0	70.5	69.0
Impatient Reader †	61.8	63.8	69.0	68.0
MemNNs (window memory) ‡	58.0	60.6	N/A	N/A
MemNNs (window memory + self-sup.) ‡	63.4	66.8	N/A	N/A
MemNNs (ensemble) ‡	66.2*	69.4*	N/A	N/A
Ours: Classifier	67.1	67.9	69.1	68.3
Ours: Neural net	**72.4**	**72.4**	**76.9**	**75.8**

Table 2: Accuracy of all models on the *CNN* and *Daily Mail* datasets. Results marked † are from (Hermann et al., 2015) and results marked ‡ are from (Hill et al., 2016). *Classifier* and *Neural net* denote our entity-centric classifier and neural network systems respectively. The numbers marked with * indicate that the results are from ensemble models.

Features	Accuracy
Full model	67.1
– whether *e* is in the passage	67.1
– whether *e* is in the question	67.0
– frequency of *e*	**63.7**
– position of *e*	65.9
– *n*-gram match	**60.5**
– word distance	65.4
– sentence co-occurrence	66.0
– dependency parse match	65.6

Table 3: Feature ablation analysis of our entity-centric classifier on the development portion of the *CNN* dataset. The numbers denote the accuracy after we exclude each feature from the full system, so a low number indicates an important feature.

dataset was created in an automatic and heuristic way, how many of the questions are trivial to answer, and how many are noisy and not answerable? (ii) What have these models learned? What are the prospects for further improving them? To study this, we randomly sampled 100 examples from the dev portion of the *CNN* dataset for analysis (see more details in Appendix A).

5.1 Breakdown of the Examples

After carefully analyzing these 100 examples, we roughly classify them into the following categories (if an example satisfies more than one category, we classify it into the earliest one):

Exact match The nearest words around the placeholder are also found in the passage surrounding an entity marker; the answer is self-evident.

Sentence-level paraphrasing The question text is entailed/rephrased by *exactly* one sentence in the passage, so the answer can definitely be identified from that sentence.

Partial clue In many cases, even though we cannot find a complete semantic match between the question text and some sentence, we are still able to infer the answer through partial clues, such as some word/concept overlap.

Multiple sentences It requires processing multiple sentences to infer the correct answer.

Coreference errors It is unavoidable that there are many coreference errors in the dataset. This category includes those examples with critical coreference errors for the answer entity or key entities appearing in the question. Basically we treat this category as "not answerable".

Ambiguous or very hard This category includes examples for which we think humans are not able to obtain the correct answer (confidently).

Table 5 provides our estimate of the percentage for each category, and Table 4 presents one representative example from each category. To our surprise,

Category	Question	Passage
Exact Match	*it 's clear @entity0 is leaning toward* **@placeholder** , says an expert who monitors @entity0	. . . @entity116 , who follows @entity0 's operations and propaganda closely , recently told @entity3 , *it 's clear @entity0 is leaning toward* **@entity60** in terms of doctrine , ideology and an emphasis on holding territory after operations . . . .
Paraphrase	**@placeholder** says he understands why @entity0 wo n't play at his tournament	. . . @entity0 called me personally to let me know that he would n't be playing here at @entity23 , " **@entity3** said on his @entity21 event 's website . . . .
Partial clue	a tv movie based on @entity2 's book **@placeholder** casts a @entity76 actor as @entity5	. . . to @entity12 @entity2 professed that his **@entity11** is not a religious book . . . .
Multiple sent.	he 's doing a his - and - her duet all by himself , @entity6 said of **@placeholder**	. . . we got some groundbreaking performances , here too , tonight , @entity6 said . we got **@entity17** , who will be doing some musical performances . he 's doing a his - and - her duet all by himself . . . .
Coref. Error	rapper **@placeholder** " disgusted , " cancels upcoming show for @entity280	. . . with hip - hop star **@entity246** saying on @entity247 that he was canceling an upcoming show for the @entity249 . . . . (but @entity249 = @entity280 = SAEs)
Hard	pilot error and snow were reasons stated for **@placeholder** plane crash	. . . a small aircraft carrying **@entity5** , @entity6 and @entity7 the @entity12 @entity3 crashed a few miles from @entity9 , near @entity10 , @entity11 . . . .

Table 4: Some representative examples from each category.

No.	Category	(%)
1	Exact match	13
2	Paraphrasing	41
3	Partial clue	19
4	Multiple sentences	2
5	Coreference errors	8
6	Ambiguous / hard	17

Table 5: An estimate of the breakdown of the dataset into classes, based on the analysis of our sampled 100 examples from the *CNN* dataset.

Category	Classifier	Neural net
Exact match	13 (100.0%)	13 (100.0%)
Paraphrasing	32 (78.1%)	39 (95.1%)
Partial clue	14 (73.7%)	17 (89.5%)
Multiple sentences	1 (50.0%)	1 (50.0%)
Coreference errors	4 (50.0%)	3 (37.5%)
Ambiguous / hard	2 (11.8%)	1 (5.9%)
All	66 (66.0%)	74 (74.0%)

Table 6: The per-category performance of our two systems.

"coreference errors" and "ambiguous/hard" cases account for 25% of this sample set, based on our manual analysis, and this certainly will be a barrier for training models with an accuracy much above 75% (although, of course, a model can sometimes make a lucky guess). Additionally, only 2 examples require multiple sentences for inference – this is a lower rate than we expected and Hermann et al. (2015) suggest. Therefore, we hypothesize that in most of the "answerable" cases, the goal is to identify the most relevant (single) sentence, and then to infer the answer based upon it.

5.2 Per-category Performance

Now, we further analyze the predictions of our two systems, based on the above categorization.

As seen in Table 6, we have the following observations: (i) The exact-match cases are quite simple and both systems get 100% correct. (ii) For the ambiguous/hard and entity-linking-error cases,

meeting our expectations, both of the systems perform poorly. (iii) The two systems mainly differ in paraphrasing cases, and some of the "partial clue" cases. This clearly shows how neural networks are better capable of learning semantic matches involving paraphrasing or lexical variation between the two sentences. (iv) We believe that the neural-net system already achieves near-optimal performance on all the single-sentence and unambiguous cases. There does not seem to be much useful headroom for exploring more sophisticated natural language understanding approaches on this dataset.

6 Related Tasks

We briefly survey other tasks related to reading comprehension.

MCTest (Richardson et al., 2013) is an open-domain reading comprehension task, in the form of fictional short stories, accompanied by multiple-choice questions. It was carefully created using crowd sourcing, and aims at a 7-year-old reading comprehension level.

On the one hand, this dataset has a high demand on various reasoning capacities: over 50% of the questions require multiple sentences to answer and also the questions come in assorted categories (*what*, *why*, *how*, *whose*, *which*, etc). On the other hand, the full dataset has only 660 paragraphs in total (each paragraph is associated with 4 questions), which renders training statistical models (especially complex ones) very difficult.

Up to now, the best solutions (Sachan et al., 2015; Wang et al., 2015) are still heavily relying on manually curated syntactic/semantic features, with the aid of additional knowledge (e.g., word embeddings, lexical/paragraph databases).

Children Book Test (Hill et al., 2016) was developed in a similar spirit to the *CNN*/*Daily Mail* datasets. It takes any consecutive 21 sentences from a children's book – the first 20 sentences are used as the passage, and the goal is to infer a missing word in the 21st sentence (question and answer). The questions are also categorized by the type of the missing word: named entity, common noun, preposition or verb. According to the first study on this dataset (Hill et al., 2016), a language model (an n-gram model or a recurrent neural network) with local context is sufficient for predicting verbs or prepositions; however, for named entities or common nouns, it improves performance to scan through the whole paragraph to make predictions. So far, the best published results are reported by window-based memory networks.

bAbI (Weston et al., 2016) is a collection of artificial datasets, consisting of 20 different reasoning types. It encourages the development of models with the ability to chain reasoning, induction/deduction, etc., so that they can answer a question like "The football is in the *playground*" after reading a sequence of sentences "John is in the playground; Bob is in the office; John picked up the football; Bob went to the kitchen." Various types of memory networks (Sukhbaatar et al., 2015; Kumar et al., 2016) have been shown effective on these tasks, and Lee et al. (2016) show that vector space models based on extensive problem analysis can obtain near-perfect accuracies on all the categories. Despite these promising results, this dataset is limited to a small vocabulary (only 100–200 words) and simple language variations, so there is still a huge gap from real-world datasets that we need to fill in.

7 Conclusion

In this paper, we carefully examined the recent *CNN*/*Daily Mail* reading comprehension task. Our systems demonstrated state-of-the-art results, but more importantly, we performed a careful analysis of the dataset by hand.

Overall, we think the *CNN*/*Daily Mail* datasets are valuable datasets, which provide a promising avenue for training effective statistical models for reading comprehension tasks. Nevertheless, we argue that: (i) this dataset is still quite noisy due to its method of data creation and coreference errors; (ii) current neural networks have almost reached a performance ceiling on this dataset; and (iii) the required reasoning and inference level of this dataset is still quite simple.

As future work, we need to consider how we can utilize these datasets (and the models trained upon them) to help solve more complex RC reasoning tasks (with less annotated data).

Acknowledgments

We thank the anonymous reviewers for their thoughtful feedback. Stanford University gratefully acknowledges the support of the Defense Advanced Research Projects Agency (DARPA) Deep Exploration and Filtering of Text (DEFT) Program under Air Force Research Laboratory (AFRL) contract no. FA8750-13-2-0040. Any opinions, findings,

and conclusion or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

References

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. 2014. Modeling biological processes for reading comprehension. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510.

Christopher J.C. Burges. 2013. Towards the machine comprehension of text: An essay. Technical report, Microsoft Research Technical Report MSR-TR-2013-125.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1684–1692.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Association for Computational Linguistics (ACL)*.

Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. 2016. Dynamic entity representation with max-pooling improves machine reading. In *North American Association for Computational Linguistics (NAACL)*.

Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *International Conference on Machine Learning (ICML)*.

Moontae Lee, Xiaodong He, Wen-tau Yih, Jianfeng Gao, Li Deng, and Paul Smolensky. 2016. Reasoning in vector space: An exploratory study of question answering. In *International Conference on Learning Representations (ICLR)*.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1412–1421.

Peter Norvig. 1978. *A Unified Theory of Inference for Text Understanding*. Ph.D. thesis, University of California, Berkeley.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. 2013. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 193–203.

Mrinmaya Sachan, Kumar Dubey, Eric Xing, and Matthew Richardson. 2015. Learning answer-entailing structures for machine comprehension. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pages 239–249.

Sainbayar Sukhbaatar, arthur szlam, Jason Weston, and Rob Fergus. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2431–2439.

Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 2015. Machine comprehension with syntax, frames, and semantics. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pages 700–706.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *International Conference on Learning Representations (ICLR)*.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2016. Towards AI-complete question answering: A set of prerequisite toy tasks. In *International Conference on Learning Representations (ICLR)*.

Qiang Wu, Christopher J. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval*, pages 254–270.

A Samples and Labeled Categories from the *CNN* Dataset

For the analysis in Section 5, we uniformly sampled 100 examples from the development set of the *CNN* dataset. Table 8 provides a full index list of our samples and Table 7 presents our labeled categories.

Category	Sample IDs
Exact match (13)	8, 11, 23, 27, 28, 32, 43, 57, 63, 72, 86, 87, 99
Sentence-level paraphrasing (41)	0, 2, 7, 9, 12, 14, 16, 18, 19, 20, 29, 30, 31, 34, 36, 37, 39, 41, 42, 44, 47, 48, 52, 54, 58, 64, 65, 66, 69, 73, 74, 78, 80, 81, 82, 84, 85, 90, 92, 95, 96
Partial clues (19)	4, 17, 21, 24, 35, 38, 45, 53, 55, 56, 61, 62, 75, 83, 88, 89, 91, 97, 98
Multiple sentences (2)	5, 76
Coreference errors (8)	6, 22, 40, 46, 51, 60, 68, 94
Ambiguous or very hard (17)	1, 3, 10, 13, 15, 25, 26, 33, 49, 50, 59, 67, 70, 71, 77, 79, 93

Table 7: Our labeled categories of the 100 samples.

ID	Filename	ID	Filename
0	ddb1e746f88a22fee654ecde8f018e7586595045.question	1	2bef8ec21b10a3294b1496d9a86f29f0592d2300.question
2	38c702812a874f983e9890c32ba832841a327351.question	3	636857045cf266dd69b67b1e53617bed5253dc33.question
4	417cbffd5e6275b3c42cb88be222a9f6c7d415f1.question	5	ef96409c707a699e4055a1d0684eecdb6e115c16.question
6	b4e157a6a34bf11a03e0b5cd55065c0f39ac8d60.question	7	1d75e7c59978c7c06f3aecaf52bc35b8919eee17.question
8	223c8e3aeddc3f65fee1964df17bb72f89b723e4.question	9	13d33b8c86375b0f5fdc856116e91a7355c6fc5a.question
10	378fd418b8ec18dff406be07ec225e6bf53659f5.question	11	d8253b7f22662911c19ec4468f81b9db29df1746.question
12	80529c792d3a368861b404c1ce4d7ad3c12e552a.question	13	728e7b365e941d814676168c78c9c4f38892a550.question
14	3cf6fb2c0d09927a12add82b4a3f248da740d0de.question	15	04b827f84e60659258e19806afe9f8d10b764db1.question
16	f0abf359d71f7896abd09ff7b3319c70f2ded81e.question	17	b6696e0f2166a75fcefbe4f28d0ad06e420eef23.question
18	881ab3139c34e9d9f29eb11601321a234d096272.question	19	66f5208d62b543ee41accb7a560d63ff40413bac.question
20	f83a70d469fa667f0952959346b496fbf3cdb35c.question	21	1853813a80f83a1661dd3f6695559674c749525e.question
22	02664d5e3af321afbaf4ee351ba1f24643746451.question	23	20417b5efb836530846ddf677d1bd0bbc831643c.question
24	42c25a01801228a863c508f9d9e95399ea5f37a4.question	25	70a3ba822770abcaf64dd131c85ec964d172c312.question
26	b6636e525ad58ffdc9a7c18187fb3412660d2cdd.question	27	6147c9f2b3d1cc6fbc57c2137f0356513f49bf46.question
28	262b855e2f24e1b2e4e0ba01ace81a1f214d729e.question	29	d7211f4d21f40461bb59954e53360eeb4bb6c664.question
30	be813e58ae9387a9fdaf771656c8e1122794e515.question	31	ad39c5217042f36e4c1458e9397b4a588bbf8cf9.question
32	9534c3907f1cd917d24a9e4f2afc5b38b82d9fca.question	33	3fbe4bfb721a6e1aa60502089c46240d5c332c05.question
34	6efa2d6bad587bde65ca22d10eca83cf0176d84f.question	35	436aa25e28d3a026c4fcd658a852b6a24fc6935e.question
36	0c44d6ef109d33543cfbd26c95c9c3f6fe33a995.question	37	8472b859c5a8d18454644d9acdb5edd1db175eb5.question
38	fb4dd20e0f464423b6407fd0d21cc4384905cf26.question	39	a192ddbcecf2b00260ae4c7c3c20df4d5ce47a85.question
40	f7133f844967483519dbf632e2f3fb90c5625a4c.question	41	29b274958eb057e8f1688f02ef8dbc1c6d06c954.question
42	8ea6ad57c1c5eb1950f50ea47231a5b3f32dd639.question	43	1e43f2349b17dac6d1b3143f8c5556e2257be92c.question
44	7f11f0b4f6bb9aaa3bdc74bffaed5c869b26be97.question	45	8e6d8d984e51adb5071aad22680419854185eaea.question
46	57fc2b7ffcfbd1068fbc33b95d5786e2bff24698.question	47	57b773478955811a8077c98840d85af03e1b4f05.question
48	d857700721b5835c3472ba73ef7abfad0c9c499f.question	49	f8eedded53c96e0cb98e2e95623714d2737f29da.question
50	4c488f41622ad48977a60c2283910f15a736417e.question	51	39680fd0bff53f2ca02f632eabbc024d698f979e.question
52	addd9cebe24c96b4a3c8e9a50cd2a57905b6defb.question	53	50317f7a626e23628e4bfd190e987ad5af7d283e.question
54	3f7ac912a75e4ef7a56987bff37440ffa14770c6.question	55	610012ef561027623f4b4e3b8310c1c41dc819cc.question
56	d9c2e9bfc71045be2ecd959676016599e4637ed1.question	57	848c068db210e0b255f83c4f8b01d2d421fb9c94.question
58	f5c2753703b66d26f43bafe7f157803dc96eedbc.question	59	4f76379f1c7b1d4acc5a4c82ced64af6313698dd.question
60	e5bb1c27d07f1591929bf0283075ad1bc1fc0b50.question	61	33b911f9074c80eb18a57f657ad01393582059be.question
62	58c4c046654af52a3cb8f6890411a41c0dd0063b.question	63	7b03f730fda1b247e9f124b692e3298859785ef3.question
64	ece6f4e047856d5a84811a67ac9780d48044e69a.question	65	35565dc6aecc0f1203842ef13aede0a14a8cf075.question
66	ddf3f2b06353fe8a9b50043f926eb3ab318e91b2.question	67	e248e59739c9c013a2b1b7385d881e0f879b341d.question
68	e86d3fa2a74625620bcae0003dfbe13416ee29cf.question	69	176bf03c9c19951a8ae5197505a568454a6d4526.question
70	ee694cb968ae99aea36f910355bf73da417274c0.question	71	7a666f78590edbaf7c4d73c4ea641c545295a513.question
72	91e3cdd46a70d6dfbe917c6241eab907da4b1562.question	73	e54d9bdcb478ecc490608459d3405571979ef3f2.question
74	f3737e4de9864f083d6697293be650e02505768c.question	75	1fc7488755d24696a4ed1aabc0a21b8b9755d8c6.question
76	fb3eadd07b9f1df1f8a7a6b136ad6d06f4981442.question	77	1406bdad74b3f932342718d5d5d0946a906d73e2.question
78	54b6396669bdb2e30715085745d4f98d058269ef.question	79	0a53102673f2bebc36ce74bf71db1b42a0187052.question
80	d5eb4f98551d23810bfeb0e5b8a94037bcf58b0d.question	81	370de4ffe0f2f9691e4bd456ff344a6a337e0edf.question
82	12f32c770c86083ff21b25de7626505c06440018.question	83	9f6b5cff3ce146e21e323a1462c3eff8fca3d4a0.question
84	1c2a14f525fa3802b8da52aebaa9abd2091f9215.question	85	f2416e14d89d40562284ba2d15f7d5cc59c7e602.question
86	adcf5881856bcbaf1ad93d06a3c5431f6a0319ba.question	87	097d34b804c4c052591984d51444c4a97a3c41ac.question
88	773066c39bb3b593f676caf03f7e7370a8cd2a43.question	89	598cf5ff08ea75dcedda31ac1300e49cdf90893a.question
90	b66ebaaefb844f1216fd3d28eb160b08f42cde62.question	91	535a44842decdc23c11bae50d9393b923897187e.question
92	e27ca3104a596171940db8501c4868ed2fbc8cea.question	93	bb07799b4193cffa90792f92a8c14d591754a7f3.question
94	83ff109c6ccd512abdf317220337b98ef551d94a.question	95	5ede07a1e4ac56a0155d852df0f5bb6bde3cb507.question
96	7a2a9a7fbb44b0e51512c61502ce2292170400c1.question	97	9dcdc052682b041cdbf2fadc8e55f1bafc88fe61.question
98	0c2e28b7f373f29f3796d29047556766cc1dd709.question	99	2bdf1696bfd2579bb719402e9a6fa99cb8dbf587.question

Table 8: A full index list of our samples.

Learning Language Games through Interaction

Sida I. Wang **Percy Liang** **Christopher D. Manning**
Computer Science Department
Stanford University
{sidaw,pliang,manning}@cs.stanford.edu

Abstract

We introduce a new language learning setting relevant to building adaptive natural language interfaces. It is inspired by Wittgenstein's language games: a human wishes to accomplish some task (e.g., achieving a certain configuration of blocks), but can only communicate with a computer, who performs the actual actions (e.g., removing all red blocks). The computer initially knows nothing about language and therefore must learn it from scratch through interaction, while the human adapts to the computer's capabilities. We created a game called SHRDLURN in a blocks world and collected interactions from 100 people playing it. First, we analyze the humans' strategies, showing that using compositionality and avoiding synonyms correlates positively with task performance. Second, we compare computer strategies, showing that modeling pragmatics on a semantic parsing model accelerates learning for more strategic players.

1 Introduction

Wittgenstein (1953) famously said that *language derives its meaning from use*, and introduced the concept of *language games* to illustrate the fluidity and purpose-orientedness of language. He described how a builder B and an assistant A can use a primitive language consisting of four words—*'block'*, *'pillar'*, *'slab'*, *'beam'*—to successfully communicate what block to pass from A to B. This is only one such language; many others would also work for accomplishing the cooperative goal.

This paper operationalizes and explores the idea of language games in a learning setting, which we call *interactive learning through language games*

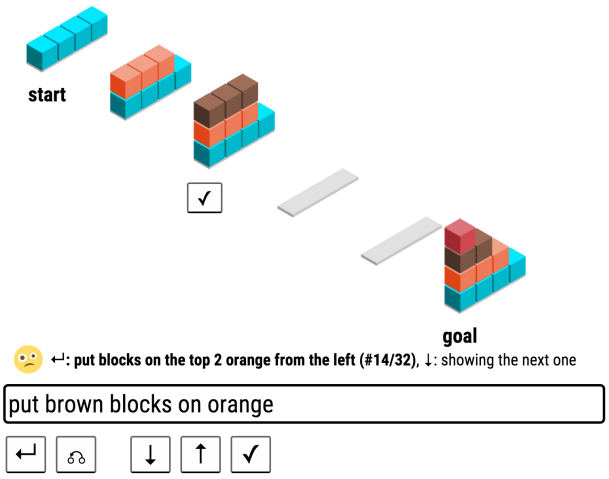


Figure 1: The SHRDLURN game: the objective is to transform the start state into the goal state. The human types in an utterance, and the computer (which does not know the goal state) tries to interpret the utterance and perform the corresponding action. The computer initially knows nothing about the language, but through the human's feedback, learns the human's language while making progress towards the game goal.

(ILLG). In the ILLG setting, the two parties do not initially speak a common language, but nonetheless need to collaboratively accomplish a goal. Specifically, we created a game called SHRDLURN,[1] in homage to the seminal work of Winograd (1972). As shown in Figure 1, the objective is to transform a start state into a goal state, but the only action the human can take is entering an utterance. The computer parses the utterance and produces a ranked list of possible interpretations according to its current model. The human scrolls through the list and chooses the intended one, simultaneously advancing the state of the blocks and providing feedback to the computer. Both the human and the computer wish to reach the goal state

[1]Demo: http://shrdlurn.sidaw.xyz

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2368–2378,
Berlin, Germany, August 7-12, 2016.

(only known to the human) with as little scrolling as possible. For the computer to be successful, it has to learn the human's language quickly over the course of the game, so that the human can accomplish the goal more efficiently. Conversely, the human must also accommodate the computer, at least partially understanding what it can and cannot do.

We model the computer in the ILLG as a semantic parser (Section 3), which maps natural language utterances (e.g., *'remove red'*) into logical forms (e.g., `remove(with(red))`). The semantic parser has no seed lexicon and no annotated logical forms, so it just generates many candidate logical forms. Based on the human's feedback, it performs online gradient updates on the parameters corresponding to simple lexical features.

During development, it became evident that while the computer was eventually able to learn the language, it was learning less quickly than one might hope. For example, after learning that *'remove red'* maps to `remove(with(red))`, it would think that *'remove cyan'* also mapped to `remove(with(red))`, whereas a human would likely use mutual exclusivity to rule out that hypothesis (Markman and Wachtel, 1988). We therefore introduce a pragmatics model in which the computer explicitly reasons about the human, in the spirit of previous work on pragmatics (Golland et al., 2010; Frank and Goodman, 2012; Smith et al., 2013). To make the model suitable for our ILLG setting, we introduce a new online learning algorithm. Empirically, we show that our pragmatic model improves the online accuracy by 8% compared to our best non-pragmatic model on the 10 most successful players (Section 5.3).

What is special about the ILLG setting is the real-time nature of learning, in which the human also learns and adapts to the computer. While the human can teach the computer any language—English, Arabic, Polish, a custom programming language—a good human player will choose to use utterances that the computer is more likely to learn quickly. In the parlance of communication theory, the human *accommodates* the computer (Giles, 2008; Ireland et al., 2011). Using Amazon Mechanical Turk, we collected and analyzed around 10k utterances from 100 games of SHRDLURN. We show that successful players tend to use compositional utterances with a consistent vocabulary and syntax, which matches the inductive biases of the computer (Section 5.2). In addition, through this interaction, many players adapt to the computer by becoming more consistent, more precise, and more concise.

On the practical side, natural language systems are often trained once and deployed, and users must live with their imperfections. We believe that studying the ILLG setting will be integral for creating adaptive and customizable systems, especially for resource-poor languages and new domains where starting from close to scratch is unavoidable.

2 Setting

We now describe the interactive learning of language games (ILLG) setting formally. There are two players, the human and the computer. The game proceeds through a fixed number of levels. In each level, both players are presented with a starting state $s \in \mathcal{Y}$, but only the human sees the goal state $t \in \mathcal{Y}$. (e.g. in SHRDLURN, $\mathcal{Y}$ is the set of all configurations of blocks). The human transmits an utterance x (e.g., *'remove red'*) to the computer. The computer then constructs a ranked list of candidate actions $Z = [z_1, \ldots, z_K] \subseteq \mathcal{Z}$ (e.g., `remove(with(red))`, `add(with(orange))`, etc.), where $\mathcal{Z}$ is all possible actions. For each $z_i \in Z$, it computes $y_i = [\![z_i]\!]_s$, the successor state from executing action z_i on state s. The computer returns to the human the ordered list $Y = [y_1, \ldots, y_K]$ of successor states. The human then chooses y_i from the list Y (we say the computer is *correct* if $i = 1$). The state then updates to $s = y_i$. The level ends when $s = t$, and the players advance to the next level.

Since only the human knows the goal state t and only the computer can perform actions, the only way for the two to play the game successfully is for the human to somehow encode the desired action in the utterance x. However, we assume the two players do not have a shared language, so the human needs to pick a language and teach it to the computer. As an additional twist, the human does not know the exact set of actions $\mathcal{Z}$ (although they might have some preconception of the computer's capabilities).[2] Finally, the human only sees the outcomes of the computer's actions, not the actual logical actions themselves.

We expect the game to proceed as follows: In the beginning, the computer does not understand

[2]This is often the case when we try to interact with a new software system or service before reading the manual.

what the human is saying and performs arbitrary actions. As the computer obtains feedback and learns, the two should become more proficient at communicating and thus playing the game. Herein lies our key design principle: *language learning should be necessary for the players to achieve good game performance*.

SHRDLURN. Let us now describe the details of our specific game, SHRDLURN. Each state $s \in \mathcal{Y}$ consists of stacks of colored blocks arranged in a line (Figure 1), where each stack is a vertical column of blocks. The actions $\mathcal{Z}$ are defined compositionally via the grammar in Table 1. Each action either adds to or removes from a set of stacks, and a set of stacks is computed via various set operations and selecting by color. For example, the action `remove(leftmost(with(red)))` removes the top block from the leftmost stack whose topmost block is red. The compositionality of the actions gives the computer non-trivial capabilities. Of course, the human must teach a language to harness those capabilities, while not quite knowing the exact extent of the capabilities. The actual game proceeds according to a curriculum, where the earlier levels only need simpler actions with fewer predicates.

We designed SHRDLURN in this way for several reasons. First, visual block manipulations are intuitive and can be easily crowdsourced, and it can be fun as an actual game that people would play. Second, the action space is designed to be compositional, mirroring the structure of natural language. Third, many actions z lead to the same successor state $y = [\![z]\!]_s$; e.g., the *'leftmost stack'* might coincide with the *'stack with red blocks'* for some state s and therefore an action involving either one would result in the same outcome. Since the human only points out the correct y, the computer must grapple with this indirect supervision, a reflection of real language learning.

3 Semantic parsing model

Following Zettlemoyer and Collins (2005) and most recent work on semantic parsing, we use a log-linear model over logical forms (actions) $z \in \mathcal{Z}$ given an utterance x:

$$p_\theta(z \mid x) \propto \exp(\theta^\mathsf{T} \phi(x, z)), \tag{1}$$

where $\phi(x, z) \in \mathbb{R}^d$ is a feature vector and $\theta \in \mathbb{R}^d$ is a parameter vector. The denotation y (successor state) is obtained by executing z on a state s; formally, $y = [\![z]\!]_s$.

Features. Our features are n-grams (including skip-grams) conjoined with tree-grams on the logical form side. Specifically, on the utterance side (e.g., *'stack red on orange'*), we use unigrams (*'stack'*, $*$, $*$), bigrams (*'red'*, *'on'*, $*$), trigrams (*'red'*, *'on'*, *'orange'*), and skip-trigrams (*'stack'*, $*$, *'on'*). On the logical form side, features corresponds to the predicates in the logical forms and their arguments. For each predicate h, let $h.i$ be the i-th argument of h. Then, we define *tree-gram* features $\psi(h, d)$ for predicate h and depth $d = 0, 1, 2, 3$ recursively as follows:

$$\psi(h, 0) = \{h\},$$
$$\psi(h, d) = \{(h, i, \psi(h.i, d-1)) \mid i = 1, 2, 3\}.$$

The set of all features is just the cross product of utterance features and logical form features. For example, if $x =$ *'enlever tout'* and $z =$ `remove(all())`, then features include:

(*'enlever'*, `all`) (*'tout'*, `all`)
(*'enlever'*, `remove`) (*'tout'*, `remove`)
(*'enlever'*, (`remove`, 1, `all`))
(*'tout'*, (`remove`, 1, `all`))

Note that we do not model an explicit alignment or derivation compositionally connecting the utterance and the logical form, in contrast to most traditional work in semantic parsing (Zettlemoyer and Collins, 2005; Wong and Mooney, 2007; Liang et al., 2011; Kwiatkowski et al., 2010; Berant et al., 2013), instead following a looser model of semantics similar to (Pasupat and Liang, 2015). Modeling explicit alignments or derivations is only computationally feasible when we are learning from annotated logical forms or have a seed lexicon, since the number of derivations is much larger than the number of logical forms. In the ILLG setting, neither are available.

Generation/parsing. We generate logical forms from smallest to largest using beam search. Specifically, for each size $n = 1, \ldots, 8$, we construct a set of logical forms of size n (with exactly n predicates) by combining logical forms of smaller sizes according to the grammar rules in Table 1. For each n, we keep the 100 logical forms z with the highest score $\theta^\mathsf{T} \phi(x, z)$ according to the current model θ. Let Z be the set of logical forms on the final beam, which contains logical forms of all sizes n. During training, due to pruning at

Rule	Semantics	Description
Set	`all()`	all stacks
Color	`cyan|brown|red|orange`	primitive color
Color → Set	`with(`c`)`	stacks whose top block has color c
Set → Set	`not(`s`)`	all stacks except those in s
Set → Set	`leftmost|rightmost(`s`)`	leftmost/rightmost stack in s
Set Color → Act	`add(`s, c`)`	add block with color c on each stack in s
Set → Act	`remove(`s`)`	remove the topmost block of each stack in s

Table 1: The formal grammar defining the compositional action space $\mathcal{Z}$ for SHRDLURN. We use c to denote a Color, and s to denote a Set. For example, one action that we have in SHRDLURN is: *'add an orange block to all but the leftmost brown block'* $\mapsto$ `add(not(leftmost(with(brown))),orange)`.

intermediate sizes, Z is not guaranteed to contain the logical form that obtains the observed state y. To mitigate this effect, we use a curriculum so that only simple actions are needed in the initial levels, giving the human an opportunity to teach the computer about basic terms such as colors first before moving to larger composite actions.

The system executes all of the logical forms on the final beam Z, and orders the resulting denotations y by the maximum probability of any logical form that produced it.[3]

Learning. When the human provides feedback in the form of a particular y, the system forms the following loss function:

$$\ell(\theta, x, y) = -\log p_\theta(y \mid x, s) + \lambda||\theta||_1, \quad (2)$$

$$p_\theta(y \mid x, s) = \sum_{z:[\![z]\!]_s = y} p_\theta(z \mid x). \quad (3)$$

Then it makes a single gradient update using AdaGrad (Duchi et al., 2010), which maintains a per-feature step size.

4 Modeling pragmatics

In our initial experience with the semantic parsing model described in Section 3, we found that it was able to learn reasonably well, but lacked a reasoning ability that one finds in human learners. To illustrate the point, consider the beginning of a game when $\theta = 0$ in the log-linear model $p_\theta(z \mid x)$. Suppose that human utters *'remove red'* and then identifies $z_{\texttt{rm-red}} = \texttt{remove(with(red))}$ as the correct logical form. The computer then performs a gradient update on the loss function (2), upweighting features such as (*'remove'*, `remove`) and (*'remove'*, `red`).

Next, suppose the human utters *'remove cyan'*. Note that $z_{\texttt{rm-red}}$ will score higher than all other formulas since the (*'remove'*, `red`) feature will fire again. While statistically justified, this behavior fails to meet our intuitive expectations for a smart language learner. Moreover, this behavior is not specific to our model, but applies to any statistical model that simply tries to fit the data without additional prior knowledge about the specific language. While we would not expect the computer to magically guess *'remove cyan'* $\mapsto$ `remove(with(cyan))`, it should at least push down the probability of $z_{\texttt{rm-red}}$ because $z_{\texttt{rm-red}}$ intuitively is already well-explained by another utterance *'remove red'*.

This phenomenon, *mutual exclusivity*, was studied by Markman and Wachtel (1988). They found that children, during their language acquisition process, reject a second label for an object and treat it instead as a label for a novel object.

The pragmatic computer. To model mutual exclusivity formally, we turn to probabilistic models of pragmatics (Golland et al., 2010; Frank and Goodman, 2012; Smith et al., 2013; Goodman and Lassiter, 2015), which operationalize the ideas of Grice (1975). The central idea in these models is to treat language as a cooperative game between a speaker (human) and a listener (computer) as we are doing, but where the listener has an explicit model of the speaker's strategy, which in turn models the listener. Formally, let $S(x \mid z)$ be the speaker's strategy and $L(z \mid x)$ be the listener's

[3] We tried ordering based on the sum of the probabilities (which corresponds to marginalizing out the logical form), but this had the degenerate effect of assigning too much probability mass to y being the set of empty stacks, which can result from many actions.

	$z_{\texttt{rm-red}}$	$z_{\texttt{rm-cyan}}$	$z_3, z_4, \ldots$
	$p_\theta(z \mid x)$		
'remove red'	0.8	0.1	0.1
'remove cyan'	**0.6**	0.2	0.2
	$S(x \mid z)$		
'remove red'	0.57	0.33	0.33
'remove cyan'	0.43	0.67	0.67
	$L(z \mid x)$		
'remove red'	0.46	0.27	0.27
'remove cyan'	0.24	**0.38**	**0.38**

Table 2: Suppose the computer saw one example of *'remove red'* $\mapsto z_{\texttt{rm-red}}$, and then the human utters *'remove cyan'*. **top**: the literal listener, $p_\theta(z \mid x)$, mistakingly chooses $z_{\texttt{rm-red}}$ over $z_{\texttt{rm-cyan}}$. **middle**: the pragmatic speaker, $S(x \mid z)$, assigns a higher probability to *'remove cyan'* given $z_{\texttt{rm-cyan}}$; **bottom**: the pragmatic listener, $L(z \mid x)$ correctly assigns a lower probability to $z_{\texttt{rm-red}}$ where $p(z)$ is uniform.

strategy. The speaker takes into account the literal semantic parsing model $p_\theta(z \mid x)$ as well as a prior over utterances $p(x)$, while the listener considers the speaker $S(x \mid z)$ and a prior $p(z)$:

$$S(x \mid z) \propto \left(p_\theta(z \mid x)p(x)\right)^\beta, \quad (4)$$

$$L(z \mid x) \propto S(x \mid z)p(z), \quad (5)$$

where $\beta \geq 1$ is a hyperparameter that sharpens the distribution (Smith et al., 2013). The computer would then use $L(z \mid x)$ to rank candidates rather than p_θ. Note that our pragmatic model only affects the ranking of actions returned to the human and does not affect the gradient updates of the model p_θ.

Let us walk through a simple example to see the effect of modeling pragmatics. Table 2 shows that the literal listener $p_\theta(z \mid x)$ assigns high probability to $z_{\texttt{rm-red}}$ for both *'remove red'* and *'remove cyan'*. Assuming a uniform $p(x)$ and $\beta = 1$, the pragmatic speaker $S(x \mid z)$ corresponds to normalizing each column of p_θ. Note that if the pragmatic speaker wanted to convey $z_{\texttt{rm-cyan}}$, there is a decent chance that they would favor *'remove cyan'*. Next, assuming a uniform $p(z)$, the pragmatic listener $L(z \mid x)$ corresponds to normalizing each row of $S(x \mid z)$. The result is that conditioned on *'remove cyan'*, $z_{\texttt{rm-cyan}}$ is now more likely than $z_{\texttt{rm-red}}$, which is the desired effect.

The pragmatic listener models the speaker as a cooperative agent who behaves in a way to maximize communicative success. Certain speaker behaviors such as avoiding synonyms (e.g., not *'delete cardinal'*) and using a consistent word ordering (e.g, not *'red remove'*) fall out of the game theory.[4] For speakers that do not follow this strategy, our pragmatic model is incorrect, but as we get more data through game play, the literal listener $p_\theta(z \mid x)$ will sharpen, so that the literal listener and the pragmatic listener will coincide in the limit.

$\forall z, C(z) \leftarrow 0$
$\forall z, Q(z) \leftarrow \epsilon$
repeat
  receive utterance x from human
  $L(z \mid x) \propto \frac{P(z)}{Q(z)} p_\theta(z \mid x)^\beta$
  send human a list Y ranked by $L(z \mid x)$
  receive $y \in Y$ from human
  $\theta \leftarrow \theta - \eta \nabla_\theta \ell(\theta, x, y)$
  $Q(z) \leftarrow Q(z) + p_\theta(z \mid x)^\beta$
  $C(z) \leftarrow C(z) + p_\theta(z \mid x, [\![z]\!]_s = y)$
  $P(z) \leftarrow \frac{C(z)+\alpha}{\sum_{z':C(z')>0}\left(C(z')+\alpha\right)}$
until *game ends*

Algorithm 1: Online learning algorithm that updates the parameters of the semantic parser θ as well as counts C, Q required to perform pragmatic reasoning.

Online learning with pragmatics. To implement the pragmatic listener as defined in (5), we need to compute the speaker's normalization constant $\sum_x p_\theta(z \mid x)p(x)$ in order to compute $S(x \mid z)$ in (4). This requires parsing all utterances x based on $p_\theta(z \mid x)$. To avoid this heavy computation in an online setting, we propose Algorithm 1, where some approximations are used for the sake of efficiency. First, to approximate the intractable sum over all utterances x, we only use the examples that are seen to compute the normalization constant $\sum_x p_\theta(z \mid x)p(x) \approx \sum_i p_\theta(z \mid x_i)$. Then, in order to avoid parsing all previous examples again using the current parameters for each new example, we store $Q(z) = \sum_i p_{\theta_i}(z \mid x_i)^\beta$, where θ_i is the parameter after the model updates on the i^{th} example x_i. While θ_i is different from the current parameter θ, $p_\theta(z \mid x_i) \approx p_{\theta_i}(z \mid x_i)$ for the relevant example x_i, which is accounted for

[4] Of course, synonyms and variable word order occur in real language. We would need a more complex game compared to SHRDLURN to capture this effect.

by both θ_i and θ.

In Algorithm 1, the pragmatic listener $L(z \mid x)$ can be interpreted as an importance-weighted version of the sharpened literal listener p_θ^β, where it is downweighted by $Q(z)$, which reflects which z's the literal listener prefers, and upweighted by $P(z)$, which is just a smoothed estimate of the actual distribution over logical forms $p(z)$. By construction, Algorithm 1 is the same as (4) except that it uses the normalization constant Q based on stale parameters θ_i after seeing example, and it uses samples to compute the sum over x. Following (5), we also need $p(z)$, which is estimated by $P(z)$ using add-α smoothing on the counts $C(z)$. Note that $Q(z)$ and $C(z)$ are updated *after* the model parameters are updated for the current example.

Lastly, there is a small complication due to only observing the denotation y and not the logical form z. We simply give each consistent logical form $\{z \mid [\![z]\!]_s = y\}$ a pseudocount based on the model: $C(z) \leftarrow C(z) + p_\theta(z \mid x, [\![z]\!]_s = y)$ where $p_\theta(z \mid x, [\![z]\!]_s = y) \propto \exp(\theta^\mathsf{T}\phi(x, z))$ for $[\![z]\!]_s = y$ (0 otherwise).

Compared to prior work where the setting is specifically designed to require pragmatic inference, pragmatics arises naturally in ILLG. We think that this form of pragmatics is the most important during learning, and becomes less important if we had more data. Indeed, if we have a lot of data and a small number of possible zs, then $L(z|x) \approx p_\theta(z|x)$ as $\sum_x p_\theta(z|x)p(x) \to p(z)$ when $\beta = 1$.[5] However, for semantic parsing, we would not be in this regime even if we have a large amount of training data. In particular, we are nowhere near that regime in SHRDLURN, and most of our utterances / logical forms are seen only once, and the importance of modeling pragmatics remains.

5 Experiments

5.1 Setting

Data. Using Amazon Mechanical Turk (AMT), we paid 100 workers 3 dollars each to play SHRDLURN. In total, we have 10223 utterances along with their starting states s. Of these, 8874 utterances are labeled with their denotations y; the rest are unlabeled, since the player can try any utterance without accepting an action. 100 players completed the entire game under identical settings. We deliberately chose to start from scratch for every worker, so that we can study the diversity of strategies that different people used in a controlled setting.

Each game consists of 50 blocks tasks divided into 5 levels of 10 tasks each, in increasing complexity. Each level aims to reach an end goal given a start state. Each game took on average 89 utterances to complete.[6] It only took 6 hours to complete these 100 games on AMT and each game took around an hour on average according to AMT's *work time* tracker (which does not account for multi-tasking players). The players were provided minimal instructions on the game controls. Importantly, we gave no example utterances in order to avoid biasing their language use. Around 20 players were confused and told us that the instructions were not clear and gave us mostly spam utterances. Fortunately, most players understood the setting and some even enjoyed SHRDLURN as reflected by their optional comments:

- *That was probably the most fun thing I have ever done on mTurk.*
- *Wow this was one mind bending games [sic].*

Metrics. We use the *number of scrolls* as a measure of game performance for each player. For each example, the number of scrolls is the position in the list Y of the action selected by the player. It was possible to complete this version of SHRDLURN by scrolling (all actions can be found in the first 125 of Y)—22 of the 100 players failed to teach an actual language, and instead finished the game mostly by scrolling. Let us call them *spam players*, who usually typed single letters, random words, digits, or random phrases (e.g. *'how are you'*). Overall, spam players had to scroll a lot: 21.6 scrolls per utterance versus only 7.4 for the non-spam players.

5.2 Human strategies

Some example utterances can be found in Table 3. Most of the players used English, but vary in their adherence to conventions such as use of determiners, plurals, and proper word ordering. 5 players invented their own language, which are more precise, more consistent than general English. One player used Polish, and another used Polish notation (bottom of Table 3).

[5] Technically, we also need p_θ to be *well-specified*.

[6] This number is not 50 because some block tasks need multiple steps and players are also allowed to explore without reaching the goal.

Most successful players (1st–20th)		
rem cy pos 1, stack or blk pos 4, rem blk pos 2 thru 5, rem blk pos 2 thru 4, stack bn blk pos 1 thru 2, fill bn blk, stack or blk pos 2 thru 6, rem cy blk pos 2 fill rd blk **(3.01)**	remove the brown block, remove all orange blocks, put brown block on orange blocks, put orange blocks on all blocks, put blue block on leftmost blue block in top row **(2.78)**	Remove the center block, Remove the red block, Remove all red blocks, Remove the first orange block, Put a brown block on the first brown block, Add blue block on first blue block **(2.72)**
Average players (21th–50th)		
reinsert pink, take brown, put in pink, remove two pink from second layer, Add two red to second layer in odd intervals, Add five pink to second layer, Remove one blue and one brown from bottom layer **(9.17)**	remove red, remove 1 red, remove 2 4 orange, add 2 red, add 1 2 3 4 blue, emove 1 3 5 orange, add 2 4 orange, add 2 orange, remove 2 3 brown, add 1 2 3 4 5 red, remove 2 3 4 5 6, remove 2, add 1 2 3 4 6 red **(8.37)**	move second cube, double red with blue, double first red with red, triple second and fourth with orange, add red, remove orange on row two, add blue to column two, add brown on first and third **(7.18)**
Least successful players (51th–)		
holdleftmost, holdbrown, holdleftmost, blueonblue, brownonblue1, blueonorange, holdblue, holdorange2, blueonred2 , holdends1, holdrightend, hold2, orangeonorangerightmost **(14.15)**	'add red cubes on center left, center right, far left and far right', 'remove blue blocks on row two column two, row two column four', remove red blocks in center left and center right on second row **(12.6)**	laugh with me, red blocks with one aqua, aqua red alternate, brown red red orange aqua orange, red brown red brown red brown, space red orange red, second level red space red space red space **(14.32)**
Spam players (∼ 85th–100)		
next, hello happy, how are you, move, gold, build goal blocks, 23,house, gabboli, x, run,,xav, d, j, xcv, dulicate goal (21.7)		
Most interesting		
usuń brązowe klocki, postaw pomarańczowy klocek na pierwszym klocku, postaw czerwone klocki na pomarańczowych, usuń pomarańczowe klocki w górnym rzędzie	rm scat + 1 c, + 1 c, rm sh, + 1 2 4 sh, + 1 c, - 4 o, rm 1 r, + 1 3 o, full fill c, rm o, full fill sh, - 1 3, full fill sh, rm sh, rm r, + 2 3 r, rm o, + 3 sh, + 2 3 sh, rm b, - 1 o, + 2 c,	mBROWN,mBLUE,mORANGE RED+ORANGEˆORANGE, BROWN+BROWNm1+BROWNm3, ORANGE +BROWN +ORANGEˆm1+ ORANGEˆm3 + BROWNˆˆ2 + BROWNˆˆ4

Table 3: Example utterances, along with the average number of scrolls for that player in parentheses. Success is measured by the number of scrolls, where the more successful players need less scrolls. 1) The 20 most successful players tend to use consistent and concise language whose semantics is similar to our logical language. 2) Average players tend to be slightly more verbose and inconsistent (left and right), or significantly different from our logical langauge (middle). 3) Reasons for being unsuccessful vary. Left: no tokenization, middle: used a coordinate system and many conjunctions; right: confused in the beginning, and used a language very different from our logical language.

Overall, we find that many players adapt in ILLG by becoming more consistent, less verbose, and more precise, even if they used standard English at the beginning. For example, some players became more consistent over time (e.g. from using both *'remove'* and *'discard'* to only using *'remove'*). In terms of verbosity, removing function words like determiners as the game progresses is a common adaptation. In each of the following examples from different players, we compare an utterance that appeared early in the game to a similar utterance that appeared later: *'Remove the red ones'* **became** *'Remove red.'*; *'add brown on top of red'* **became** *'add orange on red'*; *'add red blocks to all red blocks'* **became** *'add red to red'*; *'dark red'* **became** *'red'*; one player used *'the'* in all of the first 20 utterances, and then never used *'the'* in the last 75 utterances.

Players also vary in precision, ranging from overspecified (e.g. *'remove the orange cube at the left'*, *'remove red blocks from top row'*) to underspecified or requiring context (e.g. *'change colors'*, *'add one blue'*, *'Build more blocus'*, *'Move the blocks fool'*,*'Add two red cubes'*). We found that some players became more precise over time, as they gain a better understanding of ILLG.

Most players use utterances that actually do not match our logical language in Table 1, even the successful players. In particular, numbers are often used. While some concepts always have the same effect in our blocks world (e.g. *'first block'* means `leftmost`), most are different. More concretely, of the top 10 players, 7 used numbers of some form and only 3 players matched our logical language. Some players who did not match the logical language performed quite well neverthe-

less. One possible explanation is because the action required is somewhat constrained by the logical language and some tokens can have unintended interpretations. For example, the computer can correctly interpret numerical positional references, as long as the player only refers to the leftmost and rightmost positions. So if the player says *'rem blk pos 4'* and *'rem blk pos 1'*, the computer can interpret *'pos'* as `rightmost` and interpret the bigram (*'pos'*, *'1'*) as `leftmost`. On the other hand, players who deviated significantly by describing the desired state declaratively (e.g. *'red orange red'*, *'246'*) rather than using actions, or a coordinate system (e.g. *'row two column two'*) performed poorly. Although players do not have to match our logical language exactly to perform well, being similar is definitely helpful.

Compositionality. As far as we can tell, all players used a compositional language; no one invented unrelated words for each action. Interestingly, 3 players did not put spaces between words. Since we assume monomorphemic words separated by spaces, they had to do a lot of scrolling as a result (e.g., 14.15 with utterances like *'orangeonorangerightmost'*).

5.3 Computer strategies

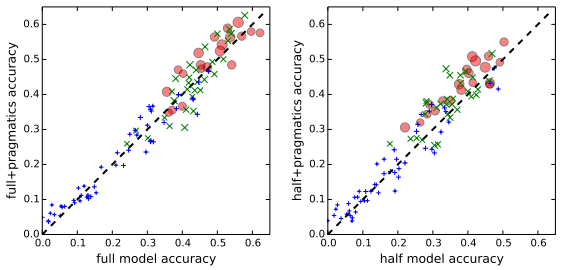
We now present quantitative results on how quickly the computer can learn, where our goal is to achieve high accuracy on new utterances as we make just a single pass over the data. The number of scrolls used to evaluate player is sensitive to outliers and not as intuitive as accuracy. Instead, we consider *online accuracy*, described as follows. Formally, if a player produced T utterances $x^{(j)}$ and labeled them $y^{(j)}$, then

$$\text{online accuracy} \stackrel{\text{def}}{=} \frac{1}{T}\sum_{j=1}^{T} \mathbb{I}\left[y^{(j)} = [\![z^{(j)}]\!]_{s^{(j)}}\right],$$

where $z^{(j)} = \arg\max_z p_{\theta^{(j-1)}}(z|x^{(j)})$ is the model prediction based on the previous parameter $\theta^{(j-1)}$. Note that the online accuracy is defined with respect to the player-reported labels, which only corresponds to the actual accuracy if the player is precise and honest. This is not true for most spam players.

Compositionality. To study the importance of compositionality, we consider two baselines. First, consider a non-compositional model (*memorize*) that just remembers pairs of complete utterance and logical forms. We implement this using indicator features on features (x, z), e.g., (*'remove all the red blocks'*, $z_{\texttt{rm-red}}$), and use a large learning rate. Second, we consider a model (*half*) that treats utterances compositionally with unigrams, bigrams, and skip-trigrams features, but the logical forms are regarded as non-compositional, so we have features such as (*'remove'*, $z_{\texttt{rm-red}}$), (*'red'*, $z_{\texttt{rm-red}}$), etc.

Table 4 shows that the full model (Section 3) significantly outperforms both the *memorize* and *half* baselines. The learning rate $\eta = 0.1$ is selected via cross validation, and we used $\alpha = 1$ and $\beta = 3$ following Smith et al. (2013).



(a) (b)

Figure 2: Pragmatics improve online accuracy. In these plots, each marker is a player. red o: players who ranked 1–20 in terms of minimizing number of scrolls, green x: players 20–50; blue +: lower than 50 (includes spam players). Marker sizes correspond to player rank, where better players are depicted with larger markers. **2a**: online accuracies with and without pragmatics on the full model; **2b**: same for the half model.

	players ranked by # of scrolls			
Method	top 10	top 20	top 50	all 100
memorize	25.4	24.5	22.5	17.6
half model	38.7	38.4	36.0	27.0
half + prag	43.7	42.7	39.7	29.4
full model	48.6	47.8	44.9	33.3
full + prag	52.8	49.8	45.8	33.8

Table 4: Average online accuracy under various settings. memorize: featurize entire utterance and logical form non-compositionally; half model: featurize the utterances with unigrams, bigrams, and skip-grams but conjoin with the entire logical form; full model: the model described in Section 3; +prag: the models above, with our online pragmatics algorithm described in Section 4. Both compositionality and pragmatics improve accuracy.

Pragmatics. Next, we study the effect of pragmatics on online accuracy. Figure 2 shows that modeling pragmatics helps successful players (e.g., top 10 by number of scrolls) who use precise and consistent languages. Interestingly, our pragmatics model did not help and can even hurt the less successful players who are less precise and consistent. This is expected behavior: the pragmatics model assumes that the human is cooperative and behaving rationally. For the bottom half of the players, this assumption is not true, in which case the pragmatics model is not useful.

6 Related Work and Discussion

Our work connects with a broad body of work on grounded language, in which language is used in some environment as a means towards some goal. Examples include playing games (Branavan et al., 2009, 2010; Reckman et al., 2010) interacting with robotics (Tellex et al., 2011, 2014), and following instructions (Vogel and Jurafsky, 2010; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013) Semantic parsing utterances to logical forms, which we leverage, plays an important role in these settings (Kollar et al., 2010; Matuszek et al., 2012; Artzi and Zettlemoyer, 2013).

What makes this work unique is our new interactive learning of language games (ILLG) setting, in which a model has to learn a language from *scratch* through interaction. While online gradient descent is frequently used, for example in semantic parsing (Zettlemoyer and Collins, 2007; Chen, 2012), we using it in a truly online setting, taking one pass over the data and measuring online accuracy (Cesa-Bianchi and Lugosi, 2006).

To speed up learning, we leverage computational models of pragmatics (Jäger, 2008; Golland et al., 2010; Frank and Goodman, 2012; Smith et al., 2013; Vogel et al., 2013). The main difference is these previous works use pragmatics with a trained base model, whereas we learn the model online. Monroe and Potts (2015) uses learning to improve the pragmatics model. In contrast, we use pragmatics to speed up the learning process by capturing phenomena like mutual exclusivity (Markman and Wachtel, 1988). We also differ from prior work in several details. First, we model pragmatics in the online learning setting where we use an online update for the pragmatics model. Second, unlikely the reference games where pragmatic effects plays an important role by design, SHRDLURN is not specifically designed to require pragmatics. The improvement we get is mainly due to players trying to be consistent in their language use. Finaly, we treat both the utterance and the logical forms as featurized compositional objects. Smith et al. (2013) treats utterances (i.e. words) and logical forms (i.e. objects) as categories; Monroe and Potts (2015) used features, but also over flat categories.

Looking forward, we believe that the ILLG setting is worth studying and has important implications for natural language interfaces. Today, these systems are trained once and deployed. If these systems could quickly adapt to user feedback in real-time as in this work, then we might be able to more readily create systems for resource-poor languages and new domains, that are customizable and improve through use.

Acknowledgments

DARPA Communicating with Computers (CwC) program under ARO prime contract no. W911NF-15-1-0462. The first author is supported by a NSERC PGS-D fellowship. In addition, we thank Will Monroe, and Chris Potts for their insightful comments and discussions on pragmatics.

Reproducibility

All code, data, and experiments for this paper are available on the CodaLab platform:
`https://worksheets.codalab.org/worksheets/0x9fe4d080bac944e9a6bd58478cb05e5e`
The client side code is here:
`https://github.com/sidaw/shrdlurn/tree/acl16-demo`
and a demo: `http://shrdlurn.sidaw.xyz`

References

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

J. Berant, A. Chou, R. Frostig, and P. Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*.

S. Branavan, H. Chen, L. S. Zettlemoyer, and R. Barzilay. 2009. Reinforcement learning for

mapping instructions to actions. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL-IJCNLP)*. pages 82–90.

S. Branavan, L. Zettlemoyer, and R. Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Association for Computational Linguistics (ACL)*. pages 1268–1277.

N. Cesa-Bianchi and G. Lugosi. 2006. *Prediction, learning, and games*. Cambridge University Press.

D. L. Chen. 2012. Fast online lexicon learning for grounded language acquisition. In *Association for Computational Linguistics (ACL)*.

D. L. Chen and R. J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Association for the Advancement of Artificial Intelligence (AAAI)*. pages 859–865.

J. Duchi, E. Hazan, and Y. Singer. 2010. Adaptive subgradient methods for online learning and stochastic optimization. In *Conference on Learning Theory (COLT)*.

M. Frank and N. D. Goodman. 2012. Predicting pragmatic reasoning in language games. *Science* 336:998–998.

H. Giles. 2008. *Communication accommodation theory*. Sage Publications, Inc.

D. Golland, P. Liang, and D. Klein. 2010. A game-theoretic approach to generating spatial descriptions. In *Empirical Methods in Natural Language Processing (EMNLP)*.

N. Goodman and D. Lassiter. 2015. *Probabilistic Semantics and Pragmatics: Uncertainty in Language and Thought*. The Handbook of Contemporary Semantic Theory, 2nd Edition Wiley-Blackwell.

H. P. Grice. 1975. Logic and conversation. *Syntax and semantics* 3:41–58.

M. E. Ireland, R. B. Slatcher, P. W. Eastwick, L. E. Scissors, E. J. Finkel, and J. W. Pennebaker. 2011. Language style matching predicts relationship initiation and stability. *Psychological Science* 22(1):39–44.

G. Jäger. 2008. Game theory in semantics and pragmatics. Technical report, University of Tübingen.

T. Kollar, S. Tellex, D. Roy, and N. Roy. 2010. Grounding verbs of motion in natural language commands to robots. In *International Symposium on Experimental Robotics (ISER)*.

T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Empirical Methods in Natural Language Processing (EMNLP)*. pages 1223–1233.

P. Liang, M. I. Jordan, and D. Klein. 2011. Learning dependency-based compositional semantics. In *Association for Computational Linguistics (ACL)*. pages 590–599.

E. Markman and G. F. Wachtel. 1988. Children's use of mutual exclusivity to constrain the meanings of words. *Cognitive Psychology* 20:125–157.

C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox. 2012. A joint model of language and perception for grounded attribute learning. In *International Conference on Machine Learning (ICML)*. pages 1671–1678.

W. Monroe and C. Potts. 2015. Learning in the Rational Speech Acts model. In *Proceedings of 20th Amsterdam Colloquium*.

P. Pasupat and P. Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Association for Computational Linguistics (ACL)*.

H. Reckman, J. Orkin, and D. Roy. 2010. Learning meanings of words and constructions, grounded in a virtual game. In *Conference on Natural Language Processing (KONVENS)*.

N. J. Smith, N. D. Goodman, and M. C. Frank. 2013. Learning and using language via recursive pragmatic reasoning about other agents. In *Advances in Neural Information Processing Systems (NIPS)*.

S. Tellex, R. Knepper, A. Li, D. Rus, and N. Roy. 2014. Asking for help using inverse semantics. In *Robotics: Science and Systems (RSS)*.

S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

A. Vogel, M. Bodoia, C. Potts, and D. Jurafsky. 2013. Emergence of gricean maxims from

multi-agent decision theory. In *North American Association for Computational Linguistics (NAACL)*. pages 1072–1081.

A. Vogel and D. Jurafsky. 2010. Learning to follow navigational directions. In *Association for Computational Linguistics (ACL)*. pages 806–814.

T. Winograd. 1972. *Understanding Natural Language*. Academic Press.

L. Wittgenstein. 1953. *Philosophical Investigations*. Blackwell, Oxford.

Y. W. Wong and R. J. Mooney. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Association for Computational Linguistics (ACL)*. pages 960–967.

L. S. Zettlemoyer and M. Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Uncertainty in Artificial Intelligence (UAI)*. pages 658–666.

L. S. Zettlemoyer and M. Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL)*. pages 678–687.

Finding Non-Arbitrary Form-Meaning Systematicity Using String-Metric Learning for Kernel Regression

E. Darío Gutiérrez
UC San Diego
edg@icsi.berkeley.edu

Roger Levy
MIT
rplevy@mit.edu

Benjamin K. Bergen
UC San Diego
bkbergen@ucsd.edu

Abstract

Arbitrariness of the sign—the notion that the forms of words are unrelated to their meanings—is an underlying assumption of many linguistic theories. Two lines of research have recently challenged this assumption, but they produce differing characterizations of non-arbitrariness in language. Behavioral and corpus studies have confirmed the validity of localized form-meaning patterns manifested in limited subsets of the lexicon. Meanwhile, global (lexicon-wide) statistical analyses instead find diffuse form-meaning systematicity across the lexicon as a whole.

We bridge the gap with an approach that can detect both local and global form-meaning systematicity in language. In the kernel regression formulation we introduce, form-meaning relationships can be used to predict words' distributional semantic vectors from their forms. Furthermore, we introduce a novel metric learning algorithm that can learn weighted edit distances that minimize kernel regression error. Our results suggest that the English lexicon exhibits far more global form-meaning systematicity than previously discovered, and that much of this systematicity is focused in localized form-meaning patterns.

1 Introduction

Arbitrariness of the sign refers to the notion that the phonetic/orthographic forms of words have no relationship to their meanings (de Saussure, 1916). It is a foundational assumption of many theories of language comprehension, production, acquisition, and evolution. For instance, Hockett's (1960) influential enumeration of the design features of human language ascribes a central role to arbitrariness in enabling the combination and recombination of phonemic units to create new words. Gasser (2004) uses simulations to show that for large vocabularies, arbitrary form-meaning mappings may provide an advantage in acquisition. Meanwhile, modular theories of language comprehension rely upon the duality of patterning to support the independence of the phonetic and semantic aspects of language comprehension (Levelt et al., 1999). Quantifying the extent to which the arbitrariness principle actually holds is important for understanding how language works.

Language researchers have long noted exceptions to arbitrariness. Most of these are patterns that occur in some relatively localized subset of the lexicon. These patterns are sub-morphemic because, unlike conventional morphemes, they cannot combine reliably to produce new words. *Phonaesthemes* (1930) are one example. A phonaestheme is a phonetic cluster that recurs in many words that have related meanings. One notable phonaestheme is the onset *gl-*, which occurs at the beginning of at least 38 English words relating to vision: *glow*, *glint*, *glaze*, *gleam*, etc. (Bergen, 2004). At least 46 candidate phonaesthemes have been posited in the linguistics literature, according to a list compiled by Hutchins (1998). *Iconicity* is another violation of arbitrariness that can lead to non-arbitrary local regularities. Iconicity occurs when the form of a word is transparently motivated by some perceptual aspect of its referent. Consequently, when several referents share perceptual features, their associated word-forms would tend to be similar as well (to the extent that they are iconic). For instance, Ohala (1984) conjectures that vowels with high acoustic frequency tend to associate with smaller items while vowels with low acoustic frequency

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2379–2388,
Berlin, Germany, August 7-12, 2016.

tend to associate with larger items, due to the experiential link between vocalizer size and frequency. Systematic iconicity is also manifested in sets of onomatopoeic words that echo similar sounds (e.g., *clink*, *clank*). Although these exceptions to non-arbitrariness differ, in each case, specific form-meaning relationships emerge in a subset of the lexicon. We will refer to all such specific localized form-meaning patterns as *phonosemantic sets*.

In recent decades, behavioral and corpus studies have empirically confirmed the psychological reality and statistical reliability of many phonosemantic sets that had previously been identified by intuition and observation. Various candidate phonaesthemes have significant effects on reaction times during language processing tasks (Hutchins, 1998; Magnus, 1998; Bergen, 2004). Sagi and Otis (2008) test the statistical significance of the 46 candidates in Hutchins's (1998) list, and find that 27 of them exhibit more within-category distributional semantic coherence than expected by chance. These results have been replicated using other corpora and distributional semantic models (Abramova et al., 2013). Klink (2000) shows that sound-symbolic attributes such as those proposed by Ohala (1984) are associated with human judgments about nonwords' semantic attributes, such as smallness or beauty. Using a statistical corpus analysis and WordNet semantic features, Monaghan et al. (2014a) examine a similar hypothesis space of sound-symbolic phonological and semantic attributes, and reach similar conclusions.

While these localized studies support the existence of some islands of non-arbitrariness in language, their results do not address how pervasive non-arbitrariness is at the global level—that is, in the lexicon of a language as a whole. After all, some seemingly non-arbitrary local patterns can be expected to emerge merely by chance. How can we measure whether local phonosemantic patterning translates into global *phonosemantic systematicity*–that is, strong, non-negligible lexicon-wide non-arbitrariness? Shillcock et al. (2001) introduce the idea of measuring phonosemantic systematicity by analyzing the correlation between phonological edit distances and distributional semantic distances. In a lexicon of monomorphemic and monosyllabic English words, they find a small but statistically significant correlation between these two distance measures. Monaghan et al. (2014b) elaborate on this methodology, showing that the statistical effect is robust to different choices of form-distance and semantic-distance metrics. They also look at the effect of leaving out each word in the lexicon on the overall correlation measure; from this, they derive a phonosemantic systematicity measure for each word. Interestingly, they find that systematicity is diffusely distributed across the words in English in a pattern indistinguishable from random chance. Hence, they conclude that "systematicity in the vocabulary is not a consequence of small clusters of sound symbolism." This line of work provides a proof-of-concept that it is possible to detect the phonosemantic systematicity of a language, and confirms that English exhibits significant phonosemantic systematicity.

Broadly speaking, both the localized tests of individual phonosemantic sets and the global analyses of phonosemantic systematicity challenge the arbitrariness of the sign. However, they attribute responsibility for non-arbitrariness differently. The local methods reveal dozens of specific phonosemantic sets that have strong, measurable behavioral effects and statistical signatures in corpora. Meanwhile, the global methods find small and diffuse systematicity. How can we reconcile this discrepancy?

Original Contributions. We attempt to bridge the gap with a new approach that builds off of previous lexicon-wide analyses, making two innovations. The first addresses the concern that the lexicon-wide methods currently in use may not be well suited to finding local regularities such as phonosemantic sets, because they make the assumption that systematicity exists only in the form of a global correlation between distances in form-space and distances in meaning-space. Instead, we model the problem using kernel regression, a non-parametric regression model. Crucially, in kernel regression the prediction for a point is based on the predictions of neighboring points; this enables us to conduct a global analysis while still capturing local, neighborhood effects. As in previous work, we represent word-forms by their orthographic strings, and word-meanings by their semantic vector representations as produced by a distributional semantic vector space model. The goal of the regression is then to learn a mapping from string-valued predictor variables to vector-valued target variables that minimizes regression

error in the vector space. Conveniently, our model allows us to produce predictions of the semantic vectors associated with both words and nonwords.

Previous work may also underestimate systematicity in that it weights all edits (substitutions, insertions, and deletions) equally in determining edit distance. A priori, there is no reason to believe this is the case—indeed, the work on individual phonosemantic sets suggests that some orthographic/phonetic attributes are more important than others for non-arbitrariness. To address this, we introduce String-Metric Learning for Kernel Regression (SMLKR), a metric-learning algorithm that is able to learn a weighted edit distance metric that minimizes the prediction error in kernel regression.

We find that SMLKR enables us to recover more systematicity from a lexicon of monomorphemic English words than reported in previous global analyses. Using SMLKR, we propose a new measure of per-word phonosemantic systematicity. Our analyses using this systematicity measure indicate that specific phonosemantic sets do contribute significantly to the global phonosemantic systematicity of English, in keeping with previous local-level analyses. Finally, we evaluate our systematicity measure against human judgments, and find that it accords with raters' intuitions about what makes a word's form well suited to its meaning.

2 Background & Related Work

2.1 Previous Approaches to Finding Lexicon-Wide Systematicity

Measuring Form, Meaning, and Systematicity. To our knowledge, all previous lexicon-level analyses of phonosemantic systematicity have used variations of the method of Shillcock et al. (2001). The inputs for this method are form-meaning tuples $(\mathbf{y}_i, s_i)$ for each word i in the lexicon, where $\mathbf{y}_i$ is the vector representation of the word in a distributional semantic model, and s_i is the string representation of the word (phonological, phonemic, or orthographic). Semantic distances are measured as cosine distances between the vectors of each pair of words. Shillcock et al. (Shillcock et al., 2001) and Monaghan et al. (Monaghan et al., 2014b) measure form-distances in terms of edit distance between each pair of strings. In addition Monaghan et al. (2014b) and Tamariz (2006) study distance measures based on a selected set binary phonological features, with similar results. Phonosemantic systematicity is then measured as the correlation between all the pairwise semantic distances and all the pairwise string distances.

Hypothesis Testing. In this line line of work, statistical significance of the results is assessed using the Mantel test, a permutation test of the correlation between two sets of pairwise distances (Mantel, 1967). The test involves randomly shuffling the assignments of semantic vectors to wordstrings in the lexicon. We can think of each form-meaning shuffle as a member of the set of all possible lexicons. Next, the correlation between the semantic distances and the string distances is computed under each reassignment. An empirical p-value for the true lexicon is then derived by performing many shufflings, and comparing the correlation coefficients measured under the shuffles to the correlation coefficient measured in the true lexicon. Under the null hypothesis that form-meaning assignments are arbitrary, the probability of observing a form-meaning correlation of at least the magnitude actually observed in the true lexicon is asymptotically equal to the proportion of reassignments that produce greater correlations than the true lexicon.

Previous Findings. Shillcock et al. (2001) find a statistically significant correlation between semantic and phonological edit distances in a lexicon of the 1733 most frequent monosyllabic monomorphemic words in the BNC. Tamariz (2008) extends these results to Spanish data, looking only at words with one of three consonant-vowel (CV) structures (CVCV, CVCCV, and CVCVCV). (2001), Monaghan et al. (2014b) derive a list of 5138 monomorphemic monosyllabic words and a list of 5604 monomorphemic polysyllabic from the CELEX database (Baayen et al., 1996), and find significant form-meaning correlations in both.

2.2 Kernel Regression

In contrast to previous studies, we study form-meaning systematicity using a kernel regression framework. Kernel regression is a nonparametric supervised learning technique that is able to learn highly nonlinear relationships between predictor variables and target variables. Rather than assuming any particular parametric relationship between the predictor and target variables, kernel regression assumes only that the value of the target vari-

able is a smooth function of the value of the predictors. In other words, given a new point in predictor space, the value of the target at that point can reasonably be estimated by the value of the targets at points that are nearby in the predictor space. In this way, kernel regression is analogous to an exemplar model. We performed kernel regression on our lexicon using the Nadaraya-Watson estimator (Nadaraya, 1964). Given a data set $\mathcal{D}$ of vector-valued predictor variables $\{\mathbf{x}_i\}_{i=1}^N$, and targets $\{\mathbf{y}_i\}_{i=1}^N$, the Nadaraya-Watson estimator of the target for sample i is

$$\hat{\mathbf{y}}_i = \hat{\mathbf{y}}(\mathbf{x}_i) = \frac{\sum_{j \neq i} k_{ij} \mathbf{y}_j}{\sum_{j \neq i} k_{ij}}, \quad (1)$$

where k_{ij} is the *kernel* between point i and point j. A commonly used kernel is the exponential kernel:

$$k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-d(\mathbf{x}_i, \mathbf{x}_j)/h),$$

where $d(\cdot, \cdot)$ is a distance metric and h is a *bandwidth* that determines the radius of the effective neighborhood around each point that contributes to its estimate. For our purposes we use the Levenshtein string edit distance metric (Levenshtein, 1966). The Levenshtein edit distance between two strings is the minimum number of edits needed to transform one string into the other, where an edit is defined as the insertion, deletion, or substitution of a single character. Using this edit distance and semantic vectors derived from a distributional semantic model, the Nadaraya-Watson estimator can estimate the position in the semantic vector space for each word in the lexicon. The exponential edit distance kernel has been useful for modeling behavior in many tasks involving word similarity and neighborhood effects; see, for example the Generalized Context Model (Nosofsky, 1986), which has been applied to word identification, recognition, and categorization, to inflectional morphology, and to artificial grammar learning (Bailey and Hahn, 2001).

2.3 Metric Learning for Kernel Regression

In kernel regression, the bandwidth h of the kernel function must be fine-tuned by testing out many different bandwidths. Moreover, for many tasks there is no reason to assume that all of the dimensions of a vector-valued predictor are equally important. This is problematic for conventional kernel regression, as the quality of its predictions is wholly reliant on the appropriateness of the given distance metric.

Weinberger and Tesauro (2007) introduce metric learning for kernel regression (MLKR), an algorithm that can learn a task-specific Mahalanobis (i.e., weighted Euclidean) distance metric over a real-vector-valued predictor space, in which small distances between two vectors imply similar target values. They note that this metric induces a kernel function whose parameters are set entirely from the data. Specifically, MLKR can learn a weight matrix W for a Mahalanobis metric that optimizes the leave-one out mean squared error of kernel regression (MSE), defined as:

$$\mathcal{L}(\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(\hat{\mathbf{y}}_i, \mathbf{y}_i) = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|_2^2,$$

where $\hat{\mathbf{y}}_i$ is estimated using $\hat{\mathbf{y}}_j$ for all $i \neq j$, as in Eq. 1.

In MLKR, the weighted distance metric is learned using stochastic gradient descent. As an added benefit, MLKR is implicitly able to learn an appropriate kernel bandwidth.

3 String-Metric Learning for Kernel Regression (SMLKR)

Our novel contribution is an extension of MLKR to situations where the predictor variables are not real-valued vectors, but strings, and the distance metric we wish to learn is a weighted Levenshtein edit distance. Vector-valued representations of the strings themselves would only approximately preserve edit distance. Fortunately, it turns out that we do not need vector-valued representations of the strings at all. Define the *minimum edit-distance path* as the smallest-length sequence of edits that is needed to transform one string into another. Observe that the weighted edit distance between two strings s_i and s_j can be represented as the weighted sum of all the edits that must take place to transform one string into the other along the minimum edit-distance path (Bellet et al., 2012). In turn, these edits can be represented by a vector $\boldsymbol{\nu}_{ij}$ constructed as in Fig 1, while the weights can be represented by a vector $\mathbf{w} = (w_1, ..., w_M)^T$:

$$d_{WL}(s_i, s_j) = \sum_{m=1}^{M} w_m \nu_{ijm} = \mathbf{w}^T \boldsymbol{\nu}_{ij}.$$

Figure 1: Each element in $\boldsymbol{\nu}_{ij}$ (the vector at left) represents a type of edit. The entry ν_{ijm} represents the number of edits of type m that occur as string s_i (*boot*) is transformed into string s_j (*bee*).

Each entry of $\boldsymbol{\nu}_{ij}$ corresponds to a particular type of edit operation (e.g., substitution of character a for character b). The value assigned to each entry is the count of the total number of times that the corresponding edit operation must be applied to achieve transformation of string i to string j along the minimum edit-distance path.

We note that $\boldsymbol{\nu}_{ij}$ does not admit a unique representation, since there are multiple ways to transform one string to another in the same number of edits, using different edit operations. However, we adopt the convention that some class of edit operations always takes priority over another—e.g., that deletions always occur before substitutions. This then enables us to specify $\boldsymbol{\nu}_{ij}$ uniquely. We also adopt the convention that the weights for edit operations are symmetric—e.g., that the weight for substituting character a for character b is the same as the weight for substituting character b for character a, so we represent every such pair of edit operations by a single entry in $\boldsymbol{\nu}_{ij}$.

As in MLKR, our goal is to minimize the leave-one-out MSE,[1] where $k_{ij} = e^{-\mathbf{w}^T\boldsymbol{\nu}_{ij}}$. The gradient of the regression error for MSE is

$$\frac{\partial\mathcal{L}(\mathcal{D})}{\partial\mathbf{w}} = \frac{2}{N}\sum_{i=1}^{N}(\mathbf{y}_i - \hat{\mathbf{y}}_i)\frac{\partial\hat{\mathbf{y}}_i}{\partial\mathbf{w}}$$

where

$$\frac{\partial\hat{\mathbf{y}}_i}{\partial\mathbf{w}} = \frac{\sum_{j\neq i}(\mathbf{y}_j - \hat{\mathbf{y}}_i)^T k_{ij}\boldsymbol{\nu}_{ij}}{\sum_{j\neq i}k_{ij}}.$$

Using this exact gradient, we can find the edit weights that minimize the loss function. We wish to constrain the weights to be non-negative, since weighted edit distance only makes sense with nonnegative weights. Thus, to minimize the loss we use the limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm for box constraints (L-BFGS-B) (Byrd et al., 1995), a quasi-Newton method that allows bounded optimization. We made a Python implementation of SMLKR available at `http://bit.ly/25Hidqg/`.

[1] We attained similar results minimizing mean cosine error. The gradient for mean cosine error is

$$\frac{\partial\mathcal{L}(\mathcal{D})}{\partial\mathbf{w}} = \frac{1}{N}\sum_{i=1}^{N}\frac{(\|\hat{\mathbf{y}}_i\|\mathbf{y}_i - \mathcal{L}(\mathbf{y}_i,\hat{\mathbf{y}}_i)\hat{\mathbf{y}}_i)}{\|\hat{\mathbf{y}}_i\|^2}\frac{\partial\hat{\mathbf{y}}_i}{\partial\mathbf{w}}.$$

4 Experimental Setup

4.1 Data

Lexicon. A principal concern is the possibility that our models may detect morphemes rather than sub-morphemic units. To minimize this concern, we adopted an approach similar to that of Shillcock et al. (2001), of training our model only on monomorphemic words. Monomorphemic words were selected by cross-referencing the morphemic analyses contained in the CELEX lexical database (Baayen et al., 1996) with the morphemic analyses contained in the etymologies of the Oxford English Dictionary Online (`http://www.oed.com`). Then, we went through the filtered list and removed any remaining polymorphemic words as well as place names, demonyms, spelling variants, and proper nouns. Finally, words that were not among the 40,000 most frequent non-filler word types in the corpus were excluded. The final lexicon was composed of 4,949 word types.

Corpus and Semantic Model. The corpus we used to train our semantic model is a concatenation of the UKWaC, BNC, and Wikipedia corpora (Ferraresi et al., 2008; BNC Consortium, 2007; Parker et al., 2011). We trained our vector-space model on this corpus using the Word2Vec (Mikolov et al., 2013), as instantiated in the GENSIM package (Řehůřek and Sojka, 2010) for Python using default parameters. We produced 100-dimensional word-embedding vectors using the SkipGram algorithm of Word2Vec and normalized the 100-dimensional vector for each word so that its Euclidean norm was equal to 1.

4.2 Training

We trained SMLKR on the 100-dimensional Word2Vec embeddings using L-BGFS-B, and placing non-negativity constraints on the weights $\mathbf{w}$. We let SMLKR run until convergence, as de-

termined by the following criterion:

$$\frac{|\mathcal{L}^{(k-1)} - \mathcal{L}^{(k)}|}{\max(|\mathcal{L}^{(k-1)}|, |\mathcal{L}^{(k)}|)} = \epsilon$$

where $\mathcal{L}^{(k)}$ is the loss at the k^{th} iteration of learning, and we set $\epsilon = 2 \times 10^{-8}$. We randomly initialized the L-BGFS-B algorithm 10 times to avoid poor local minima, and kept the solution with the lowest loss.

5 Experiments

5.1 Model Analysis

Weighted Edit Distance Reveals More Non-Arbitrariness. We first assessed whether the structure found by kernel regression could arise merely by arbitrary, random pairings of form and meaning (i.e., strings and semantic vectors). We adopt a Monte Carlo testing procedure similar to the Mantel test of §2.1. We first randomly shuffled the assignment of the semantic vectors of all the words in the lexicon. We then trained SMLKR on the shuffled lexicon just as we did on the true lexicon. We measured the mean squared error of the SMLKR prediction. Out of 1000 reassignments, none produced a prediction error as small as the prediction error in the true lexicon (i.e., empirical p-value of $p < .001$).

For comparison, we analyzed our corpus using the correlation method of Monaghan et al. (2014b). In our implementation, we measured the correlation between the pairwise cosine distances produced by Word2Vec and pairwise orthographic edit distances for all pairs of words in our lexicon. The correlation between the Word2Vec semantic distances and the orthographic edit distances in our corpus was $r = 0.0194$, similar to the correlation reported by Monaghan et al. of $r = 0.016$ between the phoneme edit distances and the semantic distances in the monomorphemic English lexicon. We also looked at the correlation between the weighted edit distances produced by SMLKR and the Word2Vec semantic distances. The correlation between these distances was $r = 0.0464$; thus, the weighted edit distance captures more than 5.7 times as much variance as the unweighted edit distance. Further, using the estimated semantic vectors produced by the SMLKR model, we can actually produce new estimates of the semantic distances between the words. The correlation between these estimated semantic distances and the true semantic distances was $r = 0.1028$, revealing much more systematicity than revealed by the simple linear correlation method. The Mantel test with 1,000 permutations produced significant empirical p-values for all correlations ($p < .001$).

Systematicity Not Evenly Distributed Across Lexicon. What could be accounting for the higher degree of systematicity detected with SMLKR? Applying a more expressive model could result in a better fit simply because incidental but inconsequential patterns are being captured. Conversely, SMLKR could be finding phonosemantic sets which the correlation method of Monaghan et al. (Monaghan et al., 2014b) is unable to detect. We investigated further by determining what was driving the better fit produced by SMLKR. Monaghan et al. measure per-word systematicity as the change in the lexicon-wide form-meaning correlation that results from removing the word from the lexicon. The more the correlation decreases from removing the word, the more systematic the word is, according to this measure. They compared the distribution of this systematicity measure across the words in the lexicon to the distribution of systematicity in lexicons with randomly shuffled form-meaning assignments, and found that the null hypothesis that the distributions were identical could not be rejected. From this, they conclude that the observed systematicity of the lexicon is not a consequence only of small pockets of sound symbolism, but is rather a feature of the mappings from sound to meaning across the lexicon as a whole. However, it is possible that their methods may not be sensitive enough to find localized phonosemantic sets.

We developed our own measure of per-word systematicity by measuring the per-word regression error of the SMLKR model. We presume words with lower regression errors to be more systematic. A list of the words with the lowest per-word regression error in our corpus can be found in Table 1. Notably, many of these words, such as *fluff*, *flutter*, and *flick*, exhibit word beginnings or word endings that have been previously identified as phonaesthemes (Hutchins, 1998; Otis and Sagi, 2008). Others exhibit regular onomatopoeia, such as *clang* and *croak*.

We decided to investigate the distribution of systematicity across two-letter word-beginnings in our lexicon using a permutation test. The goal of the permutation test is to estimate a p-value for the

SMLKR	Correlation	Random
gurgle	emu	tunic
tingle	nexus	decay
hoop	asylum	skirmish
chink	ethic	scroll
swirl	odd	silk
ladle	slime	prom
flick	snare	knob
wobble	scarlet	havoc
tangle	deem	irate
knuckle	balustrade	veer
glitter	envoy	wear
twig	scrape	phone
fluff	essay	surgeon
rasp	ambit	hiccup
quill	echo	bowel
flutter	onus	sack
whirl	exam	lens
croak	pirouette	hovel
squeal	kohl	challenge
clang	chandelier	box

Table 1: *Left*: Most systematic words according to SMLKR. *Center*: Most systematic words according to the leave-one-out correlation method proposed by Monaghan et al. (2014b). *Right*: Randomly generated list for comparison.

likelihood that each set of words sharing a word beginning would exhibit the mean regression error it exhibits, if systematicity is randomly distributed across the lexicon. For each set $\mathcal{S}_\omega$ of words with word-beginning ω, we measured the mean SMLKR regression error of the words in $\mathcal{S}_\omega$. To get an empirical p-value for each $\mathcal{S}_\omega$ with cardinality greater than 5 (i.e., more than 5 word tokens), we randomly chose 10^5 sets of words in the lexicon with the same cardinality, and measured the mean SMLKR regression error for each of these random sets. If r of the randomly assembled sets had a lower mean regression error than $\mathcal{S}_\omega$ did, we assign an empirical p-value of $\frac{r}{10^5}$ to $\mathcal{S}_\omega$. A histogram of empirical p-values is in Fig. 2. From the figure, it seems clear that the p-values are not uniformly distributed; instead, an inordinate number of word-beginnings exhibit mean errors that are unlikely to occur if error is distributed arbitrarily across word-beginnings.

We can confirm this observation statistically. On the assumption that systematicity is arbitrarily distributed across word-beginnings, the empirical p-values of the permutation test should approximately conform to a Unif$(0, 1)$ distribution. We can test this hypothesis using a χ^2 test on the negative logarithms of the p-values (Fisher, 1932). Using this test, we reject the hypothesis that the p-values are uniformly distributed with $p < .0001$ ($\chi^2_{156} = 707.8$). The particular word-beginnings

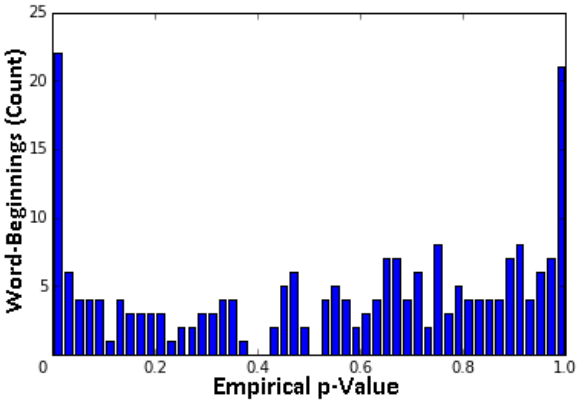


Figure 2: Histogram showing distribution of systematicity across two-letter word-beginnings, as measured by permutation-test empirical p-value.

Onset	p-value
fl-	$< 1 \times 10^{-4}$
sn-	$< 1 \times 10^{-4}$
sw-	$< 1 \times 10^{-4}$
tw-	$< 1 \times 10^{-4}$
gl-	1×10^{-3}
sl-	1×10^{-3}
bu-	1×10^{-3}
mu-	2×10^{-3}
wh-	2×10^{-3}
sc-/sk-	3×10^{-3}

Table 2: Word-beginnings with mean errors lower than predicted by random distribution of errors across lexicon. **Bold** are among the phonaesthemes identified by Hutchins (1998). *Italics* were identified by Otis and Sagi (2008).

with statistically significant empirical p-values ($p < .05$ after Benjamini-Hochberg (1995) correction for multiple comparisons) are in Table 2. Eight of these ten features are among the 18 two-letter onsets posited to be phonaesthemes by Hutchins (1998). For comparison, Otis and Sagi (2008) identified eight of Hutchins's 18 two-letter word-beginning candidate phonaesthemes (and 12 two-letter word-beginnings overall) as statistically significant, though they restricted their hypothesis space to only 50 pre-specified word-beginnings and word-endings. We are able to identify just as many candidate phonaesthemes, but with a much less restricted hypothesis space of candidates (225 rather than the 50 in Otis and Sagi's analysis) and with a general model not specifically attuned to finding phonaesthemes in particular, but rather systematicity in general.

5.2 Behavioral Evaluation of Systematicity Measure

We empirically tested whether the systematicity measure based on SMLKR regression error accords with naïve human judgments about how well-suited a word's form is to its meaning (its "phonosemantic feeling") (Stefanowitsch, 2002). We recruited 60 native English-speaking participants through Mechanical Turk, and asked them to judge the phonosemantic feeling of the 60 words in Table 1 on a sliding scale from 1 to 5.[2] We used Cronbach's α to measure inter-annotator reliability at $\alpha = 0.96$, indicating a high degree of inter-annotator reliability (Cronbach, 1951; George, 2000). The results showed that the words in the SMLKR list were rated higher for phonosemantic feeling than the words in the Correlation and Random lists. We fit a parametric linear mixed-effects model to the phonosemantic feeling judgments (Baayen et al., 2008), as implemented in the `lme4` library for R. As fixed effects, we entered the list identity (SMLKR, Correlation, Random), the word length, and the log frequency of the word in our corpus. Our random effects structure included a random intercept for word, and random subject slopes for all fixed effects, with all correlations allowed (a "maximal" random-effects structure (Barr et al., 2013)). Including list identity in the maximal mixed-effects model significantly improved model fit ($\chi^2_{11} = 126.08$, $p < 10^{-6}$). Post-hoc analysis revealed that the SMLKR list elicited average suitability judgments that were 0.49 points higher than the Random list ($p < 10^{-6}$) and 0.59 points higher than the Correlation list ($p < 10^{-6}$). Post-hoc analysis did not find a significant difference in suitability judgments between the Random and Correlation lists ($p > .16$).[3]

6 Conclusion

In this paper, we proposed SMLKR, a novel algorithm that can learn weighted string edit distances that minimize kernel regression error. We succeed in applying this algorithm to the problem of finding form-meaning systematicity in the monomorphemic English lexicon. Our algorithm offers improved global predictions of word-meaning given word-form at the lexicon-wide level. We show that this improvement seems related to localized pockets of form-meaning systematicity such as those previously uncovered in behavioral and corpus analyses. Unlike previous lexicon-wide analyses, we find that form-meaning systematicity is not randomly distributed throughout the English lexicon. Moreover, the measure of systematicity that we compute using SMLKR accords significantly with human raters' judgments about form-meaning correspondences in English.

Future work may investigate to what extent the SMLKR model can predict human intuitions about form-meaning systematicity in language. We do not know, for instance, if our model can predict human semantic judgments of novel words that have never been encountered. This is a question that has received attention in the market research literature, where new brand names are tested for the emotions they elicit (Klink, 2000). We would also like to investigate the degree to which our statistical model predicts the behavioral effects of phonosemantic systematicity during human semantic processing that have been reported in the psycholinguistics literature. Our model makes precise quantitative predictions that should allow us to address these questions.

While developing our model on preliminary versions of the monomorphemic lexicon, we noticed that the model detected high degrees of systematicity in words with suffixes such as *-ate* and *-tet* (e.g., *quintet*, *quartet*). We removed such words in the final analysis since they are polymorphemic, but this observation suggests that our algorithm may have applications in unsupervised morpheme discovery.

Finally, we would like to test our model using other representations of word-form and word-meaning. We chose to use orthographic rather than phonetic representations of words because of the variance in pronunciation present in the dialects of English that are manifested in our corpus. However, it would be interesting to verify our results in a phonological setting, perhaps using a monodialectal corpus. Moreover, previous local-level analyses suggest that systematicity seems to be concentrated in word-beginnings and word-

[2] Participants were given the following guidance: "Your job is to decide how well-suited each word is to what it means. This is known as the 'phonosemantic feeling.' Basically, most people feel like some of the words in their native language sound right, given what they mean." Full instructions and experiment available at `http://goo.gl/Z6Lzlp`

[3] Post hoc analyses were produced by comparing the items in only two of the lists at a time, and fitting the same mixed-effects model as above.

endings. Thus, it may be worthwhile to augment the representation of edit distance in our model by making it context-sensitive. Future work could also test whether a more interpretable meaning-space representation such as that provided by binary WordNet feature vectors reveals patterns of systematicity not found using a distributional semantic space.

Acknowledgments

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number ACI-1053575.

References

Ekaterina Abramova, Raquel Fernández, and Federico Sangati. 2013. Automatic labeling of phonesthemic senses. In *Proeedings of the 35th Annual Conference of the Cognitive Science Society*, volume 35. Cognitive Science Society.

R Harald Baayen, Richard Piepenbrock, and Léon Gulikers. 1996. CELEX2 (CD-ROM).

R. Harald Baayen, Douglas J. Davidson, and Douglas M. Bates. 2008. Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412.

Todd M. Bailey and Ulrike Hahn. 2001. Determinants of wordlikeness: Phonotactics or lexical neighborhoods? *Journal of Memory and Language*, 44(4):568–591.

Dale J. Barr, Roger Levy, Christoph Scheepers, and Harry J. Tily. 2013. Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3):255–278.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. 2012. Good edit similarity learning by loss minimization. *Machine Learning*, pages 5–35.

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300.

Benjamin K. Bergen. 2004. The psychological reality of phonaesthemes. *Language*, pages 290–311.

BNC Consortium. 2007. British National Corpus, Version 3 BNC XML edition.

Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, 16(5):1190–1208.

Lee J. Cronbach. 1951. Coefficient alpha and the internal structure of tests. *Psychometrika*, 16(3):297–334.

Ferdinand de Saussure. 1916. *Course in General Linguistics*. McGraw-Hill, New York.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating UKWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google*, pages 47–54.

John R. Firth. 1930. *Speech*. Benn's Sixpenny Library, London.

R.A. Fisher. 1932. *Statistical methods for research workers*. Oliver and Boyd, London.

Michael Gasser. 2004. The origins of arbitrariness in language. In *Proceedings of the 26th Annual Conference of the Cognitive Science Society*, volume 26, pages 4–7.

Darren George. 2000. *SPSS for Windows Step by Step: A Simple Guide and Reference, 11.0 Update (4th ed.)*. Allyn & Bacon, London.

Charles F. Hockett. 1960. The origin of speech. *Scientific American*, 203:88–96.

Sharon Suzanne Hutchins. 1998. *The psychological reality, variability, and compositionality of English phonesthemes*. Ph.D. thesis, Emory University, Atlanta.

Richard R. Klink. 2000. Creating brand names with meaning: The use of sound symbolism. *Marketing Letters*, 11(1):5–20.

Willem J.M. Levelt, Ardi Roelofs, and Antje S. Meyer. 1999. A theory of lexical access in speech production. *Behavioral and Brain Sciences*, 22(01):1–38.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, volume 10, pages 707–710.

Margaret Magnus. 1998. *What's in a Word? Evidence for Phonosemantics*. Ph.D. thesis, University of Trondheim, Trondheim, Norway.

Nathan Mantel. 1967. The detection of disease clustering and a generalized regression approach. *Cancer Research*, 27(2 Part 1):209–220.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Padraic Monaghan, Gary Lupyan, and Morten H Christiansen. 2014a. The systematicity of the sign: Modeling activation of semantic attributes from nonwords. In P. Bello, M. Guarini, M. McShane, and B. Scassellati, editors, *Proceedings of the 36th Annual Meeting of the Cognitive Science Society*, pages 2741–2746, Austin, TX. Cognitive Science Society.

Padraic Monaghan, Richard C. Shillcock, Morten H. Christiansen, and Simon Kirby. 2014b. How arbitrary is language? *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 369(1651).

Elizbar A. Nadaraya. 1964. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142.

Robert M. Nosofsky. 1986. Attention, similarity, and the identification–categorization relationship. *Journal of Experimental Psychology: General*, 115(1):39.

John J. Ohala. 1984. An ethological perspective on common cross-language utilization of f0 of voice. *Phonetica*, 41(1):1–16.

Katya Otis and Eyal Sagi. 2008. Phonaesthemes: A corpus-based analysis. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 65–70.

Robert Parker, David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2011. English Gigaword Fifth Edition.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May. ELRA. urlhttp://is.muni.cz/publication/884893/en.

Eyal Sagi and Katya Otis. 2008. Semantic glimmers: Phonaesthemes facilitate access to sentence meaning. In *9th Conference on Conceptual Structure, Discourse, & Language (CSDL9)*.

Richard Shillcock, Simon Kirby, Scott McDonald, and Chris Brew. 2001. Filled pauses and their status in the mental lexicon. In *ISCA Tutorial and Research Workshop (ITRW) on Disfluency in Spontaneous Speech*.

Anatol Stefanowitsch. 2002. Sound symbolism in a usage-driven model. Unpublished manuscript, Rice University, Houston, Texas, USA.

Monica Tamariz. 2006. *Exploring the adaptive structure of the mental lexicon*. Ph.D. thesis, University of Edinburgh, Edinburgh.

Monica Tamariz. 2008. Exploring systematicity between phonological and context-cooccurrence representations of the mental lexicon. *The Mental Lexicon*, 3(2):259–278.

Killian Q. Weinberger and Gerald Tesauro. 2007. Metric learning for kernel regression. In *Eleventh International Conference on Artificial Intelligence and Statistics*, pages 608–615.

Improving Hypernymy Detection with an Integrated Path-based and Distributional Method

Vered Shwartz Yoav Goldberg Ido Dagan
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
vered1986@gmail.com yoav.goldberg@gmail.com dagan@cs.biu.ac.il

Abstract

Detecting hypernymy relations is a key task in NLP, which is addressed in the literature using two complementary approaches. Distributional methods, whose supervised variants are the current best performers, and path-based methods, which received less research attention. We suggest an improved path-based algorithm, in which the dependency paths are encoded using a recurrent neural network, that achieves results comparable to distributional methods. We then extend the approach to integrate both path-based and distributional signals, significantly improving upon the state-of-the-art on this task.

1 Introduction

Hypernymy is an important lexical-semantic relation for NLP tasks. For instance, knowing that *Tom Cruise* is an *actor* can help a question answering system answer the question "which actors are involved in Scientology?". While semantic taxonomies, like WordNet (Fellbaum, 1998), define hypernymy relations between word types, they are limited in scope and domain. Therefore, automated methods have been developed to determine, for a given term-pair (x, y), whether y is an hypernym of x, based on their occurrences in a large corpus.

For a couple of decades, this task has been addressed by two types of approaches: distributional, and path-based. In distributional methods, the decision whether y is a hypernym of x is based on the distributional representations of these terms. Lately, with the popularity of word embeddings (Mikolov et al., 2013), most focus has shifted towards supervised distributional methods, in which each (x, y) term-pair is represented using some combination of the terms' embedding vectors.

In contrast to distributional methods, in which the decision is based on the *separate* contexts of x and y, path-based methods base the decision on the lexico-syntactic paths connecting the *joint* occurrences of x and y in a corpus. Hearst (1992) identified a small set of frequent paths that indicate hypernymy, e.g. *Y such as X*. Snow et al. (2004) represented each (x, y) term-pair as the multiset of dependency paths connecting their co-occurrences in a corpus, and trained a classifier to predict hypernymy, based on these features.

Using individual paths as features results in a huge, sparse feature space. While some paths are rare, they often consist of certain unimportant components. For instance, "Spelt is a species of wheat" and "Fantasy is a genre of fiction" yield two different paths: *X be species of Y* and *X be genre of Y*, while both indicating that X is-a Y. A possible solution is to generalize paths by replacing words along the path with their part-of-speech tags or with wild cards, as done in the PATTY system (Nakashole et al., 2012).

Overall, the state-of-the-art path-based methods perform worse than the distributional ones. This stems from a major limitation of path-based methods: they require that the terms of the pair occur together in the corpus, limiting the recall of these methods. While distributional methods have no such requirement, they are usually less precise in detecting a specific semantic relation like hypernymy, and perform best on detecting broad semantic similarity between terms. Though these approaches seem complementary, there has been rather little work on integrating them (Mirkin et al., 2006; Kaji and Kitsuregawa, 2008).

In this paper, we present HypeNET, an integrated path-based and distributional method for hypernymy detection. Inspired by recent progress

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2389–2398,
Berlin, Germany, August 7-12, 2016.

in relation classification, we use a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) to encode dependency paths. In order to create enough training data for our network, we followed previous methodology of constructing a dataset based on knowledge resources.

We first show that our path-based approach, on its own, substantially improves performance over prior path-based methods, yielding performance comparable to state-of-the-art distributional methods. Our analysis suggests that the neural path representation enables better generalizations. While coarse-grained generalizations, such as replacing a word by its POS tag, capture mostly syntactic similarities between paths, HypeNET captures also semantic similarities.

We then show that we can easily integrate distributional signals in the network. The integration results confirm that the distributional and path-based signals indeed provide complementary information, with the combined model yielding an improvement of up to 14 F_1 points over each individual model.[1]

2 Background

We introduce the two main approaches for hypernymy detection: distributional (Section 2.1), and path-based (Section 2.2). We then discuss the recent use of recurrent neural networks in the related task of relation classification (Section 2.3).

2.1 Distributional Methods

Hypernymy detection is commonly addressed using distributional methods. In these methods, the decision whether y is a hypernym of x is based on the distributional representations of the two terms, i.e., the contexts with which each term occurs *separately* in the corpus.

Earlier methods developed unsupervised measures for hypernymy, starting with symmetric similarity measures (Lin, 1998), and followed by directional measures based on the distributional inclusion hypothesis (Weeds and Weir, 2003; Kotlerman et al., 2010). This hypothesis states that the contexts of a hyponym are expected to be largely included in those of its hypernym. More recent work (Santus et al., 2014; Rimell, 2014) introduce new measures, based on the assumption that the most typical linguistic contexts of a hypernym are less informative than those of its hyponyms.

More recently, the focus of the distributional approach shifted to supervised methods. In these methods, the (x, y) term-pair is represented by a feature vector, and a classifier is trained on these vectors to predict hypernymy. Several methods are used to represent term-pairs as a combination of each term's embeddings vector: concatenation $\vec{x} \oplus \vec{y}$ (Baroni et al., 2012), difference $\vec{y} - \vec{x}$ (Roller et al., 2014; Weeds et al., 2014), and dot-product $\vec{x} \cdot \vec{y}$. Using neural word embeddings (Mikolov et al., 2013; Pennington et al., 2014), these methods are easy to apply, and show good results (Baroni et al., 2012; Roller et al., 2014).

2.2 Path-based Methods

A different approach to detecting hypernymy between a pair of terms (x, y) considers the lexico-syntactic paths that connect the *joint* occurrences of x and y in a large corpus. Automatic acquisition of hypernyms from free text, based on such paths, was first proposed by Hearst (1992), who identified a small set of lexico-syntactic paths that indicate hypernymy relations (e.g. *Y such as X*, *X and other Y*).

In a later work, Snow et al. (2004) learned to detect hypernymy. Rather than searching for specific paths that indicate hypernymy, they represent each (x, y) term-pair as the multiset of all dependency paths that connect x and y in the corpus, and train a logistic regression classifier to predict whether y is a hypernym of x, based on these paths.

Paths that indicate hypernymy are those that were assigned high weights by the classifier. The paths identified by this method were shown to subsume those found by Hearst (1992), yielding improved performance. Variations of Snow et al.'s (2004) method were later used in tasks such as taxonomy construction (Snow et al., 2006; Kozareva and Hovy, 2010; Carlson et al., 2010; Riedel et al., 2013), analogy identification (Turney, 2006), and definition extraction (Borg et al., 2009; Navigli and Velardi, 2010).

A major limitation in relying on lexico-syntactic paths is the sparsity of the feature space. Since similar paths may somewhat vary at the lexical level, generalizing such variations into more abstract paths can increase recall. The PATTY algorithm (Nakashole et al., 2012) applied such generalizations for the purpose of acquiring a taxon-

[1] Our code and data are available in: https://github.com/vered1986/HypeNET

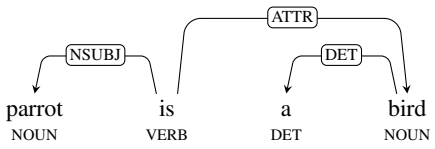


Figure 1: An example dependency tree of the sentence "parrot is a bird", with *x=parrot* and *y=bird*, represented in our notation as `X/NOUN/nsubj/< be/VERB/ROOT/-Y/NOUN/attr/>`.

omy of term relations from free text. For each path, they added generalized versions in which a subset of words along the path were replaced by either their POS tags, their ontological types or wild-cards. This generalization increased recall while maintaining the same level of precision.

2.3 RNNs for Relation Classification

Relation classification is a related task whose goal is to classify the relation that is expressed between two target terms in a given sentence to one of predefined relation classes. To illustrate, consider the following sentence, from the SemEval-2010 relation classification task dataset (Hendrickx et al., 2009): "The $[\text{apples}]_{e_1}$ are in the $[\text{basket}]_{e_2}$". Here, the relation expressed between the target entities is $Content - Container(e_1, e_2)$.

The shortest dependency paths between the target entities were shown to be informative for this task (Fundel et al., 2007). Recently, deep learning techniques showed good performance in capturing the indicative information in such paths.

In particular, several papers show improved performance using recurrent neural networks (RNN) that process a dependency path edge-by-edge. Xu et al. (2015; 2016) apply a separate long short-term memory (LSTM) network to each sequence of words, POS tags, dependency labels and WordNet hypernyms along the path. A max-pooling layer on the LSTM outputs is used as the input of a network that predicts the classification. Other papers suggest incorporating additional network architectures to further improve performance (Nguyen and Grishman, 2015; Liu et al., 2015).

While relation classification and hypernymy detection are both concerned with identifying semantic relations that hold for pairs of terms, they differ in a major respect. In relation classification the relation should be expressed in the given text, while in hypernymy detection, the goal is to recognize a generic lexical-semantic relation between terms that holds in many contexts. Accordingly, in relation classification a term-pair is represented by a single dependency path, while in hypernymy detection it is represented by the multiset of all dependency paths in which they co-occur in the corpus.

3 LSTM-based Hypernymy Detection

We present HypeNET, an LSTM-based method for hypernymy detection. We first focus on improving path representation (Section 3.1), and then integrate distributional signals into our network, resulting in a combined method (Section 3.2).

3.1 Path-based Network

Similarly to prior work, we represent each dependency path as a sequence of edges that leads from x to y in the dependency tree.[2] Each edge contains the lemma and part-of-speech tag of the source node, the dependency label, and the edge direction between two subsequent nodes. We denote each edge as $lemma/POS/dep/dir$. See figure 1 for an illustration.

Rather than treating an entire dependency path as a single feature, we encode the sequence of edges using a long short-term memory (LSTM) network. The vectors obtained for the different paths of a given (x, y) pair are pooled, and the resulting vector is used for classification. Figure 2 depicts the overall network structure, which is described below.

Edge Representation We represent each edge by the concatenation of its components' vectors:

$$\vec{v_e} = [\vec{v_l}, \vec{v_{pos}}, \vec{v_{dep}}, \vec{v_{dir}}]$$

where $\vec{v_l}, \vec{v_{pos}}, \vec{v_{dep}}, \vec{v_{dir}}$ represent the embedding vectors of the lemma, part-of-speech, dependency label and dependency direction (along the path from x to y), respectively.

Path Representation For a path p composed of edges $e_1, ..., e_k$, the edge vectors $\vec{v_{e_1}}, ..., \vec{v_{e_k}}$ are fed in order to an LSTM encoder, resulting in a vector $\vec{o_p}$ representing the entire path p. The LSTM architecture is effective at capturing temporal patterns in sequences. We expect the training procedure to drive the LSTM encoder to focus on parts of the path that are more informative for the classification task while ignoring others.

[2] Like Snow et al. (2004), we added for each path, additional paths containing single daughters of x or y not already contained in the path, referred by Snow et al. (2004) as "satellite edges". This enables including paths like *Such Y as X*, in which the word "such" is not in the path between x and y.

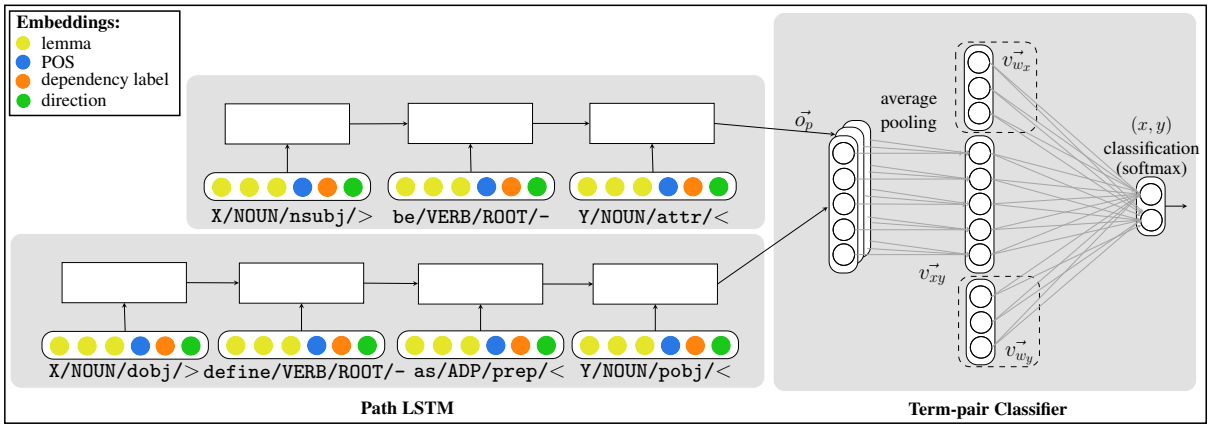


Figure 2: An illustration of term-pair classification. Each term-pair is represented by several paths. Each path is a sequence of edges, and each edge consists of four components: lemma, POS, dependency label and dependency direction. Each edge vector is fed in sequence into the LSTM, resulting in a path embedding vector $\vec{o_p}$. The averaged path vector becomes the term-pair's feature vector, used for classification. The dashed $\vec{v_{w_x}}, \vec{v_{w_y}}$ vectors refer to the integrated network described in Section 3.2.

Term-Pair Classification Each (x, y) term-pair is represented by the multiset of lexico-syntactic paths that connected x and y in the corpus, denoted as $paths(x, y)$, while the supervision is given for the term pairs. We represent each (x, y) term-pair as the weighted-average of its path vectors, by applying average pooling on its path vectors, as follows:

$$\vec{v_{xy}} = \vec{v}_{paths(x,y)} = \frac{\sum_{p \in paths(x,y)} f_{p,(x,y)} \cdot \vec{o_p}}{\sum_{p \in paths(x,y)} f_{p,(x,y)}} \quad (1)$$

where $f_{p,(x,y)}$ is the frequency of p in $paths(x, y)$. We then feed this path vector to a single-layer network that performs binary classification to decide whether y is a hypernym of x.

$$c = softmax(W \cdot \vec{v_{xy}}) \quad (2)$$

c is a 2-dimensional vector whose components sum to 1, and we classify a pair as positive if $c[1] > 0.5$.

Implementation Details To train the network, we used PyCNN.[3] We minimize the cross entropy loss using gradient-based optimization, with mini-batches of size 10 and the Adam update rule (Kingma and Ba, 2014). Regularization is applied by a dropout on each of the components' embeddings. We tuned the hyper-parameters (learning rate and dropout rate) on the validation set (see the appendix for the hyper-parameters values).

We initialized the lemma embeddings with the pre-trained GloVe word embeddings (Pennington et al., 2014), trained on Wikipedia. We tried both the 50-dimensional and 100-dimensional embedding vectors and selected the ones that yield better performance on the validation set.[4] The other embeddings, as well as out-of-vocabulary lemmas, are initialized randomly. We update all embedding vectors during training.

3.2 Integrated Network

The network presented in Section 3.1 classifies each (x, y) term-pair based on the paths that connect x and y in the corpus. Our goal was to improve upon previous path-based methods for hypernymy detection, and we show in Section 6 that our network indeed outperforms them. Yet, as path-based and distributional methods are considered complementary, we present a simple way to integrate distributional features in the network, yielding improved performance.

We extended the network to take into account distributional information on each term. Inspired by the supervised distributional concatenation method (Baroni et al., 2012), we simply concatenate x and y word embeddings to the (x, y) feature vector, redefining $\vec{v_{xy}}$:

$$\vec{v_{xy}} = [\vec{v_{w_x}}, \vec{v}_{paths(x,y)}, \vec{v_{w_y}}] \quad (3)$$

where $\vec{v_{w_x}}$ and $\vec{v_{w_y}}$ are x and y's word embeddings, respectively, and $\vec{v}_{paths(x,y)}$ is the averaged path vector defined in equation 1. This way, each (x, y) pair is represented using both the distributional features of x and y, and their path-based features.

[3] https://github.com/clab/cnn

[4] Higher-dimensional embeddings seem not to improve performance, while hurting the training runtime.

resource	relations
WordNet	instance hypernym, hypernym
DBPedia	type
Wikidata	subclass of, instance of
Yago	subclass of

Table 1: Hypernymy relations in each resource.

	train	test	val	all
random split	49,475	17,670	3,534	70,679
lexical split	20,335	6,610	1,350	28,295

Table 2: The number of instances in each dataset.

4 Dataset

4.1 Creating Instances

Neural networks typically require a large amount of training data, whereas the existing hypernymy datasets, like BLESS (Baroni and Lenci, 2011), are relatively small. Therefore, we followed the common methodology of creating a dataset using distant supervision from knowledge resources (Snow et al., 2004; Riedel et al., 2013). Following Snow et al. (2004), who constructed their dataset based on WordNet hypernymy, and aiming to create a larger dataset, we extract hypernymy relations from several resources: WordNet (Fellbaum, 1998), DBPedia (Auer et al., 2007), Wikidata (Vrandečić, 2012) and Yago (Suchanek et al., 2007).

All instances in our dataset, both positive and negative, are pairs of terms that are directly related in at least one of the resources. These resources contain thousands of relations, some of which indicate hypernymy with varying degrees of certainty. To avoid including questionable relation types, we consider as denoting positive examples only indisputable hypernymy relations (Table 1), which we manually selected from the set of hypernymy indicating relations in Shwartz et al. (2015).

Term-pairs related by other relations (including hyponymy), are considered as negative instances. Using related rather than random term-pairs as negative instances tests our method's ability to distinguish between hypernymy and other kinds of semantic relatedness. We maintain a ratio of 1:4 positive to negative pairs in the dataset.

Like Snow et al. (2004), we include only term-pairs that have joint occurrences in the corpus, requiring at least two different dependency paths for each pair.

4.2 Random and Lexical Dataset Splits

As our primary dataset, we perform standard random splitting, with 70% train, 25% test and 5% validation sets.

As pointed out by Levy et al. (2015), supervised distributional lexical inference methods tend to perform "lexical memorization", i.e., instead of learning a relation between the two terms, they mostly learn an independent property of a single term in the pair: whether it is a "prototypical hypernym" or not. For instance, if the training set contains term-pairs such as *(dog, animal)*, *(cat, animal)*, and *(cow, animal)*, all annotated as positive examples, the algorithm may learn that *animal* is a prototypical hypernym, classifying any new *(x, animal)* pair as positive, regardless of the relation between x and *animal*. Levy et al. (2015) suggested to split the train and test sets such that each will contain a distinct vocabulary ("lexical split"), in order to prevent the model from overfitting by lexical memorization.

To investigate such behaviors, we present results also for a lexical split of our dataset. In this case, we split the train, test and validation sets such that each contains a distinct vocabulary. We note that this differs from Levy et al. (2015), who split only the train and the test sets, and dedicated a subset of the train for validation. We chose to deviate from Levy et al. (2015) because we noticed that when the validation set contains terms from the train set, the model is rewarded for lexical memorization when tuning the hyper-parameters, consequently yielding suboptimal performance on the lexically-distinct test set. When each set has a distinct vocabulary, the hyper-parameters are tuned to avoid lexical memorization and are likely to perform better on the test set. We tried to keep roughly the same 70/25/5 ratio in our lexical split.[5] The sizes of the two datasets are shown in Table 2.

Indeed, training a model on a lexically split dataset may result in a more general model, that can better handle pairs consisting of two unseen terms during inference. However, we argue that in the common applied scenario, the inference involves an unseen pair (x, y), in which x and/or y have already been observed separately. Models trained on a random split may introduce the model with a term's "prior probability" of being a hypernym or a hyponym, and this information can be exploited beneficially at inference time.

[5]The lexical split discards many pairs consisting of cross-set terms.

path
X/NOUN/dobj/> establish/VERB/ROOT/- as/ADP/prep/< Y/NOUN/pobj/<
X/NOUN/dobj/> VERB as/ADP/prep/< Y/NOUN/pobj/<
X/NOUN/dobj/> * as/ADP/prep/< Y/NOUN/pobj/<
X/NOUN/dobj/> establish/VERB/ROOT/- ADP Y/NOUN/pobj/<
X/NOUN/dobj/> establish/VERB/ROOT/- * Y/NOUN/pobj/<

Table 3: Example generalizations of *X was established as Y*.

5 Baselines

We compare HypeNET with several state-of-the-art methods for hypernymy detection, as described in Section 2: path-based methods (Section 5.1), and distributional methods (Section 5.2). Due to different works using different datasets and corpora, we replicated the baselines rather than comparing to the reported results.

We use the Wikipedia dump from May 2015 as the underlying corpus of all the methods, and parse it using spaCy.[6] We perform model selection on the validation set to tune the hyper-parameters of each method.[7] The best hyper-parameters are reported in the appendix.

5.1 Path-based Methods

Snow We follow the original paper, and extract all shortest paths of four edges or less between terms in a dependency tree. Like Snow et al. (2004), we add paths with "satellite edges", i.e., single words not already contained in the dependency path, which are connected to either X or Y, allowing paths like *such Y as X*. The number of distinct paths was 324,578. We apply χ^2 feature selection to keep only the 100,000 most informative paths and train a logistic regression classifier.

Generalization We also compare our method to a baseline that uses generalized dependency paths. Following PATTY's approach to generalizing paths (Nakashole et al., 2012), we replace edges with their part-of-speech tags as well as with wild cards. We generate the powerset of all possible generalizations, including the original paths. See Table 3 for examples. The number of features after generalization went up to 2,093,220. Similarly to the first baseline, we apply feature selection, this time keeping the 1,000,000 most informative paths, and train a logistic regression classifier over the generalized paths.[8]

[6] https://spacy.io/

[7] We applied grid search for a range of values, and picked the ones that yield the highest F_1 score on the validation set.

[8] We also tried keeping the 100,000 most informative paths, but the performance was worse.

5.2 Distributional Methods

Unsupervised SLQS (Santus et al., 2014) is an entropy-based measure for hypernymy detection, reported to outperform previous state-of-the-art unsupervised methods (Weeds and Weir, 2003; Kotlerman et al., 2010). The original paper was evaluated on the BLESS dataset (Baroni and Lenci, 2011), which consists of mostly frequent words. Applying the vanilla settings of SLQS on our dataset, that contains also rare terms, resulted in low performance. Therefore, we received assistance from Enrico Santus, who kindly provided the results of SLQS on our dataset after tuning the system as follows.

The validation set was used to tune the threshold for classifying a pair as positive, as well as the maximum number of each term's most associated contexts (N). In contrast to the original paper, in which the number of each term's contexts is fixed to N, in this adaptation it was set to the minimum between the number of contexts with LMI score above zero and N. In addition, the SLQS scores were not multiplied by the cosine similarity scores between terms, and terms were lemmatized prior to computing the SLQS scores, significantly improving recall.

As our results suggest, while this method is state-of-the-art for unsupervised hypernymy detection, it is basically designed for classifying specificity level of related terms, rather than hypernymy in particular.

Supervised To represent term-pairs with distributional features, we tried several state-of-the-art methods: concatenation $\vec{x} \oplus \vec{y}$ (Baroni et al., 2012), difference $\vec{y} - \vec{x}$ (Roller et al., 2014; Weeds et al., 2014), and dot-product $\vec{x} \cdot \vec{y}$. We downloaded several pre-trained embeddings (Mikolov et al., 2013; Pennington et al., 2014) of different sizes, and trained a number of classifiers: logistic regression, SVM, and SVM with RBF kernel, which was reported by Levy et al. (2015) to perform best in this setting. We perform model selection on the validation set to select the best vectors, method and regularization factor (see the appendix).

method		random split			lexical split		
		precision	**recall**	**F_1**	**precision**	**recall**	**F_1**
Path-based	Snow	0.843	0.452	0.589	0.760	0.438	0.556
	Snow + Gen	0.852	0.561	0.676	0.759	0.530	0.624
	HypeNET Path-based	0.811	0.716	0.761	0.691	**0.632**	0.660
Distributional	SLQS (Santus et al., 2014)	0.491	0.737	0.589	0.375	0.610	0.464
	Best supervised (concatenation)	0.901	0.637	0.746	0.754	0.551	0.637
Combined	HypeNET Integrated	**0.913**	**0.890**	**0.901**	**0.809**	0.617	**0.700**

Table 4: Performance scores of our method compared to the path-based baselines and the state-of-the-art distributional methods for hypernymy detection, on both variations of the dataset – with lexical and random split to train / test / validation.

6 Results

Table 4 displays performance scores of HypeNET and the baselines. *HypeNET Path-based* is our path-based recurrent neural network model (Section 3.1) and *HypeNET Integrated* is our combined method (Section 3.2). Comparing the path-based methods shows that generalizing paths improves recall while maintaining similar levels of precision, reassessing the behavior found in Nakashole et al. (2012). *HypeNET Path-based* outperforms both path-based baselines by a significant improvement in recall and with slightly lower precision. The recall boost is due to better path generalization, as demonstrated in Section 7.1.

Regarding distributional methods, the unsupervised SLQS baseline performed slightly worse on our dataset. The low precision stems from its inability to distinguish between hypernyms and meronyms, which are common in our dataset, causing many false positive pairs such as *(zabrze, poland)* and *(kibbutz, israel)*. We sampled 50 false positive pairs of each dataset split, and found that 38% of the false positive pairs in the random split and 48% of those in the lexical split were holonym-meronym pairs.

In accordance with previously reported results, the supervised embedding-based method is the best performing baseline on our dataset as well. *HypeNET Path-based* performs slightly better, achieving state-of-the-art results. Adding distributional features to our method shows that these two approaches are indeed complementary. On both dataset splits, the performance differences between *HypeNET Integrated* and *HypeNET Path-based*, as well as the supervised distributional method, are substantial, and statistically significant with p-value of 1% (paired t-test).

We also reassess that indeed supervised distributional methods perform worse on a lexical split (Levy et al., 2015). We further observe a similar reduction when using HypeNET, which is not a result of lexical memorization, but rather stems from over-generalization (Section 7.1).

7 Analysis

7.1 Qualitative Analysis of Learned Paths

We analyze HypeNET's ability to generalize over path structures, by comparing prominent indicative paths which were learned by each of the path-based methods. We do so by finding high-scoring paths that contributed to the classification of true-positive pairs in the dataset. In the path-based baselines, these are the highest-weighted features as learned by the logistic regression classifier. In the LSTM-based method, it is less straightforward to identify the most indicative paths. We assess the contribution of a certain path p to classification by regarding it as the only path that appeared for the term-pair, and compute its TRUE label score from the class distribution: $softmax(W \cdot \vec{v_{xy}})[1]$, setting $\vec{v_{xy}} = [\vec{0}, \vec{o_p}, \vec{0}]$.

A notable pattern is that Snow's method learns specific paths, like *X is Y from* (e.g. ***Megadeth** is an American thrash metal **band** from Los Angeles*). While Snow's method can only rely on verbatim paths, limiting its recall, the generalized version of Snow often makes coarse generalizations, such as *X VERB Y from*. Clearly, such a path is too general, and almost any verb assigned to it results in a non-indicative path (e.g. *X take Y from*). Efforts by the learning method to avoid such generalization, again, lower the recall. HypeNET provides a better midpoint, making fine-grained generalizations by learning additional semantically similar paths such as *X become Y from* and *X remain Y from*. See table 5 for additional example paths which illustrate these behaviors.

We also noticed that while on the random split our model learns a range of specific paths such as *X is Y published* (learned for e.g. *Y=magazine*) and *X is Y produced* (*Y=film*), in the lexical split it only learns the general *X is Y* path for these re-

method	path	example text
Snow	X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/< direct/VERB/acl/>	*Eyeball* is a 1975 Italian-Spanish *film* directed by Umberto Lenzi
	X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/< publish/VERB/acl/>	*Allure* is a U.S. women's beauty *magazine* published monthly
Snow + Gen	X/NOUN/compound/> NOUN* be/VERB/ROOT/- Y/NOUN/attr/< base/VERB/acl/>	*Calico* Light Weapons Inc. (CLWS) is an American privately held manufacturing *company* based in Cornelius, Oregon
	X/NOUN/compound/> NOUN Y/NOUN/compound/<	*Weston Town* Council
HypeNET Integrated	X/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/< (release|direct|produce|write)/VERB/acl/>	*Blinky* is a 1923 American comedy *film* directed by Edward Sedgwick
	X/NOUN/compound/> (association|co.|company|corporation|foundation|group|inc.|international|limited|ltd.)/NOUN/nsubj/> be/VERB/ROOT/- Y/NOUN/attr/< ((create|found|headquarter|own|specialize)/VERB/acl/>)?	*Retalix* Ltd. is a software *company*

Table 5: Examples of indicative paths learned by each method, with corresponding true positive term-pairs from the random split test set. Hypernyms are marked red and hyponyms are marked blue.

Relation	%
synonymy	21.37%
hyponymy	29.45%
holonymy / meronymy	9.36%
hypernymy-like relations	21.03%
other relations	18.77%

Table 6: Distribution of relations holding between each pair of terms in the resources among false positive pairs.

lations. We note that *X is Y* is a rather "noisy" path, which may occur in ad-hoc contexts without indicating generic hypernymy relations (e.g. ***chocolate*** *is a big* ***problem*** in the context of children's health). While such a model may identify hypernymy relations between unseen terms, based on general paths, it is prone to over-generalization, hurting its performance, as seen in Table 4. As discussed in § 4.2, we suspect that this scenario, in which both terms are unseen, is usually not common enough to justify this limiting training setup.

7.2 Error Analysis

False Positives We categorized the false positive pairs on the random split according to the relation holding between each pair of terms in the resources used to construct the dataset. We grouped several semantic relations from different resources to broad categories, e.g. *synonym* includes also *alias* and *Wikipedia redirection*. Table 6 displays the distribution of semantic relations among false positive pairs.

More than 20% of the errors stem from confusing synonymy with hypernymy, which are known to be difficult to distinguish.

An additional 30% of the term-pairs are reversed hypernym-hyponym pairs (y is a hyponym of x). Examining a sample of these pairs suggests that they are usually near-synonyms, i.e., it is not that clear whether one term is truely more general than the other or not. For instance, *fiction* is annotated in WordNet as a hypernym of *story*, while our method classified *fiction* as its hyponym.

A possible future research direction might be to quite simply extend our network to classify term-pairs simultaneously to multiple semantic relations, as in Pavlick et al. (2015). Such a multi-class model can hopefully better distinguish between these similar semantic relations.

Another notable category is hypernymy-like relations: these are other relations in the resources that could also be considered as hypernymy, but were annotated as negative due to our restrictive selection of only indisputable hypernymy relations from the resources (see Section 4.1). These include instances like *(Goethe, occupation, novelist)* and *(Homo, subdivisionRanks, species)*.

Lastly, other errors made by the model often correspond to term-pairs that co-occur very few times in the corpus, e.g. $xebec$, a studio producing Anime, was falsely classified as a hyponym of *anime*.

False Negatives We sampled 50 term-pairs that were falsely annotated as negative, and analyzed the major (overlapping) types of errors (Table 7).

Most of these pairs had only few co-occurrences in the corpus. This is often either due to infrequent terms (e.g. *cbc.ca*), or a rare sense of x in which the hypernymy relation holds (e.g. *(night,*

	Error Type	%
1	low statistics	80%
2	infrequent term	36%
3	rare hyponym sense	16%
4	annotation error	8%

Table 7: (Overlapping) categories of false negative pairs: (1) x and y co-occurred less than 25 times (average co-occurrences for true positive pairs is 99.7). (2) Either x or y is infrequent. (3) The hypernymy relation holds for a rare sense of x. (4) (x, y) was incorrectly annotated as positive.

play) holding for "Night", a dramatic sketch by Harold Pinter). Such a term-pair may have too few hypernymy-indicating paths, leading to classifying it as negative.

8 Conclusion

We presented HypeNET: a neural-networks-based method for hypernymy detection. First, we focused on improving path representation using LSTM, resulting in a path-based model that performs significantly better than prior path-based methods, and matches the performance of the previously superior distributional methods. In particular, we demonstrated that the increase in recall is a result of generalizing semantically-similar paths, in contrast to prior methods, which either make no generalizations or over-generalize paths.

We then extended our network by integrating distributional signals, yielding an improvement of additional 14 F_1 points, and demonstrating that the path-based and the distributional approaches are indeed complementary.

Finally, our architecture seems straightforwardly applicable for multi-class classification, which, in future work, could be used to classify term-pairs to multiple semantic relations.

Acknowledgments

We would like to thank Omer Levy for his involvement and assistance in the early stage of this project and Enrico Santus for helping us by computing the results of SLQS (Santus et al., 2014) on our dataset.

This work was partially supported by an Intel ICRI-CI grant, the Israel Science Foundation grant 880/12, and the German Research Foundation through the German-Israeli Project Cooperation (DIP, grant DA 1600/1-1).

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. *Dbpedia: A nucleus for a web of open data*. Springer.

Marco Baroni and Alessandro Lenci. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, pages 1–10.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. 2012. Entailment above the word level in distributional semantics. In *EACL*, pages 23–32.

Claudia Borg, Mike Rosner, and Gordon Pace. 2009. Evolutionary algorithms for definition extraction. In *Proceedings of the 1st Workshop on Definition Extraction*, pages 26–32.

Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R Hruschka Jr, and Tom M Mitchell. 2010. Toward an architecture for never-ending language learning. In *AAAI*.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Katrin Fundel, Robert Küffner, and Ralf Zimmer. 2007. Relexrelation extraction using dependency parse trees. *Bioinformatics*, pages 365–371.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *ACL*, pages 539–545.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2009. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *SemEval*, pages 94–99.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, pages 1735–1780.

Nobuhiro Kaji and Masaru Kitsuregawa. 2008. Using hidden markov random fields to combine distributional and pattern-based word clustering. In *COLING*, pages 401–408.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lili Kotlerman, Ido Dagan, Idan Szpektor, and Maayan Zhitomirsky-Geffet. 2010. Directional distributional similarity for lexical inference. *NLE*, pages 359–389.

Zornitsa Kozareva and Eduard Hovy. 2010. A semi-supervised method to learn and construct taxonomies using the web. In *EMNLP*, pages 1110–1118.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. 2015. Do supervised distributional methods really learn lexical inference relations. *NAACL*.

Dekang Lin. 1998. An information-theoretic definition of similarity. In *ICML*, pages 296–304.

Yang Liu, Furu Wei, Sujian Li, Heng Ji, Ming Zhou, and Houfeng Wang. 2015. A dependency-based neural network for relation classification. *arXiv preprint arXiv:1507.04646*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *COLING and ACL*, pages 579–586.

Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. Patty: a taxonomy of relational patterns with semantic types. In *EMNLP and CoNLL*, pages 1135–1145.

Roberto Navigli and Paola Velardi. 2010. Learning word-class lattices for definition and hypernym extraction. In *ACL*, pages 1318–1327.

Thien Huu Nguyen and Ralph Grishman. 2015. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*.

Ellie Pavlick, Johan Bos, Malvina Nissim, Charley Beller, Benjamin Van Durme, and Chris Callison-Burch. 2015. Adding semantics to data-driven paraphrasing. In *ACL*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

Laura Rimell. 2014. Distributional lexical entailment by topic coherence. In *EACL*, pages 511–519.

Stephen Roller, Katrin Erk, and Gemma Boleda. 2014. Inclusive yet selective: Supervised distributional hypernymy detection. In *COLING*, pages 1025–1036.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte Im Walde. 2014. Chasing hypernyms in vector spaces with entropy. In *EACL*, pages 38–42.

Vered Shwartz, Omer Levy, Ido Dagan, and Jacob Goldberger. 2015. Learning to exploit structured resources for lexical inference. In *CoNLL*, page 175.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2004. Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. 2006. Semantic taxonomy induction from heterogenous evidence. In *ACL*, pages 801–808.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *WWW*, pages 697–706.

Peter D Turney. 2006. Similarity of semantic relations. *CL*, pages 379–416.

Denny Vrandečić. 2012. Wikidata: A new platform for collaborative data collection. In *WWW*, pages 1063–1064.

Julie Weeds and David Weir. 2003. A general framework for distributional similarity. In *EMLP*, pages 81–88.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David Weir, and Bill Keller. 2014. Learning to distinguish hypernyms and co-hyponyms. In *COLING*, pages 2249–2259.

Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. 2015. Classifying relations via long short term memory networks along shortest dependency paths. In *EMNLP*.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*.

Appendix A Best Hyper-parameters

Table 8 displays the chosen hyper-parameters of each method, yielding the highest F_1 score on the validation set.

	method	values
random split	Snow	regularization: L_2
	Snow + Gen	regularization: L_1
	LSTM	embeddings: GloVe-100-Wiki learning rate: $\alpha = 0.001$ dropout: $d = 0.5$
	SLQS	N=100, threshold = 0.000464
	Best Supervised	method: concatenation, classifier: SVM embeddings: GloVe-300-Wiki
	LSTM-Integrated	embeddings: GloVe-50-Wiki learning rate: $\alpha = 0.001$ word dropout: $d = 0.3$
lexical split	Snow	regularization: L_2
	Snow + Gen	regularization: L_2
	LSTM	embeddings: GloVe-50-Wiki learning rate: $\alpha = 0.001$ dropout: $d = 0.5$
	SLQS	N=100, threshold = 0.007629
	Best Supervised	method: concatenation, classifier: SVM embeddings: GloVe-100-Wikipedia
	LSTM-Integrated	embeddings: GloVe-50-Wiki learning rate: $\alpha = 0.001$ word dropout: $d = 0.3$

Table 8: The best hyper-parameters in every model.

Multimodal Pivots for Image Caption Translation

Julian Hitschler and **Shigehiko Schamoni**
Computational Linguistics
Heidelberg University
69120 Heidelberg, Germany
{hitschler,schamoni}@cl.uni-heidelberg.de

Stefan Riezler
Computational Linguistics & IWR
Heidelberg University
69120 Heidelberg, Germany
riezler@cl.uni-heidelberg.de

Abstract

We present an approach to improve statistical machine translation of image descriptions by multimodal pivots defined in visual space. The key idea is to perform image retrieval over a database of images that are captioned in the target language, and use the captions of the most similar images for crosslingual reranking of translation outputs. Our approach does not depend on the availability of large amounts of in-domain parallel data, but only relies on available large datasets of monolingually captioned images, and on state-of-the-art convolutional neural networks to compute image similarities. Our experimental evaluation shows improvements of 1 BLEU point over strong baselines.

1 Introduction

Multimodal data consisting of images and natural language descriptions (henceforth called *captions*) are an abundant source of information that has led to a recent surge in research integrating language and vision. Recently, the aspect of multilinguality has been added to multimodal language processing in a shared task at the WMT16 conference.[1] There is clearly also a practical demand for multilingual image captions, e.g., automatic translation of descriptions of art works would allow access to digitized art catalogues across language barriers and is thus of social and cultural interest; multilingual product descriptions are of high commercial interest since they would allow to widen e-commerce transactions automatically to international markets. However, while datasets of images and monolingual captions already include millions of tuples (Ferraro et al., 2015), the largest multilingual datasets of images and captions known to the authors contain 20,000 (Grubinger et al., 2006) or 30,000[2] triples of images with German and English descriptions.

In this paper, we want to address the problem of multilingual captioning from the perspective of statistical machine translation (SMT). In contrast to prior work on generating captions directly from images (Kulkarni et al. (2011), Karpathy and Fei-Fei (2015), Vinyals et al. (2015), *inter alia*), our goal is to integrate visual information into an SMT pipeline. Visual context provides orthogonal information that is free of the ambiguities of natural language, therefore it serves to disambiguate and to guide the translation process by grounding the translation of a source caption in the accompanying image. Since datasets consisting of source language captions, images, and target language captions are not available in large quantities, we would instead like to utilize large datasets of images and target-side monolingual captions to improve SMT models trained on modest amounts of parallel captions.

Let the task of *caption translation* be defined as follows: For production of a target caption e_i of an image i, a system may use as input an image caption for image i in the source language f_i, as well as the image i itself. The system may safely assume that f_i is relevant to i, i.e., the identification of relevant captions for i (Hodosh et al., 2013) is not itself part of the task of caption translation. In contrast to the inference problem of finding $\hat{e} = \text{argmax}_e\, p(e|f)$ in text-based SMT, multimodal caption translation allows to take into consideration i as well as f_i in finding $\hat{e}_i$:

$$\hat{e}_i = \underset{e_i}{\text{argmax}}\, p(e_i|f_i, i)$$

[1] http://www.statmt.org/wmt16/multimodal-task.html

[2] The dataset used at the WMT16 shared task is based on translations of Flickr30K captions (Rashtchian et al., 2010).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2399–2409,
Berlin, Germany, August 7-12, 2016.

In this paper, we approach caption translation by a general crosslingual reranking framework where for a given pair of source caption and image, monolingual captions in the target language are used to rerank the output of the SMT system. We present two approaches to retrieve target language captions for reranking by pivoting on images that are similar to the input image. One approach calculates image similarity based deep convolutional neural network (CNN) representations. Another approach calculates similarity in visual space by comparing manually annotated object categories. We compare the multimodal pivot approaches to reranking approaches that are based on text only, and to standard SMT baselines trained on parallel data. Compared to a strong baseline trained on 29,000 parallel caption data, we find improvements of over 1 BLEU point for reranking based on visual pivots. Notably, our reranking approach does not rely on large amounts of in-domain parallel data which are not available in practical scenarios such as e-commerce localization. However, in such scenarios, monolingual product descriptions are naturally given in large amounts, thus our work is a promising pilot study towards real-world caption translation.

2 Related Work

Caption generation from images alone has only recently come into the scope of realistically solvable problems in image processing (Kulkarni et al. (2011), Karpathy and Fei-Fei (2015), Vinyals et al. (2015), *inter alia*). Recent approaches also employ reranking of image captions by measuring similarity between image and text using deep representations (Fang et al., 2015). The tool of choice in these works are neural networks whose deep representations have greatly increased the quality of feature representations of images, enabling robust and semantically salient analysis of image content. We rely on the CNN framework (Socher et al., 2014; Simonyan and Zisserman, 2015) to solve semantic classification and disambiguation tasks in NLP with the help of supervision signals from visual feedback. However, we consider image captioning as a different task than caption translation since it is not given the information of the source language string. Therefore we do not compare our work to caption generation models.

In the area of SMT, Wäschle and Riezler (2015) presented a framework for integrating a large, in-domain, target-side monolingual corpus into machine translation by making use of techniques from crosslingual information retrieval. The intuition behind their approach is to generate one or several translation hypotheses using an SMT system, which act as queries to find matching, semantically similar sentences in the target side corpus. These can in turn be used as templates for refinement of the translation hypotheses, with the overall effect of improving translation quality. Our work can be seen as an extension of this method, with visual similarity feedback as additional constraint on the crosslingual retrieval model. Calixto et al. (2012) suggest using images as supplementary context information for statistical machine translation. They cite examples from the news domain where visual context could potentially be helpful in the disambiguation aspect of SMT and discuss possible features and distance metrics for context images, but do not report experiments involving a full SMT pipeline using visual context. In parallel to our work, Elliott et al. (2015) addressed the problem of caption translation from the perspective of neural machine translation.[3] Their approach uses a model which is considerably more involved than ours and relies exclusively on the availability of parallel captions as training data. Both approaches crucially rely on neural networks, where they use a visually enriched neural encoder-decoder SMT approach, while we follow a retrieval paradigm for caption translation, using CNNs to compute similarity in visual space.

Integration of multimodal information into NLP problems has been another active area of recent research. For example, Silberer and Lapata (2014) show that distributional word embeddings grounded in visual representations outperform competitive baselines on term similarity scoring and word categorization tasks. The orthogonality of visual feedback has previously been exploited in a multilingual setting by Kiela et al. (2015) (relying on previous work by Bergsma and Van Durme (2011)), who induce a bilingual lexicon using term-specific multimodal representations obtained by querying the Google image

[3] We replicated the results of Elliott et al. (2015) on the IAPR TC-12 data. However, we decided to not include their model as baseline in this paper since we found our hierarchical phrase-based baselines to yield considerably better results on IAPR TC-12 as well as on MS COCO.

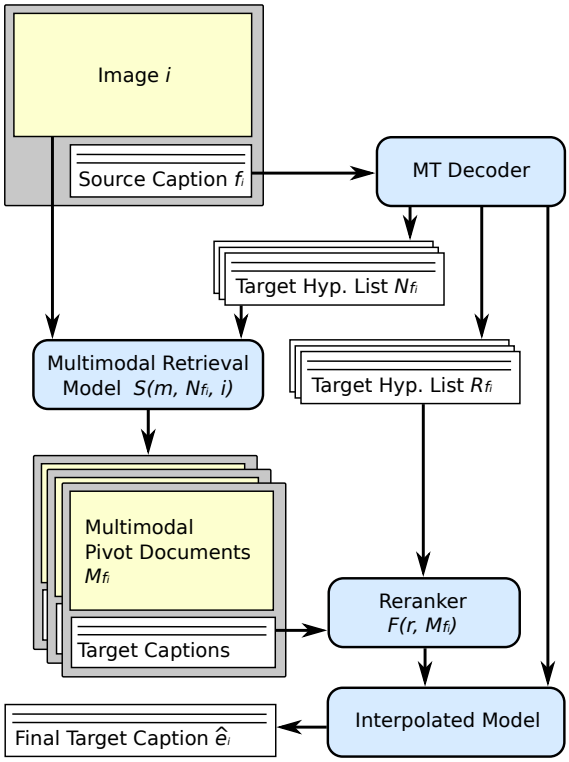


Figure 1: Overview of model architecture.

search engine.[4] Funaki and Nakayama (2015) use visual similarity for crosslingual document retrieval in a multimodal and bilingual vector space obtained by generalized canonical correlation analysis, greatly reducing the need for parallel training data. The common element is that CNN-based visual similarity information is used as a "hub" (Funaki and Nakayama, 2015) or pivot connecting corpora in two natural languages which lack direct parallelism, a strategy which we apply to the problem of caption translation.

3 Models

3.1 Overview

Following the basic approach set out by Wäschle and Riezler (2015), we use a crosslingual retrieval model to find sentences in a target language document collection C, and use these to rerank target language translations e of a source caption f.

The systems described in our work differ from that of Wäschle and Riezler (2015) in a number of aspects. Instead of a two-step architecture of coarse-grained and fine-grained retrieval, our system uses relevance scoring functions for retrieval of matches in the document collection C, and for reranking of translation candidates that are based on inverse document frequency of terms (Spärck Jones, 1972) and represent variants of the popular TF-IDF relevance measure.

A schematic overview of our approach is given in Figure 1. It consists of the following components:

Input: Source caption f_i, image i, target-side collection C of image-captions pairs

Translation: Generate unique list N_{f_i} of k_n-best translations, generate unique list R_{f_i} of k_r-best list of translations[5] using MT decoder

Multimodal retrieval: For list of translations N_{f_i}, find set M_{f_i} of k_m-most relevant pairs of images and captions in a target-side collection C, using a heuristic relevance scoring function $S(m, N_{f_i}, i), m \in C$

Crosslingual reranking: Use list M_{f_i} of image-caption pairs to rerank list of translations R_{f_i}, applying relevance scoring function $F(r, M_{f_i})$ to all $r \in R_{f_i}$

Output: Determine best translation hypothesis $\hat{e}_i$ by interpolating decoder score d_r for a hypothesis $r \in R_{f_i}$ with its relevance score $F(r, M_{f_i})$ with weight λ s.t.

$$\hat{e}_i = \underset{r \in R_{f_i}}{\operatorname{argmax}}\, d_r + \lambda \cdot F(r, M_{f_i})$$

The central concept is the scoring function $S(m, N_{f_i}, i)$ which defines three variants of target-side retrieval (TSR), all of which make use of the procedure outlined above. In the baseline text-based reranking model (TSR-TXT), we use relevance scoring function S_{TXT}. This function is purely text-based and does not make use of multimodal context information (as such, it comes closest to the models used for target-side retrieval in Wäschle and Riezler (2015)). In the retrieval model enhanced by visual information from a deep convolutional neural network (TSR-CNN), the scoring function S_{CNN} incorporates a textual relevance score with visual similarity information extracted from the neural network. Finally, we evaluate these models against a relevance score based on human object-category annotations (TSR-HCA), using the scoring function

[4] https://images.google.com/

[5] In practice, the first hypothesis list may be reused. We distinguish between the two hypothesis lists N_{f_i} and R_{f_i} for notational clarity since in general, the two hypothesis lists need not be of equal length.

S_{HCA}. This function makes use of the object annotations available for the MS COCO corpus (Lin et al., 2014) to give an indication of the effectiveness of our automatically extracted visual similarity metric. The three models are discussed in detail below.

3.2 Target Side Retrieval Models

Text-Based Target Side Retrieval. In the TSR-TXT retrieval scenario, a match candidate $m \in C$ is scored in the following way:

$$S_{TXT}(m, N_{f_i}) = Z_m \sum_{n \in N_{f_i}} \sum_{w_n \in tok(n)} \sum_{w_m \in typ(m)} \delta(w_m, w_n) idf(w_m),$$

where δ is the Kronecker δ-function, N_{f_i} is the set of the k_n-best translation hypotheses for a source caption f_i of image i by decoder score, $typ(a)$ is a function yielding the set of types (unique tokens) contained in a caption a,[6] $tok(a)$ is a function yielding the tokens of caption a, $idf(w)$ is the inverse document frequency (Spärck Jones, 1972) of term w, and $Z_m = \frac{1}{|typ(m)|}$ is a normalization term introduced in order to avoid biasing the system towards long match candidates containing many low-frequency terms. Term frequencies were computed on monolingual data from Europarl (Koehn, 2005) and the News Commentary and News Discussions English datasets provided for the WMT15 workshop.[7] Note that in this model, information from the image i is not used.

Multimodal Target Side Retrieval using CNNs. In the TSR-CNN scenario, we supplement the textual target-side TSR model with visual similarity information from a deep convolutional neural network. We formalize this by introduction of the positive-semidefinite distance function $v(i_x, i_y) \rightarrow [0, \infty)$ for images i_x, i_y (smaller values indicating more similar images). The relevance scoring function S_{CNN} used in this model takes the following form:

$$S_{CNN}(m, N_{f_i}, i) = \begin{cases} S_{TXT}(m, N_{f_i}) e^{-bv(i_m, i)}, & v(i_m, i) < d \\ 0 & otherwise, \end{cases}$$

where i_m is the image to which the caption m refers and d is a cutoff maximum distance, above which match candidates are considered irrelevant, and b is a weight term which controls the impact of the visual distance score $v(i_m, i)$ on the overall score.[8]

Our visual distance measure v was computed using the VGG16 deep convolutional model of Simonyan and Zisserman (2015), which was pretrained on ImageNet (Russakovsky et al., 2014). We extracted feature values for all input and reference images from the penultimate fully-connected layer (`fc7`) of the model and computed the Euclidean distance between feature vectors of images. If no neighboring images fell within distance d, the text-based retrieval procedure S_{TXT} was used as a fallback strategy, which occurred 47 out of 500 times on our test data.

Target Side Retrieval by Human Category Annotations. For contrastive purposes, we evaluated a TSR-HCA retrieval model which makes use of the human object category annotations for MS COCO. Each image in the MS COCO corpus is annotated with object polygons classified into 91 categories of common objects. In this scenario, a match candidate m is scored in the following way:

$$S_{HCA}(m, N_{f_i}, i) = \delta(cat(i_m), cat(i)) S_{TXT}(m, N_{f_i}),$$

where $cat(i)$ returns the set of object categories with which image i is annotated. The amounts to enforcing a strict match between the category annotations of i and the reference image i_m, thus pre-filtering the S_{TXT} scoring to captions for images with strict category match.[9] In cases where i was annotated with a unique set of object categories and thus no match candidates with nonzero scores were returned by S_{HCA}, S_{TXT} was used as a fallback strategy, which occurred 77 out of 500 times on our test data.

[6] The choice for per-type scoring of reference captions was primarily driven by performance considerations. Since captions rarely contain repetitions of low-frequency terms, this has very little effect in practice, other than to mitigate the influence of stopwords.

[7] `http://www.statmt.org/wmt15/translation-task.html`

[8] The value of $b = 0.01$ was found on development data and kept constant throughout the experiments.

[9] Attempts to relax this strict matching criterion led to strong performance degradation on the development test set.

3.3 Translation Candidate Re-scoring

The relevance score $F(r, M_{f_i})$ used in the reranking model was computed in the following way for all three models:

$$F(r, M_{f_i}) = Z_{M_{f_i}} \sum_{m \in M_{f_i}} \sum_{w_m \in typ(m)} \sum_{w_r \in tok(r)} \delta(w_m, w_r) idf(w_m)$$

with normalization term

$$Z_{M_{f_i}} = (\sum_{m \in M_{f_i}} |tok(m)|)^{-1},$$

where r is a translation candidate and M_{f_i} is a list of k_m-top target side retrieval matches. Because the model should return a score that is reflective of the relevance of r with respect to M_{f_i}, irrespective of the length of M_{f_i}, normalization with respect to the token count of M_{f_i} is necessary. The term $Z_{M_{f_i}}$ serves this purpose.

4 Experiments

4.1 Bilingual Image-Caption Data

We constructed a German-English parallel dataset based on the MS COCO image corpus (Lin et al., 2014). 1,000 images were selected at random from the 2014 training section[10] and, in a second step, one of their five English captions was chosen randomly. This caption was then translated into German by a native German speaker. Note that our experiments were performed with German as the source and English as the target language, therefore, our reference data was not produced by a single speaker but reflects the heterogeneity of the MS COCO dataset at large. The data was split into a development set of 250 captions, a development test set of 250 captions for testing work in progress, and a test set of 500 captions. For our retrieval experiments, we used only the images and captions that were not included in the development, development test or test data, a total of 81,822 images with 5 English captions per image. All data was tokenized and converted to lower case using the `cdec`[11] utilities `tokenized-anything.pl` and `lowercase.pl`. For the German data, we performed compound-splitting using the method described by Dyer (2009), as implemented by the `cdec` utility `compound-split.pl`. Table 1 gives an overview of the dataset. Our parallel development, development test and test data is publicly available.[12]

Section	Images	Captions	Languages
DEV	250	250	DE-EN
DEVTEST	250	250	DE-EN
TEST	500	500	DE-EN
RETRIEVAL (C)	81,822	409,110	EN

Table 1: Number of images and sentences in MS COCO image and caption data used in experiments.

4.2 Translation Baselines

We compare our approach to two baseline machine translation systems, one trained on out-of-domain data exclusively and one domain-adapted system. Table 2 gives an overview of the training data for the machine translation systems.

Out-of-Domain Baseline. Our baseline SMT framework is hierarchical phrase-based translation using synchronous context free grammars (Chiang, 2007), as implemented by the `cdec` decoder (Dyer et al., 2010). Data from the Europarl (Koehn, 2005), News Commentary and Common Crawl corpora (Smith et al., 2013) as provided for the WMT15 workshop was used to train the translation model, with German as source and English as target language.

Like the retrieval dataset, training, development and test data was tokenized and converted to lower case, using the same `cdec` tools. Sentences with lengths over 80 words in either the source or the target language were discarded before training. Source text compound splitting was performed using `compound-split.pl`. Alignments were extracted bidirectionally using the `fast-align` utility of `cdec` and symmetrized with the `atools` utility (also part of `cdec`) using the `grow-diag-final-and` symmetrization heuristic. The alignments were then used by the `cdec` grammar extractor to extract a synchronous context free grammar from the parallel data.

[10] We constructed our parallel dataset using only the training rather than the validation section of MS COCO so as to keep the latter pristine for future work based on this research.

[11] `https://github.com/redpony/cdec`

[12] `www.cl.uni-heidelberg.de/decoco/`

Corpus	Sentences	Languages	System
Europarl	1,920,209	DE-EN	O/I
News Commentary	216,190	DE-EN	O/I
Common Crawl	2,399,123	DE-EN	O/I
Flickr30k WMT16	29,000	DE-EN	I
Europarl	2,218,201	EN	O/I
News Crawl	28,127,448	EN	O/I
News Discussions	57,803,684	EN	O/I
Flickr30k WMT16	29,000	EN	I

Table 2: Parallel and monolingual data used for training machine translation systems. Sentence counts are given for raw data without preprocessing. O/I: both out-of-domain and in-domain system, I: in-domain system only.

The target language model was trained on monolingual data from Europarl, as well as the News Crawl and News Discussions English datasets provided for the WMT15 workshop (the same data as was used for estimating term frequencies for the retrieval models) with the `KenLM` toolkit (Heafield et al., 2013; Heafield, 2011).[13]

We optimized the parameters of the translation system for translation quality as measured by IBM BLEU (Papineni et al., 2002) using the Margin Infused Relaxed Algorithm (MIRA) (Crammer and Singer, 2003). For tuning the translation models used for extraction of the hypothesis lists for final evaluation, MIRA was run for 20 iterations on the development set, and the best run was chosen for final testing.

In-Domain Baseline. We also compared our models to a domain-adapted machine translation system. The domain-adapted system was identical to the out-of-domain system, except that it was supplied with additional parallel training data from the image caption domain. For this purpose, we used 29,000 parallel German-English image captions as provided for the WMT16 shared task on multimodal machine translation. The English captions in this dataset belong to the Flickr30k corpus (Rashtchian et al., 2010) and are very similar to those of the MS COCO corpus. The German captions are expert translations. The English captions were also used as additional training data for the target-side language model. We generated k_n- and k_r-best lists of translation candidates using this in-domain baseline system.

[13] `https://kheafield.com/code/kenlm/`

Model	k_n	k_m	k_r	λ
TSR-TXT	300	500	5	$5 \cdot 10^4$
TSR-CNN	300	300	5	$70 \cdot 10^4$
TSR-HCA	300	500	5	$10 \cdot 10^4$

Table 3: Optimized hyperparameter values used in final evaluation.

4.3 Optimization of TSR Hyperparameters

For each of our retrieval models, we performed a step-wise exhaustive search of the hyperparameter space over the four system hyperparameters for IBM BLEU on the development set: The length of the k_n-best list the entries of which are used as queries for retrieval; the number of k_m-best-matching captions retrieved; the length of the final k_r-best list used in reranking; the interpolation weight λ of the relevance score F relative to the translation hypothesis log probability returned by the decoder. The parameter ranges to be explored were determined manually, by examining system output for prototypical examples. Table 3 gives an overview over the hyperparameter values obtained.

For TSR-CNN, we initially set the cutoff distance d to 90.0, after manually inspecting sets of nearest neighbors returned for various maximum distance values. After optimization of retrieval parameters, we performed an exhaustive search from $d = 80.0$ to $d = 100.0$, with step size 1.0 on the development set, while keeping all other hyperparameters fixed, which confirmed out initial choice of $d = 90.0$ as the optimal value.

Explored parameter spaces were identical for all models and each model was evaluated on the test set using its own optimal configuration of hyperparameters.

4.4 Significance Testing

Significance tests on the differences in translation quality were performed using the approximate randomization technique for measuring performance differences of machine translation systems described in Riezler and Maxwell (2005) and implemented by Clark et al. (2011) as part of the `Multeval` toolkit.[14]

[14] `https://github.com/jhclark/multeval`

System	BLEU ↑	p_c	p_t	p_d	p_o
cdec out-dom.	25.5				
cdec in-dom.	29.6				0.00
TSR-TXT	29.7			0.45	0.00
TSR-CNN	**30.6**		0.04	0.02	0.00
TSR-HCA	**30.3**	0.42	0.01	0.00	0.00
System	METEOR ↑	p_c	p_t	p_d	p_o
cdec out-dom.	31.7				
cdec in-dom.	34.0				0.00
TSR-TXT	34.1			0.41	0.00
TSR-CNN	**34.7**		0.00	0.00	0.00
TSR-HCA	**34.4**	0.09	0.00	0.00	0.00
System	TER ↓	p_c	p_t	p_d	p_o
cdec out-dom.	49.3				
cdec in-dom.	46.1				0.00
TSR-TXT	45.8			0.12	0.00
TSR-CNN	**45.1**		0.03	0.00	0.00
TSR-HCA	**45.3**	0.34	0.02	0.00	0.00

Table 4: Metric scores for all systems and their significance levels as reported by Multeval. p_o-values are relative to the cdec out-of-domain baseline, p_d-values are relative to the cdec in-domain baseline, p_t-values are relative to TSR-TXT and p_c-values are relative to TSR-CNN. Best results are reported in **bold** face.[15]

4.5 Experimental Results

Table 4 summarizes the results for all models on an unseen test set of 500 captions. Domain adaptation led to a considerable improvement of +4.1 BLEU and large improvements in terms of METEOR and Translation Edit Rate (TER). We found that the target-side retrieval model enhanced with multimodal pivots from a deep convolutional neural network, TSR-CNN and TSR-HCA, consistently outperformed both the domain-adapted cdec baseline, as well as the text-based target side retrieval model TSR-TXT. These models therefore achieve a performance gain which goes beyond the effect of generic domain-adaptation. The gain in performance for TSR-CNN and TSR-HCA was significant at $p < 0.05$ for BLEU, METEOR, and TER. For all evaluation metrics, the difference between TSR-CNN and TSR-HCA was not significant, demonstrating that retrieval using our CNN-derived distance metric could match retrieval based the human object category annotations.

[15] A baseline for which a random hypothesis was chosen from the top-5 candidates of the in-domain system lies between the other two baseline systems: 27.5 / 33.3 / 47.7 (BLEU / METEOR / TER).

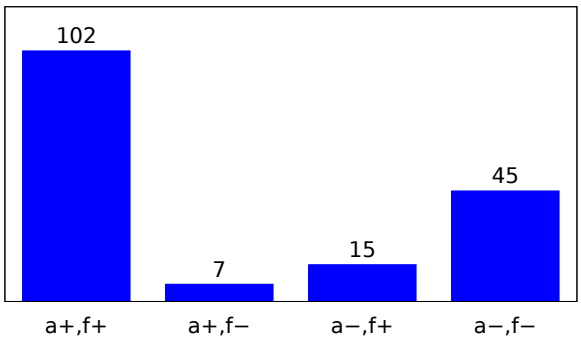


Figure 2: Results of the human pairwise preference ranking experiment, given as the joint distribution of both rankings: $a+$ denotes preference for TSR-CNN in terms of accuracy, $f+$ in terms of fluency; $a-$ denotes preference for the in-domain baseline in terms of accuracy, $f-$ in terms of fluency.

The text-based retrieval baseline TSR-TXT never significantly outperformed the in-domain cdec baseline, but there were slight nominal improvements in terms of BLEU, METEOR and TER. This finding is actually consistent with Wäschle and Riezler (2015) who report performance gains for text-based, target side retrieval models only on highly technical, narrow-domain corpora and even report performance degradation on medium-diversity corpora such as Europarl. Our experiments show that it is the addition of visual similarity information by incorporation of multimodal pivots into the image-enhanced models TSR-CNN and TSR-HCA which makes such techniques effective on MS COCO, thus upholding our hypothesis that visual information can be exploited for improvement of caption translation.

4.6 Human Evaluation

The in-domain baseline and TSR-CNN differed in their output in 169 out of 500 cases on the test set. These 169 cases were presented to a human judge alongside the German source captions in a double-blinded pairwise preference ranking experiment. The order of presentation was randomized for the two systems. The judge was asked to rank fluency and accuracy of the translations independently. The results are given in Figure 2. Overall, there was a clear preference for the output of TSR-CNN.

4.7 Examples

Table 5 shows example translations produced by both cdec baselines, TSR-TXT, TSR-CNN, and TSR-HCA, together with source caption, image, and reference translation. The visual information induced by target side captions of pivot images allows a disambiguation of translation alternatives such as "skirt" versus "rock (music)" for the German "Rock", "pole" versus "mast" for the German "Masten", and is able to repair mistranslations such as "foot" instead of "mouth" for the German "Maul".

5 Conclusions and Further Work

We demonstrated that the incorporation of multimodal pivots into a target-side retrieval model improved SMT performance compared to a strong in-domain baseline in terms of BLEU, METEOR and TER on our parallel dataset derived from MS COCO. The gain in performance was comparable between a distance metric based on a deep convolutional network and one based on human object category annotations, demonstrating the effectiveness of the CNN-derived distance measure. Using our approach, SMT can, in certain cases, profit from multimodal context information. Crucially, this is possible without using large amounts of in-domain parallel text data, but instead using large amounts of monolingual image captions that are more readily available.

Learning semantically informative distance metrics using deep learning techniques is an area under active investigation (Wu et al., 2013; Wang et al., 2014; Wang et al., 2015). Despite the fact that our simple distance metric performed comparably to human object annotations, using such high-level semantic distance metrics for caption translation by multimodal pivots is a promising avenue for further research.

The results were achieved on one language pair (German-English) and one corpus (MS COCO) only. As with all retrieval-based methods, generalized statements about the relative performance on corpora of various domains, sizes and qualities are difficult to substantiate. This problem is aggravated in the multimodal case, since the relevance of captions with respect to images varies greatly between different corpora (Hodosh et al., 2013). In future work, we plan to evaluate our approach in more naturalistic settings, such machine translation for captions in online multimedia repositories

Image:	
Source:	Eine Person in einem Anzug und Krawatte und einem Rock.
cdec out-dom:	a person in a suit and tie and a rock .
cdec in-dom:	a person in a suit and tie and a rock .
TSR-TXT:	a person in a suit and tie and a rock .
TSR-CNN:	a person in a suit and tie and a skirt .
TSR-HCA:	a person in a suit and tie and a rock .
Reference:	a person wearing a suit and tie and a skirt
Image:	
Source:	Ein Masten mit zwei Ampeln für Autofahrer.
cdec out-dom:	a mast with two lights for drivers .
cdec in-dom:	a mast with two lights for drivers .
TSR-TXT:	a mast with two lights for drivers .
TSR-CNN:	a pole with two lights for drivers .
TSR-HCA:	a pole with two lights for drivers .
Reference:	a pole has two street lights on it for drivers .
Image:	
Source:	Ein Hund auf einer Wiese mit einem Frisbee im Maul.
cdec out-dom:	a dog on a lawn with a frisbee in the foot .
cdec in-dom:	a dog with a frisbee in a grassy field .
TSR-TXT:	a dog with a frisbee in a grassy field .
TSR-CNN:	a dog in a grassy field with a frisbee in its mouth .
TSR-HCA:	a dog with a frisbee in a grassy field .
Reference:	a dog in a field with a frisbee in its mouth

Table 5: Examples for improved caption translation by multimodal feedback.

such as Wikimedia Commons[16] and digitized art catalogues, as well as e-commerce localization.

A further avenue of future research is improving models such as that presented in Elliott et al. (2015) by crucial components of neural MT such as "attention mechanisms". For example, the attention mechanism of Bahdanau et al. (2015) serves as a soft alignment that helps to guide the translation process by influencing the sequence in which source tokens are translated. A similar mechanism is used in Xu et al. (2015) to decide which part of the image should influence which part of the generated caption. Combining these two types of attention mechanisms in a neural caption translation model is a natural next step in caption translation. While this is beyond the scope of this work, our models should provide an informative baseline against which to evaluate such methods.

Acknowledgments

This research was supported in part by DFG grant RI-2221/2-1 "Grounding Statistical Machine Translation in Perception and Action", and by an Amazon Academic Research Award (AARA) "Multimodal Pivots for Low Resource Machine Translation in E-Commerce Localization".

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, California, USA.

Shane Bergsma and Benjamin Van Durme. 2011. Learning bilingual lexicons using the visual similarity of labeled web images. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, Barcelona, Spain.

Iacer Calixto, Teófilo de Compos, and Lucia Specia. 2012. Images as context in statistical machine translation. In *Proceedings of the Workshop on Vision and Language (VL)*, Sheffield, England, United Kingdom.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Jonathan Clark, Chris Dyer, Alon Lavie, and Noah Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proceedings of the Association for Computational Lingustics (ACL)*, Portland, Oregon, USA.

Koby Crammer and Yoram Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

Chris Dyer, Adam Lopez, Juri Ganitkevitch, Johnathan Weese, Ferhan Ture, Phil Blunsom, Hendra Setiawan, Vladimir Eidelman, and Philip Resnik. 2010. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the Association for Computational Linguistics (ACL)*, Uppsala, Sweden.

Chris Dyer. 2009. Using a maximum entropy model to build segmentation lattices for mt. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL HLT)*, Boulder, Colorado, USA.

Desmond Elliott, Stella Frank, and Eva Hasler. 2015. Multi-language image description with neural sequence models. *CoRR*, abs/1510.04709.

Hao Fang, Li Deng, Margaret Mitchell, Saurabh Gupta, Piotr Dollar, John C. Platt, Forrest Iandola, Jianfeng Gao, C. Lawrence Zitnick, Rupesh K. Srivastava, Xiaodeng He, and Geoffrey Zweit. 2015. From captions to visual concepts and back. In *In Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA.

Francis Ferraro, Nasrin Mostafazadeh, Ting-Hao (Kenneth) Huang, Lucy Vanderwende, Jacob Devlin, Michel Galley, and Margaret Mitchell. 2015. A survey of current datasets for vision and language research. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal.

Ruka Funaki and Hideki Nakayama. 2015. Image-mediated learning for zero-shot cross-lingual document retrieval. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal.

Michael Grubinger, Paul Clough, Henning Müller, and Thomas Deselaers. 2006. The IAPR TC-12 benchmark: A new evaluatioin resource for visual information systems. In *In Proceedings of LREC*, Genova, Italy.

Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. 2013. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria.

Kenneth Heafield. 2011. KenLM: faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation (WMT)*, Edinburgh, Scotland, United Kingdom.

[16] https://commons.wikimedia.org/wiki/Main_Page

Micah Hodosh, Peter Young, and Julia Hockenmaier. 2013. Framing image description as a ranking task: Data, models and evaluation metrics. *Journal of Artificial Intelligence Research*, 47:853–899.

Andrey Karpathy and Li Fei-Fei. 2015. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, Massachusetts, USA.

Douwe Kiela, Ivan Vulić, and Stephen Clark. 2015. Visual bilingual lexicon induction with transferred convnet features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the Machine Translation Summit*, Phuket, Thailand.

Girish Kulkarni, Visruth Premraj, Sagnik Dhar, Siming Li, Yejin Choi, Alexander C Berg, and Tamara L Berg. 2011. Baby talk: Understanding and generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Colorado Springs, Colorado, USA.

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: common objects in context. *Computing Research Repository*, abs/1405.0312.

Kishore Papineni, Salim Roukos, Todd Ard, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, Philadelphia, Pennsylvania, USA.

Cyrus Rashtchian, Peter Young, Micah Hodosh, and Julia Hockenmaier. 2010. Collecting image annotations using amazon's mechanical turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, Los Angeles, California, USA.

Stefan Riezler and John Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Methods for MT and Summarization (MTSE) at the 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Michigan, USA.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. 2014. Imagenet large scale visual recognition challenge. *Computing Research Repository*, abs/1409.0575.

Carina Silberer and Mirella Lapata. 2014. Learning grounded meaning representations with autoencoders. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL)*, Baltimore, Maryland, USA.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, San Diego, CA.

Jason Smith, Herve Saint-Amand, Magdalena Plamada, Philipp Koehn, Chris Callison-Burch, and Adam Lopez. 2013. Dirt cheap web-scale parallel text from the Common Crawl. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL)*, Sofia, Bulgaria.

Richard Socher, Andrej Karpathy, Quoc V. Le, Christopher D. Manning, and Andrew Y. Ng. 2014. Grounded compositional semantics for finding and describing images with sentences. *Transactions of the Association for Computational Linguistics*, 2(1):207–218.

Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston,Massachusetts, USA.

Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. 2014. Learning fine-grained image similarity with deep ranking. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Columbus, Ohio, USA.

Zhaowen Wang, Jianchao Yang, Zhe Lin, Jonathan Brandt, Shiyu Chang, and Thomas Huang. 2015. Scalable similarity learning using large margin neighborhood embedding. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, Washington, DC, USA.

Katharina Wäschle and Stefan Riezler. 2015. Integrating a large, monolingual corpus as translation memory into statistical machine translation. In *Proceedings of the 18th Annual Conference of the European Association for Machine Translation (EAMT)*, Antalya, Turkey.

Pengcheng Wu, Steven C.H. Hoi, Hao Xia, Peilin Zhao, Dayong Wang, and Chunyan Miao. 2013. Online multimodal deep similarity learning with application to image retrieval. In *Proceedings of the 21st ACM International Conference on Multimedia*, Barcelona, Spain.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, Lille, France.

Harnessing Deep Neural Networks with Logic Rules

Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, Eric P. Xing
School of Computer Science
Carnegie Mellon University
{zhitingh,xuezhem,liu,epxing}@cs.cmu.edu, hovy@cmu.edu

Abstract

Combining deep neural networks with structured logic rules is desirable to harness flexibility and reduce uninterpretability of the neural models. We propose a general framework capable of enhancing various types of neural networks (e.g., CNNs and RNNs) with declarative first-order logic rules. Specifically, we develop an iterative distillation method that transfers the structured information of logic rules into the weights of neural networks. We deploy the framework on a CNN for sentiment analysis, and an RNN for named entity recognition. With a few highly intuitive rules, we obtain substantial improvements and achieve state-of-the-art or comparable results to previous best-performing systems.

1 Introduction

Deep neural networks provide a powerful mechanism for learning patterns from massive data, achieving new levels of performance on image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), machine translation (Bahdanau et al., 2014), playing strategic board games (Silver et al., 2016), and so forth.

Despite the impressive advances, the widely-used DNN methods still have limitations. The high predictive accuracy has heavily relied on large amounts of labeled data; and the purely data-driven learning can lead to uninterpretable and sometimes counter-intuitive results (Szegedy et al., 2014; Nguyen et al., 2015). It is also difficult to encode human intention to guide the models to capture desired patterns, without expensive direct supervision or ad-hoc initialization.

On the other hand, the cognitive process of human beings have indicated that people learn not only from concrete examples (as DNNs do) but also from different forms of general knowledge and rich experiences (Minksy, 1980; Lake et al., 2015). *Logic rules* provide a flexible declarative language for communicating high-level cognition and expressing structured knowledge. It is therefore desirable to integrate logic rules into DNNs, to transfer human intention and domain knowledge to neural models, and regulate the learning process.

In this paper, we present a framework capable of enhancing general types of neural networks, such as convolutional networks (CNNs) and recurrent networks (RNNs), on various tasks, with logic rule knowledge. Combining symbolic representations with neural methods have been considered in different contexts. Neural-symbolic systems (Garcez et al., 2012) construct a network from a given rule set to execute reasoning. To exploit *a priori* knowledge in general neural architectures, recent work augments each raw data instance with useful features (Collobert et al., 2011), while network training, however, is still limited to instance-label supervision and suffers from the same issues mentioned above. Besides, a large variety of structural knowledge cannot be naturally encoded in the feature-label form.

Our framework enables a neural network to learn simultaneously from labeled instances as well as logic rules, through an *iterative rule knowledge distillation* procedure that transfers the structured information encoded in the logic rules into the network parameters. Since the general logic rules are complementary to the specific data labels, a natural "side-product" of the integration is the support for semi-supervised learning where unlabeled data is used to better absorb the logical knowledge. Methodologically, our approach can be seen as a combination of the knowledge distillation (Hinton et al., 2015; Bucilu et al., 2006) and the posterior regularization (PR) method (Ganchev et al., 2010).

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2410–2420,
Berlin, Germany, August 7-12, 2016.

In particular, at each iteration we adapt the posterior constraint principle from PR to construct a rule-regularized *teacher*, and train the *student* network of interest to imitate the predictions of the teacher network. We leverage soft logic to support flexible rule encoding.

We apply the proposed framework on both CNN and RNN, and deploy on the task of sentiment analysis (SA) and named entity recognition (NER), respectively. With only a few (one or two) very intuitive rules, both the distilled networks and the joint teacher networks strongly improve over their basic forms (without rules), and achieve better or comparable performance to state-of-the-art models which typically have more parameters and complicated architectures.

To the best of our knowledge, this is the first work to integrate logic rules with general workhorse types of deep neural networks in a principled framework. The encouraging results indicate our method can be potentially useful for incorporating richer types of human knowledge, and improving other application domains.

2 Related Work

Combination of logic rules and neural networks has been considered in different contexts. Neural-symbolic systems (Garcez et al., 2012), such as KBANN (Towell et al., 1990) and CILP++ (França et al., 2014), construct network architectures from given rules to perform reasoning and knowledge acquisition. A related line of research, such as Markov logic networks (Richardson and Domingos, 2006), derives probabilistic graphical models (rather than neural networks) from the rule set.

With the recent success of deep neural networks in a vast variety of application domains, it is increasingly desirable to incorporate structured logic knowledge into general types of networks to harness flexibility and reduce uninterpretability. Recent work that trains on extra features from domain knowledge (Collobert et al., 2011), while producing improved results, does not go beyond the data-label paradigm. Kulkarni et al. (2015) uses a specialized training procedure with careful ordering of training instances to obtain an interpretable neural layer of an image network. Karaletsos et al. (2016) develops a generative model jointly over data-labels and similarity knowledge expressed in triplet format to learn improved disentangled representations.

Though there do exist general frameworks that allow encoding various structured constraints on latent variable models (Ganchev et al., 2010; Zhu et al., 2014; Liang et al., 2009), they either are not directly applicable to the NN case, or could yield inferior performance as in our empirical study. Liang et al. (2008) transfers predictive power of pre-trained structured models to unstructured ones in a pipelined fashion.

Our proposed approach is distinct in that we use an iterative rule distillation process to effectively transfer rich structured knowledge, expressed in the declarative first-order logic language, into parameters of general neural networks. We show that the proposed approach strongly outperforms an extensive array of other either ad-hoc or general integration methods.

3 Method

In this section we present our framework which encapsulates the logical structured knowledge into a neural network. This is achieved by forcing the network to emulate the predictions of a rule-regularized teacher, and evolving both models iteratively throughout training (section 3.2). The process is agnostic to the network architecture, and thus applicable to general types of neural models including CNNs and RNNs. We construct the teacher network in each iteration by adapting the posterior regularization principle in our logical constraint setting (section 3.3), where our formulation provides a closed-form solution. Figure 1 shows an overview of the proposed framework.

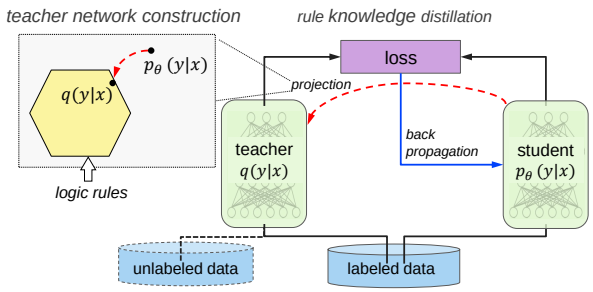


Figure 1: Framework Overview. At each iteration, the teacher network is obtained by projecting the student network to a rule-regularized subspace (red dashed arrow); and the student network is updated to balance between emulating the teacher's output and predicting the true labels (black/blue solid arrows).

3.1 Learning Resources: Instances and Rules

Our approach allows neural networks to learn from both specific examples and general rules. Here we give the settings of these "learning resources".

Assume we have input variable $\boldsymbol{x} \in \mathcal{X}$ and target variable $\boldsymbol{y} \in \mathcal{Y}$. For clarity, we focus on K-way classification, where $\mathcal{Y} = \Delta^K$ is the K-dimensional probability simplex and $\boldsymbol{y} \in \{0,1\}^K \subset \mathcal{Y}$ is a one-hot encoding of the class label. However, our method specification can straightforwardly be applied to other contexts such as regression and sequence learning (e.g., NER tagging, which is a sequence of classification decisions). The training data $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^N$ is a set of instantiations of $(\boldsymbol{x}, \boldsymbol{y})$.

Further consider a set of first-order logic (FOL) rules with confidences, denoted as $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^L$, where R_l is the lth rule over the input-target space $(\mathcal{X}, \mathcal{Y})$, and $\lambda_l \in [0, \infty]$ is the confidence level with $\lambda_l = \infty$ indicating a hard rule, i.e., all groundings are required to be true (=1). Here a grounding is the logic expression with all variables being instantiated. Given a set of examples $(\boldsymbol{X}, \boldsymbol{Y}) \subset (\mathcal{X}, \mathcal{Y})$ (e.g., a minibatch from $\mathcal{D}$), the set of groundings of R_l are denoted as $\{r_{lg}(\boldsymbol{X}, \boldsymbol{Y})\}_{g=1}^{G_l}$. In practice a rule grounding is typically relevant to only a single or subset of examples, though here we give the most general form on the entire set.

We encode the FOL rules using soft logic (Bach et al., 2015) for flexible encoding and stable optimization. Specifically, soft logic allows continuous truth values from the interval $[0, 1]$ instead of $\{0, 1\}$, and the Boolean logic operators are reformulated as:

$$\begin{aligned} &A \& B = \max\{A + B - 1, 0\} \\ &A \vee B = \min\{A + B, 1\} \\ &A_1 \wedge \cdots \wedge A_N = \textstyle\sum_i A_i / N \\ &\neg A = 1 - A \end{aligned} \tag{1}$$

Here $\&$ and $\wedge$ are two different approximations to logical conjunction (Foulds et al., 2015): $\&$ is useful as a selection operator (e.g., $A \& B = B$ when $A = 1$, and $A \& B = 0$ when $A = 0$), while $\wedge$ is an averaging operator.

3.2 Rule Knowledge Distillation

A neural network defines a conditional probability $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ by using a softmax output layer that produces a K-dimensional soft prediction vector denoted as $\boldsymbol{\sigma}_\theta(\boldsymbol{x})$. The network is parameterized by weights $\boldsymbol{\theta}$. Standard neural network training has been to iteratively update $\boldsymbol{\theta}$ to produce the correct labels of training instances. To integrate the information encoded in the rules, we propose to train the network to also imitate the outputs of a rule-regularized projection of $p_\theta(\boldsymbol{y}|\boldsymbol{x})$, denoted as $q(\boldsymbol{y}|\boldsymbol{x})$, which explicitly includes rule constraints as regularization terms. In each iteration q is constructed by projecting p_θ into a subspace constrained by the rules, and thus has desirable properties. We present the construction in the next section. The prediction behavior of q reveals the information of the regularized subspace and structured rules. Emulating the q outputs serves to transfer this knowledge into p_θ. The new objective is then formulated as a balancing between imitating the soft predictions of q and predicting the true hard labels:

$$\begin{aligned} \boldsymbol{\theta}^{(t+1)} = \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{n=1}^{N} &(1 - \pi) \ell(\boldsymbol{y}_n, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)) \\ &+ \pi \ell(\boldsymbol{s}_n^{(t)}, \boldsymbol{\sigma}_\theta(\boldsymbol{x}_n)), \end{aligned} \tag{2}$$

where ℓ denotes the loss function selected according to specific applications (e.g., the cross entropy loss for classification); $\boldsymbol{s}_n^{(t)}$ is the soft prediction vector of q on $\boldsymbol{x}_n$ at iteration t; and π is the imitation parameter calibrating the relative importance of the two objectives.

A similar imitation procedure has been used in other settings such as model compression (Bucilu et al., 2006; Hinton et al., 2015) where the process is termed *distillation*. Following them we call $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ the "student" and $q(\boldsymbol{y}|\boldsymbol{x})$ the "teacher", which can be intuitively explained in analogous to human education where a teacher is aware of systematic general rules and she instructs students by providing her solutions to particular questions (i.e., the soft predictions). An important difference from previous distillation work, where the teacher is obtained beforehand and the student is trained thereafter, is that our teacher and student are learned simultaneously during training.

Though it is possible to combine a neural network with rule constraints by projecting the network to the rule-regularized subspace after it is fully trained as before with only data-label instances, or by optimizing projected network directly, we found our iterative teacher-student distillation approach provides a much superior performance, as shown in the experiments. Moreover, since p_θ distills the rule information into the

weights $\boldsymbol{\theta}$ instead of relying on explicit rule representations, we can use p_θ for predicting new examples at test time when the rule assessment is expensive or even unavailable (i.e., the *privileged information* setting (Lopez-Paz et al., 2016)) while still enjoying the benefit of integration. Besides, the second loss term in Eq.(2) can be augmented with rich unlabeled data in addition to the labeled examples, which enables *semi-supervised* learning for better absorbing the rule knowledge.

3.3 Teacher Network Construction

We now proceed to construct the teacher network $q(\boldsymbol{y}|\boldsymbol{x})$ at each iteration from $p_\theta(\boldsymbol{y}|\boldsymbol{x})$. The iteration index t is omitted for clarity. We adapt the posterior regularization principle in our logic constraint setting. Our formulation ensures a closed-form solution for q and thus avoids any significant increases in computational overhead.

Recall the set of FOL rules $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^L$. Our goal is to find the optimal q that fits the rules while at the same time staying close to p_θ. For the first property, we apply a commonly-used strategy that imposes the rule constraints on q through an expectation operator. That is, for each rule (indexed by l) and each of its groundings (indexed by g) on $(\boldsymbol{X}, \boldsymbol{Y})$, we expect $\mathbb{E}_{q(\boldsymbol{Y}|\boldsymbol{X})}[r_{lg}(\boldsymbol{X}, \boldsymbol{Y})] = 1$, with confidence λ_l. The constraints define a rule-regularized space of all valid distributions. For the second property, we measure the closeness between q and p_θ with KL-divergence, and wish to minimize it. Combining the two factors together and further allowing slackness for the constraints, we finally get the following optimization problem:

$$\begin{aligned} \min_{q, \boldsymbol{\xi} \geq 0} \ & \mathrm{KL}(q(\boldsymbol{Y}|\boldsymbol{X}) \| p_\theta(\boldsymbol{Y}|\boldsymbol{X})) + C \sum\nolimits_{l, g_l} \xi_{l, g_l} \\ \text{s.t. } & \lambda_l (1 - \mathbb{E}_q[r_{l, g_l}(\boldsymbol{X}, \boldsymbol{Y})]) \leq \xi_{l, g_l} \\ & g_l = 1, \ldots, G_l, \ \ l = 1, \ldots, L, \end{aligned} \tag{3}$$

where $\xi_{l,g_l} \geq 0$ is the slack variable for respective logic constraint; and C is the regularization parameter. The problem can be seen as projecting p_θ into the constrained subspace. The problem is convex and can be efficiently solved in its dual form with closed-form solutions. We provide the detailed derivation in the supplementary materials and directly give the solution here:

$$q^*(\boldsymbol{Y}|\boldsymbol{X}) \propto p_\theta(\boldsymbol{Y}|\boldsymbol{X}) \exp\left\{ -\sum_{l, g_l} C \lambda_l (1 - r_{l, g_l}(\boldsymbol{X}, \boldsymbol{Y})) \right\} \tag{4}$$

Intuitively, a strong rule with large λ_l will lead to low probabilities of predictions that fail to meet the constraints. We discuss the computation of the normalization factor in section 3.4.

Our framework is related to the posterior regularization (PR) method (Ganchev et al., 2010) which places constraints over model posterior in unsupervised setting. In classification, our optimization procedure is analogous to the modified EM algorithm for PR, by using cross-entropy loss in Eq.(2) and evaluating the second loss term on unlabeled data differing from $\mathcal{D}$, so that Eq.(4) corresponds to the E-step and Eq.(2) is analogous to the M-step. This sheds light from another perspective on why our framework would work. However, we found in our experiments (section 5) that to produce strong performance it is crucial to use the same labeled data $\boldsymbol{x}_n$ in the two losses of Eq.(2) so as to form a direct trade-off between imitating soft predictions and predicting correct hard labels.

3.4 Implementations

The procedure of iterative distilling optimization of our framework is summarized in Algorithm 1.

During training we need to compute the soft predictions of q at each iteration, which is straightforward through direct enumeration if the rule constraints in Eq.(4) are factored in the same way as the base neural model p_θ (e.g., the "but"-rule of sentiment classification in section 4.1). If the constraints introduce additional dependencies, e.g., bigram dependency as the transition rule in the NER task (section 4.2), we can use dynamic programming for efficient computation. For higher-order constraints (e.g., the listing rule in NER), we approximate through Gibbs sampling that iteratively samples from $q(\boldsymbol{y}_i|\boldsymbol{y}_{-i}, \boldsymbol{x})$ for each position i. If the constraints span multiple instances, we group the relevant instances in minibatches for joint inference (and randomly break some dependencies when a group is too large). Note that calculating the soft predictions is efficient since only one NN forward pass is required to compute the base distribution $p_\theta(\boldsymbol{y}|\boldsymbol{x})$ (and few more, if needed, for calculating the truth values of relevant rules).

p v.s. q at Test Time At test time we can use either the distilled student network p, or the teacher network q after a final projection. Our empirical results show that both models substantially improve over the base network that is trained with only data-label instances. In general q performs better than p. Particularly, q is more suitable when the logic rules introduce additional dependencies (e.g., span-

Algorithm 1 Harnessing NN with Rules

Input: The training data $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$,
The rule set $\mathcal{R} = \{(R_l, \lambda_l)\}_{l=1}^{L}$,
Parameters: π – imitation parameter
C – regularization strength

1: Initialize neural network parameter $\boldsymbol{\theta}$
2: **repeat**
3: Sample a minibatch $(\boldsymbol{X}, \boldsymbol{Y}) \subset \mathcal{D}$
4: Construct teacher network q with Eq.(4)
5: Transfer knowledge into p_θ by updating $\boldsymbol{\theta}$ with Eq.(2)
6: **until** convergence

Output: Distill student network p_θ and teacher network q

ning over multiple examples), requiring joint inference. In contrast, as mentioned above, p is more lightweight and efficient, and useful when rule evaluation is expensive or impossible at prediction time. Our experiments compare the performance of p and q extensively.

Imitation Strength π The imitation parameter π in Eq.(2) balances between emulating the teacher soft predictions and predicting the true hard labels. Since the teacher network is constructed from p_θ, which, at the beginning of training, would produce low-quality predictions, we thus favor predicting the true labels more at initial stage. As training goes on, we gradually bias towards emulating the teacher predictions to effectively distill the structured knowledge. Specifically, we define $\pi^{(t)} = \min\{\pi_0, 1 - \alpha^t\}$ at iteration $t \geq 0$, where $\alpha \leq 1$ specifies the speed of decay and $\pi_0 < 1$ is a lower bound.

4 Applications

We have presented our framework that is general enough to improve various types of neural networks with rules, and easy to use in that users are allowed to impose their knowledge and intentions through the declarative first-order logic. In this section we illustrate the versatility of our approach by applying it on two workhorse network architectures, i.e., convolutional network and recurrent network, on two representative applications, i.e., *sentence-level sentiment analysis* which is a classification problem, and *named entity recognition* which is a sequence learning problem.

For each task, we first briefly describe the base neural network. Since we are not focusing on tuning network architectures, we largely use the same or similar networks to previous successful neural models. We then design the linguistically-motivated rules to be integrated.

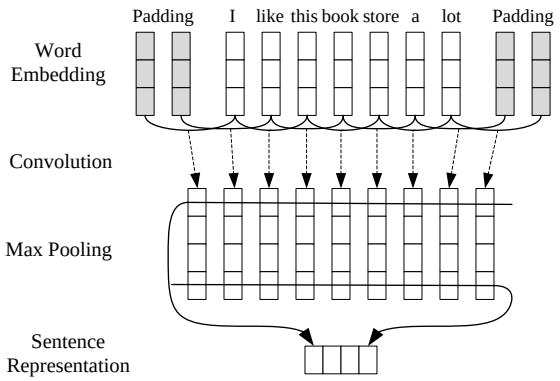


Figure 2: The CNN architecture for sentence-level sentiment analysis. The sentence representation vector is followed by a fully-connected layer with softmax output activation, to output sentiment predictions.

4.1 Sentiment Classification

Sentence-level sentiment analysis is to identify the sentiment (e.g., positive or negative) underlying an individual sentence. The task is crucial for many opinion mining applications. One challenging point of the task is to capture the contrastive sense (e.g., by conjunction "but") within a sentence.

Base Network We use the single-channel convolutional network proposed in (Kim, 2014). The simple model has achieved compelling performance on various sentiment classification benchmarks. The network contains a convolutional layer on top of word vectors of a given sentence, followed by a max-over-time pooling layer and then a fully-connected layer with softmax output activation. A convolution operation is to apply a filter to word windows. Multiple filters with varying window sizes are used to obtain multiple features. Figure 2 shows the network architecture.

Logic Rules One difficulty for the plain neural network is to identify contrastive sense in order to capture the dominant sentiment precisely. The conjunction word "but" is one of the strong indicators for such sentiment changes in a sentence, where the sentiment of clauses following "but" generally dominates. We thus consider sentences S with an "A-but-B" structure, and expect the sentiment of the whole sentence to be consistent with the sentiment of clause B. The logic rule is written as:

$$\begin{aligned} &\text{has-'A-but-B'-structure}(S) \Rightarrow \\ &(\mathbf{1}(y = +) \Rightarrow \boldsymbol{\sigma}_\theta(B)_+ \ \wedge \ \boldsymbol{\sigma}_\theta(B)_+ \Rightarrow \mathbf{1}(y = +)), \end{aligned} \tag{5}$$

where $\mathbf{1}(\cdot)$ is an indicator function that takes 1 when its argument is true, and 0 otherwise; class '+' represents 'positive'; and $\boldsymbol{\sigma}_\theta(B)_+$ is the element of $\boldsymbol{\sigma}_\theta(B)$ for class '+'. By Eq.(1), when S has the 'A-but-B' structure, the truth value of the above logic rule equals to $(1 + \boldsymbol{\sigma}_\theta(B)_+)/2$ when $y = +$, and $(2 - \boldsymbol{\sigma}_\theta(B)_+)/2$ otherwise [1]. Note that here we assume two-way classification (i.e., positive and negative), though it is straightforward to design rules for finer grained sentiment classification.

4.2 Named Entity Recognition

NER is to locate and classify elements in text into entity categories such as "persons" and "organizations". It is an essential first step for downstream language understanding applications. The task assigns to each word a named entity tag in an "X-Y" format where X is one of BIEOS (Beginning, Inside, End, Outside, and Singleton) and Y is the entity category. A valid tag sequence has to follow certain constraints by the definition of the tagging scheme. Besides, text with structures (e.g., lists) within or across sentences can usually expose some consistency patterns.

Base Network The base network has a similar architecture with the bi-directional LSTM recurrent network (called BLSTM-CNN) proposed in (Chiu and Nichols, 2015) for NER which has outperformed most of previous neural models. The model uses a CNN and pre-trained word vectors to capture character- and word-level information, respectively. These features are then fed into a bi-directional RNN with LSTM units for sequence tagging. Compared to (Chiu and Nichols, 2015) we omit the character type and capitalization features, as well as the additive transition matrix in the output layer. Figure 3 shows the network architecture.

Logic Rules The base network largely makes independent tagging decisions at each position, ignoring the constraints on successive labels for a valid tag sequence (e.g., I-ORG cannot follow B-PER). In contrast to recent work (Lample et al., 2016) which adds a conditional random field (CRF) to capture bi-gram dependencies between outputs, we instead apply logic rules which does not introduce extra parameters to learn. An example rule is:

$$\text{equal}(y_{i-1}, \text{I-ORG}) \Rightarrow \neg\, \text{equal}(y_i, \text{B-PER}) \qquad (6)$$

[1] Replacing $\wedge$ with & in Eq.(5) leads to a probably more intuitive rule which takes the value $\boldsymbol{\sigma}_\theta(B)_+$ when $y = +$, and $1 - \boldsymbol{\sigma}_\theta(B)_+$ otherwise.

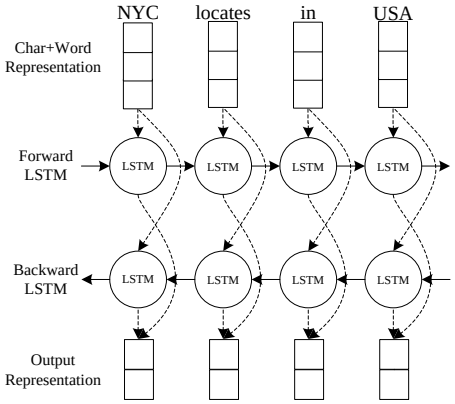


Figure 3: The architecture of the bidirectional LSTM recurrent network for NER. The CNN for extracting character representation is omitted.

The confidence levels are set to ∞ to prevent any violation.

We further leverage the *list* structures within and across sentences of the same documents. Specifically, named entities at corresponding positions in a list are likely to be in the same categories. For instance, in "1. Juventus, 2. Barcelona, 3. ..." we know "Barcelona" must be an organization rather than a location, since its counterpart entity "Juventus" is an organization. We describe our simple procedure for identifying lists and counterparts in the supplementary materials. The logic rule is encoded as:

$$\text{is-counterpart}(X, A) \Rightarrow 1 - \|c(\boldsymbol{e}_y) - c(\boldsymbol{\sigma}_\theta(A))\|_2, \quad (7)$$

where $\boldsymbol{e}_y$ is the one-hot encoding of y (the class prediction of X); $c(\cdot)$ collapses the probability mass on the labels with the same categories into a single probability, yielding a vector with length equaling to the number of categories. We use ℓ_2 distance as a measure for the closeness between predictions of X and its counterpart A. Note that the distance takes value in $[0, 1]$ which is a proper soft truth value. The list rule can span multiple sentences (within the same document). We found the teacher network q that enables explicit joint inference provides much better performance over the distilled student network p (section 5).

5 Experiments

We validate our framework by evaluating its applications of sentiment classification and named entity recognition on a variety of public benchmarks. By integrating the simple yet effective rules with

	Model	SST2	MR	CR
1	CNN (Kim, 2014)	87.2	81.3±0.1	84.3±0.2
2	CNN-Rule-p	88.8	81.6±0.1	85.0±0.3
3	CNN-Rule-q	89.3	**81.7±0.1**	**85.3±0.3**
4	MGNC-CNN (Zhang et al., 2016)	88.4	–	–
5	MVCNN (Yin and Schutze, 2015)	**89.4**	–	–
6	CNN-multichannel (Kim, 2014)	88.1	81.1	85.0
7	Paragraph-Vec (Le and Mikolov, 2014)	87.8	–	–
8	CRF-PR (Yang and Cardie, 2014)	–	–	82.7
9	RNTN (Socher et al., 2013)	85.4	–	–
10	G-Dropout (Wang and Manning, 2013)	–	79.0	82.1

Table 1: Accuracy (%) of Sentiment Classification. Row 1, CNN (Kim, 2014) is the base network corresponding to the "CNN-non-static" model in (Kim, 2014). Rows 2-3 are the networks enhanced by our framework: CNN-Rule-p is the student network and CNN-Rule-q is the teacher network. For MR and CR, we report the average accuracy±one standard deviation using 10-fold cross validation.

the base networks, we obtain substantial improvements on both tasks and achieve state-of-the-art or comparable results to previous best-performing systems. Comparison with a diverse set of other rule integration methods demonstrates the unique effectiveness of our framework. Our approach also shows promising potentials in the semi-supervised learning and sparse data context.

Throughout the experiments we set the regularization parameter to $C = 400$. In sentiment classification we set the imitation parameter to $\pi^{(t)} = 1 - 0.9^t$, while in NER $\pi^{(t)} = \min\{0.9, 1 - 0.9^t\}$ to downplay the noisy listing rule. The confidence levels of rules are set to $\lambda_l = 1$, except for hard constraints whose confidence is ∞. For neural network configuration, we largely followed the reference work, as specified in the following respective sections. All experiments were performed on a Linux machine with eight 4.0GHz CPU cores, one Tesla K40c GPU, and 32GB RAM. We implemented neural networks using Theano [2], a popular deep learning platform.

5.1 Sentiment Classification

5.1.1 Setup

We test our method on a number of commonly used benchmarks, including **1) SST2**, Stanford Sentiment Treebank (Socher et al., 2013) which contains 2 classes (negative and positive), and 6920/872/1821 sentences in the train/dev/test sets respectively. Following (Kim, 2014) we train models on both sentences and phrases since all labels are provided. **2) MR** (Pang and Lee, 2005), a set of 10,662 one-sentence movie reviews with negative or positive sentiment. **3) CR** (Hu and Liu, 2004), customer reviews of various products, containing 2 classes and 3,775 instances. For MR and CR, we use 10-fold cross validation as in previous work. In each of the three datasets, around 15% sentences contains the word "but".

For the base neural network we use the "non-static" version in (Kim, 2014) with the exact same configurations. Specifically, word vectors are initialized using word2vec (Mikolov et al., 2013) and fine-tuned throughout training, and the neural parameters are trained using SGD with the Adadelta update rule (Zeiler, 2012).

5.1.2 Results

Table 1 shows the sentiment classification performance. Rows 1-3 compare the base neural model with the models enhanced by our framework with the "but"-rule (Eq.(5)). We see that our method provides a strong boost on accuracy over all three datasets. The teacher network q further improves over the student network p, though the student network is more widely applicable in certain contexts as discussed in sections 3.2 and 3.4. Rows 4-10 show the accuracy of recent top-performing methods. On the MR and CR datasets, our model outperforms all the baselines. On SST2, MVCNN (Yin and Schutze, 2015) (Row 5) is the only system that shows a slightly better result than ours. Their neural network has combined diverse sets of pre-trained word embeddings (while we use only word2vec) and contained more neural layers and parameters than our model.

To further investigate the effectiveness of our framework in integrating structured rule knowledge, we compare with an extensive array of other

[2] http://deeplearning.net/software/theano

	Model	Accuracy (%)
1	CNN (Kim, 2014)	87.2
2	-but-clause	87.3
3	-ℓ_2-reg	87.5
4	-project	87.9
5	-opt-project	88.3
6	-pipeline	87.9
7	-Rule-p	88.8
8	-Rule-q	**89.3**

Table 2: Performance of different rule integration methods on SST2. 1) CNN is the base network; 2) "-but-clause" takes the clause after "but" as input; 3) "-ℓ_2-reg" imposes a regularization term $\gamma\|\boldsymbol{\sigma}_\theta(S) - \boldsymbol{\sigma}_\theta(Y)\|_2$ to the CNN objective, with the strength γ selected on dev set; 4) "-project" projects the trained base CNN to the rule-regularized subspace with Eq.(3); 5) "-opt-project" directly optimizes the projected CNN; 6) "-pipeline" distills the pre-trained "-opt-project" to a plain CNN; 7-8) "-Rule-p" and "-Rule-q" are our models with p being the distilled student network and q the teacher network. Note that "-but-clause" and "-ℓ_2-reg" are ad-hoc methods applicable specifically to the "but"-rule.

possible integration approaches. Table 2 lists these methods and their performance on the SST2 task. We see that: 1) Although all methods lead to different degrees of improvement, our framework outperforms all other competitors with a large margin. 2) In particular, compared to the pipelined method in Row 6 which is in analogous to the structure compilation work (Liang et al., 2008), our iterative distillation (section 3.2) provides better performance. Another advantage of our method is that we only train one set of neural parameters, as opposed to two separate sets as in the pipelined approach. 3) The distilled student network "-Rule-p" achieves much superior accuracy compared to the base CNN, as well as "-project" and "-opt-project" which explicitly project CNN to the rule-constrained subspace. This validates that our distillation procedure transfers the structured knowledge into the neural parameters effectively. The inferior accuracy of "-opt-project" can be partially attributed to the poor performance of its neural network part which achieves only 85.1% accuracy and leads to inaccurate evaluation of the "but"-rule in Eq.(5).

We next explore the performance of our framework with varying numbers of labeled instances as well as the effect of exploiting unlabeled data. Intuitively, with less labeled examples we expect the general rules would contribute more to the performance, and unlabeled data should help better learn from the rules. This can be a useful property especially when data are sparse and labels are expensive to obtain. Table 3 shows the results. The subsampling is conducted on the sentence level. That is, for instance, in "5%" we first selected 5% training sentences uniformly at random, then trained the models on these sentences as well as their phrases. The results verify our expectations. 1) Rows 1-3 give the accuracy of using only data-label subsets for training. In every setting our methods consistently outperform the base CNN. 2) "-Rule-q" provides higher improvement on 5% data (with margin 2.6%) than on larger data (e.g., 2.3% on 10% data, and 2.0% on 30% data), showing promising potential in the sparse data context. 3) By adding unlabeled instances for semi-supervised learning as in Rows 5-6, we get further improved accuracy. 4) Row 4, "-semi-PR" is the posterior regularization (Ganchev et al., 2010) which imposes the rule constraint through only unlabeled data during training. Our distillation framework consistently provides substantially better results.

	Data size	5%	10%	30%	100%
1	CNN	79.9	81.6	83.6	87.2
2	-Rule-p	81.5	83.2	84.5	88.8
3	-Rule-q	82.5	83.9	85.6	**89.3**
4	-semi-PR	81.5	83.1	84.6	–
5	-semi-Rule-p	81.7	83.3	84.7	–
6	-semi-Rule-q	**82.7**	**84.2**	**85.7**	–

Table 3: Accuracy (%) on SST2 with varying sizes of labeled data and semi-supervised learning. The header row is the percentage of labeled examples for training. Rows 1-3 use only the supervised data. Rows 4-6 use semi-supervised learning where the remaining training data are used as unlabeled examples. For "-semi-PR" we only report its projected solution (in analogous to q) which performs better than the non-projected one (in analogous to p).

5.2 Named Entity Recognition

5.2.1 Setup

We evaluate on the well-established CoNLL-2003 NER benchmark (Tjong Kim Sang and De Meulder, 2003), which contains 14,987/3,466/3,684 sentences and 204,567/51,578/46,666 tokens in train/dev/test sets, respectively. The dataset includes 4 categories, i.e., *person, location, organization*, and *misc*. BIOES tagging scheme is used.

	Model	F1
1	BLSTM	89.55
2	BLSTM-Rule-trans	p: 89.80, q: 91.11
3	BLSTM-Rules	p: 89.93, q: **91.18**
4	NN-lex (Collobert et al., 2011)	89.59
5	S-LSTM (Lample et al., 2016)	90.33
6	BLSTM-lex (Chiu and Nichols, 2015)	90.77
7	BLSTM-CRF$_1$ (Lample et al., 2016)	90.94
8	Joint-NER-EL (Luo et al., 2015)	91.20
9	BLSTM-CRF$_2$ (Ma and Hovy, 2016)	**91.21**

Table 4: Performance of NER on CoNLL-2003. Row 2, BLSTM-Rule-trans imposes the transition rules (Eq.(6)) on the base BLSTM. Row 3, BLSTM-Rules further incorporates the list rule (Eq.(7)). We report the performance of both the student model p and the teacher model q.

Around 1.7% named entities occur in lists.

We use the mostly same configurations for the base BLSTM network as in (Chiu and Nichols, 2015), except that, besides the slight architecture difference (section 4.2), we apply Adadelta for parameter updating. GloVe (Pennington et al., 2014) word vectors are used to initialize word features.

5.2.2 Results

Table 4 presents the performance on the NER task. By incorporating the bi-gram transition rules (Row 2), the joint teacher model q achieves 1.56 improvement in F1 score that outperforms most previous neural based methods (Rows 4-7), including the BLSTM-CRF model (Lample et al., 2016) which applies a conditional random field (CRF) on top of a BLSTM in order to capture the transition patterns and encourage valid sequences. In contrast, our method implements the desired constraints in a more straightforward way by using the declarative logic rule language, and at the same time does not introduce extra model parameters to learn. Further integration of the list rule (Row 3) provides a second boost in performance, achieving an F1 score very close to the best-performing systems including Joint-NER-EL (Luo et al., 2015) (Row 8), a probabilistic graphical model optimizing NER and entity linking jointly with massive external resources, and BLSTM-CRF (Ma and Hovy, 2016), a combination of BLSTM and CRF with more parameters than our rule-enhanced neural networks.

From the table we see that the accuracy gap between the joint teacher model q and the distilled student p is relatively larger than in the sentiment classification task (Table 1). This is because in the NER task we have used logic rules that introduce extra dependencies between adjacent tag positions as well as multiple instances, making the explicit joint inference of q useful for fulfilling these structured constraints.

6 Discussion and Future Work

We have developed a framework which combines deep neural networks with first-order logic rules to allow integrating human knowledge and intentions into the neural models. In particular, we proposed an iterative distillation procedure that transfers the structured information of logic rules into the weights of neural networks. The transferring is done via a teacher network constructed using the posterior regularization principle. Our framework is general and applicable to various types of neural architectures. With a few intuitive rules, our framework significantly improves base networks on sentiment analysis and named entity recognition, demonstrating the practical significance of our approach.

Though we have focused on first-order logic rules, we leveraged soft logic formulation which can be easily extended to general probabilistic models for expressing structured distributions and performing inference and reasoning (Lake et al., 2015). We plan to explore these diverse knowledge representations to guide the DNN learning. The proposed iterative distillation procedure also reveals connections to recent neural autoencoders (Kingma and Welling, 2014; Rezende et al., 2014) where generative models encode probabilistic structures and neural recognition models distill the information through iterative optimization (Rezende et al., 2016; Johnson et al., 2016; Karaletsos et al., 2016).

The encouraging empirical results indicate a strong potential of our approach for improving other application domains such as vision tasks, which we plan to explore in the future. Finally, we also would like to generalize our framework to automatically learn the confidence of different rules, and derive new rules from data.

Acknowledgments

We thank the anonymous reviewers for their valuable comments. This work is supported by NSF IIS1218282, NSF IIS1447676, Air Force FA8721-05-C-0003, and FA8750-12-2-0342.

References

Stephen H Bach, Matthias Broecheler, Bert Huang, and Lise Getoor. 2015. Hinge-loss Markov random fields and probabilistic soft logic. *arXiv preprint arXiv:1505.04406*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *Proc. of ICLR*.

Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proc. of KDD*, pages 535–541. ACM.

Jason PC Chiu and Eric Nichols. 2015. Named entity recognition with bidirectional LSTM-CNNs. *arXiv preprint arXiv:1511.08308*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12:2493–2537.

James Foulds, Shachi Kumar, and Lise Getoor. 2015. Latent topic networks: A versatile probabilistic programming framework for topic models. In *Proc. of ICML*, pages 777–786.

Manoel VM França, Gerson Zaverucha, and Artur S dAvila Garcez. 2014. Fast relational learning using bottom clause propositionalization with artificial neural networks. *Machine learning*, 94(1):81–104.

Kuzman Ganchev, Joao Graça, Jennifer Gillenwater, and Ben Taskar. 2010. Posterior regularization for structured latent variable models. *JMLR*, 11:2001–2049.

Artur S d'Avila Garcez, Krysia Broda, and Dov M Gabbay. 2012. *Neural-symbolic learning systems: foundations and applications*. Springer Science & Business Media.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE*, 29(6):82–97.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of KDD*, pages 168–177. ACM.

Matthew J Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. 2016. Structured VAEs: Composing probabilistic graphical models and variational autoencoders. *arXiv preprint arXiv:1603.06277*.

Theofanis Karaletsos, Serge Belongie, Cornell Tech, and Gunnar Rätsch. 2016. Bayesian representation learning with oracle constraints. In *Proc. of ICLR*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *Proc. of EMNLP*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. In *Proc. of ICLR*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. of NIPS*, pages 1097–1105.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. 2015. Deep convolutional inverse graphics network. In *Proc. of NIPS*, pages 2530–2538.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL*.

Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *Proc. of ICML*.

Percy Liang, Hal Daumé III, and Dan Klein. 2008. Structure compilation: trading structure for features. In *Proc. of ICML*, pages 592–599. ACM.

Percy Liang, Michael I Jordan, and Dan Klein. 2009. Learning from measurements in exponential families. In *Proc. of ICML*, pages 641–648. ACM.

David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. 2016. Unifying distillation and privileged information. *Prof. of ICLR*.

Gang Luo, Xiaojiang Huang, Chin-Yew Lin, and Zaiqing Nie. 2015. Joint named entity recognition and disambiguation. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. of NIPS*, pages 3111–3119.

Marvin Minksy. 1980. Learning meaning. *Technical Report AI Lab Memo. Project MAC. MIT*.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proc. of CVPR*, pages 427–436. IEEE.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of ACL*, pages 115–124.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proc. of EMNLP*, volume 14, pages 1532–1543.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *Proc. of ICML*.

Danilo Jimenez Rezende, Shakir Mohamed, Ivo Danihelka, Karol Gregor, and Daan Wierstra. 2016. One-shot generalization in deep generative models. *arXiv preprint arXiv:1603.05106*.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proc. of EMNLP*, volume 1631, page 1642. Citeseer.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. *Proc. of ICLR*.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proc. of CoNLL*, pages 142–147. Association for Computational Linguistics.

Geoffrey G Towell, Jude W Shavlik, and Michiel O Noordewier. 1990. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the eighth National conference on Artificial intelligence*, pages 861–866. Boston, MA.

Sida Wang and Christopher Manning. 2013. Fast dropout training. In *Proc. of ICML*, pages 118–126.

Bishan Yang and Claire Cardie. 2014. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In *Proc. of ACL*, pages 325–335.

Wenpeng Yin and Hinrich Schutze. 2015. Multichannel variable-size convolution for sentence classification. *Proc. of CONLL*.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Ye Zhang, Stephen Roller, and Byron Wallace. 2016. MGNC-CNN: A simple approach to exploiting multiple word embeddings for sentence classification. *Proc. of NAACL*.

Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent SVMs. *JMLR*, 15(1):1799–1847.

Case and Cause in Icelandic: Reconstructing Causal Networks of Cascaded Language Changes

Fermín Moscoso del Prado Martín and **Christian Brendel**
University of California, Santa Barbara
Department of Linguistics, South Hall 3521
Santa Barbara, CA 93106, USA
fmoscoso@linguistics.ucsb.edu and cdbrendel@gmail.com

Abstract

Linguistic drift is a process that produces slow irreversible changes in the grammar and function of a language's constructions. Importantly, changes in a part of a language can have trickle down effects, triggering changes elsewhere in that language. Although such causally triggered chains of changes have long been hypothesized by historical linguists, no explicit demonstration of the actual causality has been provided. In this study, we use co-occurrence statistics and machine learning to demonstrate that the functions of morphological cases experience a slow, irreversible drift along history, even in a language as conservative as is Icelandic. Crucially, we then move on to demonstrate –using the notion of Granger-causality– that there are explicit *causal* connections between the changes in the functions of the different cases, which are consistent with documented processes in the history of Icelandic. Our technique provides a means for the quantitative reconstruction of connected networks of subtle linguistic changes.

1 Introduction

Sapir (1921/2014, p. 123) noticed that "Language moves down on a current of its own making. It has a *drift*" (emphasis added). In Sapir's view, the formation of different dialects requires that the small changes constantly being introduced by the speakers are not just plain white noise, but rather random walks in which minute changes accumulate over time. The very high dimensionality on which languages operate makes cumulative linguistic changes irreversible. Once a change has been effected there is very little chance that the language will ever return to its original state before the change, in the same way that a diffusion process in a very high dimensional space is never going to return to the exact same point in the space. Drift in language is in this respect reminiscent of random genetic drift from evolutionary biology (Wright, 1955). However, Sapir's idea of drift goes further in that he viewed it as a *directional* process, more similar to Wright's (1929) concept of a directional drift related to selectional pressures. In Sapir's view, "language has a 'slope"'; the small changes that accumulate in linguistic drift are not fully random, but rather they reflect the speakers' unconscious cognitive tendency to increase the consistency within their languages. This idea is currently challenged by some researchers (Croft, 2000; Lupyan and Dale, 2015), who are of the opinion that purely random drift –of the same type as that found in genetics–, when coupled with adequate selection mechanisms, is sufficient to account for the diachronic changes observed in the world's languages. Sapir motivated the need for directional change in what he saw as apparent causal chains in language change, which he illustrated with the progressive loss and functional shift of English oblique case markers, into an absolutive case-free system encoding animacy and position relative to the head noun.

'Chain reactions' along the history of a language are particularly well-studied in phonology. Chain shifts (Martinet, 1952) are processes by which the position in perceptual/articulatory space of a phoneme changes in response to the change in position of another phoneme (either moving away from the second phoneme, in a 'push' chain, or moving to occupy the space left void by the other, in a 'pull' chain). A famous example of a chain shift is the Great English Vowel Shift. In a similar fashion, one could think of functional chain shifts

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2421–2430,
Berlin, Germany, August 7-12, 2016.

in morphology, by which a certain morphological category takes over some of the functions of another, triggering a chain of 'push' and/or 'pull' movements in other categories. Such cascaded changes have often been reported in diachronic linguistics (Biberauer and Roberts, 2008; Fisiak, 1984; Lightfoot, 2002; Wittmann, 1983).

Icelandic is a famously conservative language. Compared to most other languages, its grammar has experienced remarkably little change since the high middle ages. For instance, Barðdal and Eythórsson (2003) argue that the changes it has experienced from its old phase (Old West Norse; mid XI century to mid XIV century) to its current phase are comparable to the slight changes occurring from early Modern English (late XV century into early XVIII century) into Modern English (from early XVIII century). In terms of inflectional morphology, change in Icelandic has been minimal. For instance, one finds that the nominal paradigms of Old West Norse, are mostly the same as those of Modern Icelandic. Notwithstanding the apparent formal stability of Icelandic cases, there is evidence that they are experiencing subtle changes in their *functions* (Barðdal, 2011; Barðdal and Eythórsson, 2003; Eythórsson, 2000). In particular, Barðdal argues that an accumulation of small syntatico-semantic shifts has finally resulted in a shift in the Icelandic dative's functions (i.e., 'dative sickness'), possibly triggered by earlier changes in nominatives and accusatives (e.g., 'nominative sickness').

In this study, we investigate whether one can reliably detect a drift in the functions of Icelandic case and –crucially– whether there is evidence for causal chain shifts in these functions. In Section 2, we describe the processing of a diachronic corpus of Icelandic to obtain co-occurrence representations of the functions of case types and tokens. Section 3 uses machine learning on the co-occurrence vectors of tokens to demonstrate that the usage of Icelandic cases has been subject to a constant drift along history, a drift that is distinguishable from the overall changes experienced by the language in this period. We then go on –in Section 4– to demonstrate using Granger-causality (Granger, 1969) that there are causal relations between the changes in the different cases, and it is possible to reconstruct a directed network of chain shifts, which is consistent with the directions of causality hypothesized by Barðdal (2011). Finally, in Section 5, we discuss the theoretical implications of our results for theories of language change, as well as the possibilities offered by the technical innovations presented here.

2 Corpus Processing

2.1 Corpus

We used the Icelandic Parsed Historical Corpus (Wallenberg et al., 2011), a sample of around one million word tokens of Icelandic texts that have been orthographically standardized, manually lemmatized, part-of-speech tagged, and parsed into context-free derivation trees. An example of the lemmatization and part-of-speech tagging for a sentence is shown in Fig. 1. The dating of the text samples ranges from 1,150 CE to 2,008 CE, covering most of the history of Icelandic (from its origins in Old West Norse, to the current official language of Iceland). The corpus is divided into 61 files of similar sizes (around 18,000 words per file), each file corresponding to a single document. The documents were chosen to cover the period in a roughly uniform manner, sampling from similar genres across the periods.

2.2 Preprocessing

We collapsed into a single file all documents that were dated on the same year. This left us with 44 files containing texts from distinct years. From each of the files, we discarded all tokens that contained anything but valid Icelandic alphabetic characters or the dollar sign (used for marking enclitic breaks within a word, such as the clitic determiner in *krossins* from the example in Fig. 1). All remaining word tokens were lower-cased, and the '$' character was removed from the stem elements of broken stem plus clitic pairs (e.g., *kross$* was changed into *kross*).

2.3 First Order Co-occurrence Vectors

Ideally, for constructing co-occurrence vectors, it is best to choose as features those words with highest overall informativity, which in fact tend to be those words with the highest occurrence frequencies (Bullinaria and Levy, 2007; Bullinaria and Levy, 2012; Lowe and McDonald, 2000). In our case, however, using plain token frequencies runs the risk of creating a representational space that is strongly uninformative about particular periods in the history of the language. Instead, we used document frequencies, as these still provide a measure

En	*armar*	*kross-*	*-ins*	*merkja*	*ást*	*við*	*Guð*	*og*	*menn*
en	*armur*	*krosur*	*hinn*	*merkja*	*ást*	*við*	*guð*	*og*	*maður*
CONJ	NS-N	N-G	D-G	VBP	N-A	P	NPR-A	CONJ	NS-A

Figure 1: Tagging and lemmatization of the Old West Norse sentence (number 819) from the *Íslensk Hómilíubók* ("Icelandic Book of Homilies"; late XII century): *En armar krossins merkja ást við Guð og menn.* ("But the arms of the cross mark the love of God and man.")

of word frequency (and therefore informativity), while at the same time ensuring that those words chosen as features are most representative across the history of the language. We selected as features all word types that occurred in at least 75% of the 44 by-year files, that is, all 529 distinct (unlemmatized) word forms that had a document frequency in the corpus of at least 33 documents.

For each (unlemmatized) word type (w) occurring at least three times in the whole corpus (17,741 distinct word types), we computed its co-occurrence frequency with each of the feature words (t). In this way, we obtained a matrix of 17,741 $\times$ 529 word by feature co-occurrence frequencies (f$[w, t]$) within a symmetrical window including the two preceding and following words.[1] The plain co-occurrence matrix was converted into a matrix of word-feature pointwise mutual informations $M = (m_{i,j})$, such that,

$$m_{i,j} = \log \frac{N \cdot \mathrm{f}[w_i, t_j]}{(W_1 - 1) \cdot \mathrm{f}[w_i] \cdot \mathrm{f}[t_j]},$$

where N = 899,763 tokens is the total number of tokens in the corpus, $W_1 = 5$ is the total sliding window size considered, and f$[w_i]$ and f$[t_j]$, are the overall corpus frequencies of words w_i and t_j, respectively. In this manner, the row $M_{i,\cdot} = (m_{i,1}, \ldots, m_{i,529})$ represents the contexts in which the word type w_i is found across the whole corpus.

2.4 Second Order Co-occurrence Vectors

The co-occurrence vectors computed above provide representations for the average contexts in which a given word type is found. In order to represent the specific context of each word token, we used second order co-occurrence vectors (Schütze and Pedersen, 1997). These provide important information about the aspects of a context that are relevant for inflectional morphology (Moscoso del Prado Martín, 2007). The second order vectors were computed by passing a symmetrical sliding window including, for each token, the immediately preceding and following word. The vector for each token was computed as the average between the first order vectors (of Subsection 2.3) of the preceding and following words. If no first order vector was available for either the preceding or the following word, the second order vector directly corresponded to the plain first order vector of the word for which there was a first order vector. We excluded those tokens for which we had first order vectors for neither the previous nor the following word type. We computed such second order vectors for all tokens in the corpus that had been tagged for grammatical case (a total of 419,910 vectors, on average 9,453 vectors per year, of which 38.14% were nominatives, 10.91% were genitives, 26.38% were accusatives, and 24.56% were datives).

2.5 Representation of the Case Prototypes

In order to represent the prototypical usages each grammatical case (i.e., nominative, genitive, accusative, and dative) in a given year, we used the first order co-occurrence technique. For each of the 44 distinct years –using the same features identified in Subsection 2.3– we computed first order co-occurrence vectors collapsing all word tokens in each grammatical case, and using a reduced window size including just the preceding and following words (i.e., $W_2 = 3$).[2] For each year (y) we obtained a 4×529 element matrix of co-occurrence frequencies (f$_y[c, t]$), indicating the number of times that each case (c)

[1] To avoid $\log 0$ values, all frequency counts in this paper were increased by one.

[2] The optimal window sizes for the first order co-occurrence vectors for words and for case prototypes were different because they were chosen to optimize different tasks. The window size for first order vectors for words were chosen to optimize the machine learning algorithm for identifying case identities of second order vectors (Section 3), whereas the first order vectors for case prototypes were optimized for clustering cases across the years (Section 4).

was found to co-occur (within the specified window) with feature (t). These matrices were transformed into case to feature pointwise mutual informations, resulting in a series of 44 matrices ($M[y] = (m[y]_{i,j})$ such that,

$$m[y]_{i,j} = \log \frac{N \cdot \mathrm{f}_y[c_i, t_j]}{(W_2 - 1) \cdot \mathrm{f}[c_i] \cdot \mathrm{f}[t_j]},$$

where N is the total number of tokens in the corpus, $\mathrm{f}[c_i]$ is the number of instances of case c_i, and $\mathrm{f}[t_j]$ denotes the number of instances of word t_j in the corpus. In this way, the rows of the $M[y]$ matrices provided a representation of the contexts in which each grammatical case was used in each year.

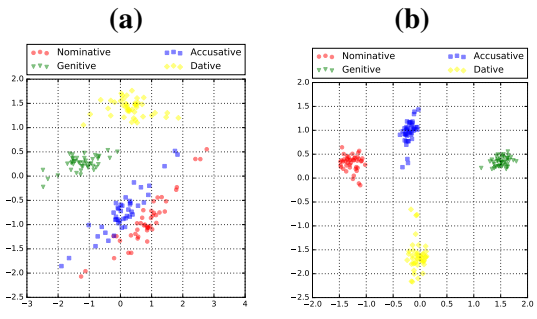


Figure 2: **(a)** Representation of the raw case vectors in SVD-reduced space (i.e., SVD dimension 1 vs. dimension 2). **(b)** Representation of the case vectors after discounting the average vector for each year (i.e., SVD dimension 1 vs. dimension 2).

Fig. 2a plots the spatial organization of the resulting vectors (after reducing to a bidimensional projection using Singular Value Decomposition; SVD). Notice that the prototypes for each case very naturally cluster together across the years. The scatter is however asymmetric, hinting at a process of change along the years common for all four cases. If we compute a yearly overall prototype vector as the average vector for the cases in each year, and we substract it from the corresponding case prototypes, we find that the case identities become clearly differentiated in space (see Fig. 2b), demonstrating that the case prototype vectors do indeed capture the contextual properties of all four cases, which are highly distinctive.

3 Functional Drift in Icelandic Cases

As was discussed in the Introduction, the inflectional paradigms marking case and number have barely –if at all– changed along the history of Icelandic. On the basis of this fact alone, one could conclude that the grammatical case system is not actually experiencing any linguistic drift, but has rather remained basically static throughout the last millennium. There is, however, another possibility. Linguistic drift could have been affecting the *functions* of grammatical cases in Icelandic. If this were the case, one would expect to observe a slow –constant rate– diachronic change in the contexts in which each of the four cases is used.

To investigate this latter possibility, for each of the 44 years documented in the corpus, we trained a basic logistic classifier in the task of assigning grammatical case to the second order co-occurrence vectors developed in Subsection 2.4. Once each of the classifiers had been trained, we tested the classifiers' performances on the vectors obtained from each of other 43 years on which they were not trained. On the one hand, if the functions of the cases have indeed remained constant along the history of the language, one would expect that the performance of a classifier tested on the data from a given year, should remain approximately constant when tested on vectors from all other years. If, on the other hand, the functions of Icelandic grammatical cases have been subject to linguistic drift, the irreversible and cumulative nature of the drift (Sapir, 1921/2014) implies that the classifier error should grow –if only so slightly– with each year passed. The reason for this is that the contexts in which one would use each case should be slightly different from year to year. One should then predict that the error of the classifier should depend on the temporal distance between the year of the testing vectors and that of the training ones. Furthermore, the change in error should be of a linear nature, with a very slight slope.

When tested on the same years in which they had been trained, the classifiers performed rather well in inferring the case to which each of the context vectors belonged (the distribution of errors across the 44 years was well approximated by a normal distribution with a mean error of 26.67%, a standard deviation of 1.99%, and best and worst classification errors of 22.17% and 30.39%, respectively).[3]

[3]Although we chose the best model among different learning algorithms, including multiple versions of Support Vector Machines, Classification Trees, and a Softmax Classifier, we have no doubt that the learning performance can be improved upon. For our purposes, however, it was sufficient

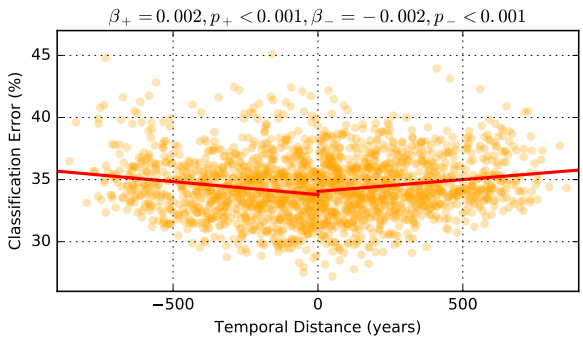


Figure 3: Correlation between the classifier error and the temporal distance from the year from which the training vectors were obtained to the year when the testing vectors were obtained. The solid lines plots a linear regressions.

We then tested the classifiers on the vectors obtained from different years. The results are plotted in Fig. 3. The scatter plots the difference in years (i.e., the difference values are positive when the classifier was tested on vectors obtained after those used for training, and negative when testing with vectors obtained before the training ones). When testing on data different from the training sets, there is a logical loss in performance (of about 8%) from the baseline of testing on the same training set. We fitted two linear regressions, one to the positive differences and another to the negative differences (plotted by the solid lines in Fig. 3). The first thing that stands out is that the performance of the classifier is remarkably good when tested on vectors obtained at considerable temporal distances from the time when the training vectors were obtained. While the error of the classifier is of about 34% when tested on vectors from the year after or before the training vectors, the error remains at 35% for vectors originating from texts that are five centuries apart. Once again, this speaks to the remarkable conservativeness of the Icelandic language. However, these small differences are in fact reliable: There are significant slopes in both regressions (positive differences: $R_+ = .161, p_+ < .001$; negative differences: $R_- = -.164, p_- < .001$). A second remarkable thing is that both regressions are substantially symmetrical, in fact their slopes are basically identical ($|\beta_+| = |\beta_-| = .002$). This indicates that the degree to which the usages of the cases at different

to have a classifier with a decent performance, as our goal was showing that the error is time-dependent.

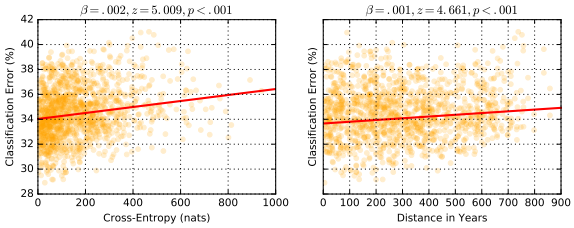


Figure 4: Independent effects of cross-entropy (left panel) and distance in years between the training and testing sets (right panel) as estimated by the linear mixed-effects regression model.

time points have diverged depends on the amount of time that has intervened, irrespective of whether it was the training or the testing set that was collected before.

One could argue that the slow drift observed may not be really due to changes in the functions of the grammatical cases themselves, but just to overall changes either in the overall language, or in the very topics that are addressed (e.g., one might guess that talk of swords, slaves, and longships was more frequent in XII century Norse than it is in Modern Icelandic). To investigate this possibility we used an information-theoretical measure of the prototypicality of a set of second order vectors for a particular year (based on that used in Moscoso del Prado Martín, 2007). From the vectors of each year, irrespective of their case, we fitted a 529-dimensional Gaussian distribution (by estimating the mean vector for that year, $\boldsymbol{\mu}_y$, and the covariance matrix, Σ_y). The average inadequacy of a given set of vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ obtained in year z to the distribution fitted to the vectors obtained in year y is measured by the *cross-entropy*, a Monte Carlo estimator of which is given by,[4]

$$H(z, y) \approx K + \frac{1}{2}\log|\Sigma_y| + \frac{1}{2n}\sum_{i=1}^{n}(\mathbf{v}_i - \boldsymbol{\mu}_y)^T\Sigma_y^{-1}(\mathbf{v}_i - \boldsymbol{\mu}_y),$$

where K is a constant.[5] In addition, one should also take into account the fact that, for some years, the classifier might generalize better or worse than for others (due to irrelevant idiosyncrasies of one specific text used for training), which could lead to a distortion of the results.

To investigate whether, after discounting for the inadequacy of the vectors to the overall distribution of those in which the classifier was trained,

[4] Assuming Σ_y is definite positive so that its inverse exists.
[5] $K = \frac{529}{2}\log 2\pi$.

there was still evidence for drift in the functions of the cases, while also accounting for the different generalization powers of the classifiers, we fitted a linear mixed-effects model to the classifier errors, including fixed-effect predictors of the cross-entropy described above, and the absolute value of the difference in years between the training set and testing set dates (as indicated above, the effects were equivalent for positive and negative values in years), and the dating of the testing set as a random effect. As expected, we found that the cross-entropies had a significant positive effect ($\beta = .002, z = 5.009, p < .001$; left panel if Fig. 4), indicating that the performances of the models were indeed worse for less adequate sets of testing vectors, irrespective of any aspect of grammatical case. However –crucially– even after considering the effect of cross-entropy, there remained a significant positive effect of the temporal distance ($\beta = .001, z = 4.661, p < .001$; right panel if Fig. 4).[6] This result therefore supports the hypothesis that the function of grammatical case has been subject to a slight constant change during the history of the language: a functional drift.

4 Functional Chain Shifts in Case

In the previous section we have demonstrated that the functions of Icelandic cases have been subject to slow linguistic drift. The question now arises of whether this drift is purely random, or rather it has some degree of directionality arising from endogenous linguistic factors. It is possible that changes in the functions of some cases caused changes in the functions of others. We investigate this possibility using the notion of Granger-causality

4.1 Granger-causality

Granger-causality (Granger, 1969) is a powerful technique for assessing whether one time series can be said to be the cause of another one. The basic idea is that one time series $\mathbf{x}$ is said to *Granger-cause* another series $\mathbf{y}$ if the past of series $\mathbf{x}$ predicts the future of series $\mathbf{y}$ over and above any predictive power that can be found on $\mathbf{y}$'s own past. This idea has proven of great value to investigate the causal connections between economic variables, sequences of behavioral responses, neural spikes, or electroencephalographic potentials. Often, the technique is used to reconstruct directional networks of variables and processes that have causal connections.

If $\mathbf{x}$ and $\mathbf{y}$ are stationary time sequences on time (τ), in order to test whether $\mathbf{x}$ Granger-causes $\mathbf{y}$, one begins by fitting *autoregressive models* (AR) that predict the values of $\mathbf{y}$ from its own n values lagged into the past. This consists on finding values $a_1, a_2, \ldots, a_n$ that minimize the error ε in the equation,

$$y[\tau] = a_0 + \overbrace{a_1 y[\tau-1] + a_2 y[\tau-2] + \ldots + a_n y[\tau-n]}^{\text{past of } \mathbf{y}} + \varepsilon[\tau].$$

One then augments the autoregression by including m lagged values of $\mathbf{x}$, with additional parameters $b_1, \ldots, b_m$ to be fitted,

$$\begin{aligned} y[\tau] = & a_0 + \overbrace{a_1 y[\tau-1] + a_2 y[\tau-2] + \ldots + a_n y[\tau-n]}^{\text{past of } \mathbf{y}} + \\ & + \underbrace{b_1 x[\tau-1] + b_2 x[\tau-2] + \ldots + b_m x[\tau-m]}_{\text{past of } \mathbf{x}} + \varepsilon[\tau]. \end{aligned}$$

where the ε sequences are uncorrelated (white) gaussian noises, reflecting the fully random or chaotic part of the system, which cannot be predicted from its past (i.e., the error, that is termed by some the 'creativity' of the model). If the second regression is a significant improvement over the first, then it can be said that $\mathbf{x}$ *Granger-causes* $\mathbf{y}$, indicating that past values of $\mathbf{x}$ significantly predict future values of $\mathbf{y}$ over and above any predictive power of $\mathbf{y}$'s own past values. This is tested using an F-test, with the null hypothesis being that the second model does not improve on the first one. The selection of the autoregressive order parameters n and m is achieved by model comparisons using information criteria.

When one is interested in reconstructing a network of causal relations between multiple variables, one can use a mutivariate generalization of the AR model, the *vector autoregressive model* (VAR). The VAR model consists of mutiple AR equations (one for each variable in the model). If we consider an autoregressive order of one (i.e., $m = n = 1$), when we are simultaneously considering p variables $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_p\}$, the VAR[1] model to be fitted can be expressed in matrix no-

[6] The estimated covariance matrices were not definite positive for two of the years, which were excluded from the analyses. In addition, in 552 out of the remaining 1,849 estimates, the cross-entropy took unusually large values, orders of magnitude larger than the rest (likely reflecting inadequacy of the multidimensional Gaussian approximation for these cases), which distorted the effect estimates. The analyses reported exclude these 552 points. However, keeping these outlying values in the regression, both key effects remained significant, but the slope estimates were less trustworthy.

tation as,

$$\begin{pmatrix} y_1[\tau] \\ \vdots \\ y_p[\tau] \end{pmatrix} = \begin{pmatrix} a_1 \\ \vdots \\ a_p \end{pmatrix} + \begin{pmatrix} A_{1,1} & \dots & A_{1,p} \\ \vdots & \ddots & \vdots \\ A_{p,1} & \dots & A_{p,p} \end{pmatrix} \begin{pmatrix} y_1[\tau-1] \\ \vdots \\ y_p[\tau-1] \end{pmatrix} + \begin{pmatrix} \varepsilon_1[\tau] \\ \vdots \\ \varepsilon_p[\tau] \end{pmatrix}.$$

This model enables testing for Granger-causality between any pair of variables $\mathbf{y}_i \in \mathcal{Y}$ and $\mathbf{y}_j \in \mathcal{Y}$, after partialling out the possible confounding effects of $\{\mathbf{y}_t,\, t \neq i, t \neq j, 1 \leq t \leq p\}$. $\mathbf{y}_j$ is said to Granger-cause $\mathbf{y}_i$ if the model coefficient $A_{i,j}$ is significantly different than zero, and the reverse holds if $A_{j,i}$ significantly different than zero (i.e., $\mathbf{y}_i$ Granger-causes $\mathbf{y}_j$).

4.2 Granger-causality in Case Drift

To investigate whether the pattern of change in one case triggers (i.e., Granger-cause) the pattern of change in another, we made use of the prototype vectors for the cases in each of the years developed in Subsection 2.5. As a measure of the amount of contextual change for a given case in a given year, we computed the city-block distances between the case prototypes from each year to the next available time point, which are plotted in Fig. 5a. Notice that there is an overall pattern of change equally affecting all cases, and the changes are therefore strongly correlated. This reflects the overall pattern of historical changes affecting Icelandic as a whole, as well as changes in the topics that would be discussed in the different time periods, as was documented in Subsection 2.5 and Section 3. Considering the changes in each case as a component in a four-dimensional vector, the modulus of this vector (plotted by the dashed orange line in Fig. 5a) gives the overall magnitude of the changes that are unspecific to the cases themselves. To remove this component from the changes, we fitted a linear regression to the sequence of changes in each case, using the overall pattern of change as a predictor. Fig. 5b plots the resulting residuals, indicating the amount of change that was specific to each case, over and above the overall pattern.[7]

A precondition for testing for Granger-causality is that the time series under consideration are stationary. In our case, the series depicted in Fig. 5b are significantly non-stationary; they exhibit, for instance, significant temporal trends. In order to remove the non-stationarities, the series were differentiated (i.e., we considered the difference between each two consecutive points). The result of this differentiation, plotted in Fig. 5c, removed the non-stationary trends from the original series.

Table 1: Results of the Granger-causality analyses. Causality directions that remained significant after FDR correction are highlighted in bold.

Direction	$F[1,144]$	p	p (FDR)	Direction	$F[1,144]$	p	p (FDR)
Nom. → Gen.	2.614	.108	.184	**Gen. → Nom.**	5.618	.019	**.046**
Nom. → Dat.	8.295	.005	**.018**	Dat. → Nom.	3.834	.052	.104
Gen. → Acc.	.566	.453	.454	Acc. → Gen.	2.408	.123	.184
Acc. → Nom.	6.802	.010	**.030**	Nom. → Acc.	.644	.424	.454
Acc. → Dat.	10.249	.002	**.018**	Dat. → Acc.	.563	.454	.454
Dat. → Gen.	1.354	.246	.329	**Gen. → Dat.**	9.034	.003	**.018**

We fitted a VAR[n] model to the four differentiated time-series. The autoregressive order found to maximize Akaike's Information Criterion (Akaike, 1974) was $n = 1$.[8] The F statistics and significance values for the coefficients in the resulting VAR[1] model are given in Tab. 1. In order to reconstruct the causality network, we also need to consider that we started out with only very vague predictions on the possible directions of causality. As the model involved twelve separate p-value tests, the p-value estimates need to be corrected for multiple comparisons. This correction was done using the false discovery rate (FDR) method (Benjamini and Hochberg, 1995), resulting in the corrected p-value estimates listed in the last column of Tab. 1.

The Granger-causality analysis leads us to reconstruct the causality network depicted in Fig. 6. It appears that the drift in the functions of Icelandic case is not plainly random. Instead, we find evidence that changes in the functions of the accusatives and genitives have had a domino effect, triggering further changes in the functions of nominatives. Finally, changes in all other three cases result in changes in the functions of the dative. In summary, the changes observed are consistent with the idea discussed in the Introduction of a *functional chain shift* affecting the morphological case system of Icelandic.

5 Discussion

We have presented evidence for a steady drift –of the precise kind advocated by Sapir (1921/2014)– even in a language as remarkably conservative as is Icelandic. This supports the claim that human languages are in a state of 'perpetual motion' (Beckner et al., 2009; Dediu et al., 2013; Hawkins and Gell-Mann, 1992; Hopper, 1987;

[7] Negative values in this figure indicate changing less than the average, rather than 'negative change'.

[8] In fact $n = 1$ was also found to maximize both Akaike's Final Prediction Error and Hannan-Quinn Criteria.

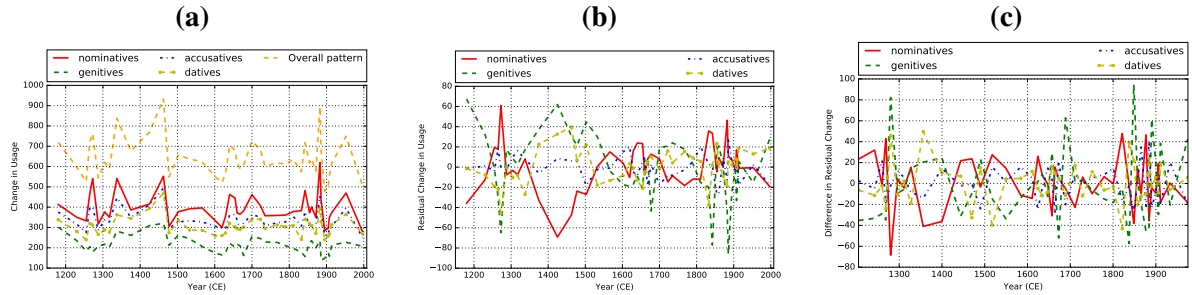


Figure 5: **(a)** Overall value of the city-block distances between the prototypical case vectors for consecutive years. **(b)** Residual value of the distances specific to each case after residualizing the overall pattern of change. **(c)** Differentiated values of the residualized distances, removing non-stationarities.

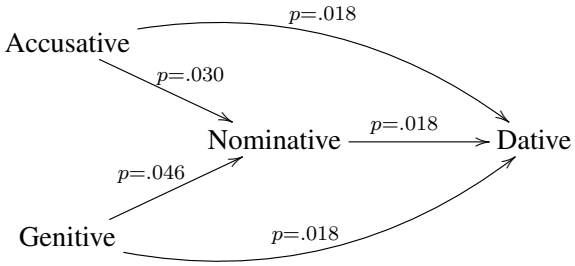


Figure 6: Reconstructed network of Granger-causal connections between diachronic changes in case functions. The p-values indicated on the causal arrows are FDR-corrected.

Larsen-Freeman and Cameron, 2007; Niyogi and Berwick, 1997). Although we have found that functional change in Icelandic case has proceeded at a constant rate, we do not think, as argued by Nettle (1999a, 1999b), that this rate of change needs to be constant across languages. There are strong arguments suggesting that in other languages such rates might be different (Wichmann, 2008; Wichmann and Holman, 2009).

The crucial innovation presented in this paper is the reconstruction of the causality network linking the changes in the four cases. Previous applications of the notion of Granger-causality to diachronic language change (Moscoso del Prado Martín, 2014) have focused on the macroscopic relation between sudden changes in syntax and morphology. Here, we have demonstrated that Granger-causality can also be used to reconstruct detailed networks of slow changes within the morphological system, at a more microscopic scale. The techniques developed offer a mechanism for investigating subtle changes in the functions of linguistic constructions, and the causal relations between them. Traditionally, historical linguists have focused on 'narrative' accounts of the the chains of change within a language. Although such type of accounts are extremely useful, the often very subtle changes in usage that can occur from one time-point to another cannot always be described with such clearcut patterns. Nevertheless, we have shown that those very small changes do accumulate in meaningful ways.

An important question addressed by this study is the presence of endogenous causal chains in language change. Lupyan and Dale (2015) argue that languages are constrained by their 'ecological niches', the communities in which they are spoken, and the extralinguistic properties of those niches can trigger exogenous change in the morphology of the languages. Following on Lupyan and Dale's ecosystem analogy, one should see that, as well as being part of ecosystems, languages are also ecosystems in themselves, in a nesting similar to that found in natural ecosystems (i.e., an animal is part of a particular ecosystem, and its body is an ecosystem in itself). Sounds, words and constructions have their own ecological niches within the language, and disturbances in the system can trigger cascaded changes, leading to readaptation (evolution) of the constructions. This contrasts with the view of changes in the function of Icelandic cases expressed by Eythórsson (2000). He showed that verbs whose arguments exhibit 'nominative sickness' and 'accusative sickness' tend to be clustered along certain syntactic and semantic lines. That it is in these particular niches that accusatives and datives ended up settling is not, however, the *cause* of the language changes. As we have shown, the case system was subject to a string of cascaded pressures. That the cases ended

up settling in new syntactico-semantic niches was the *result*, rather than the cause of the changes.

References

Hirotogu Akaike. 1974. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19:716–723.

Jóhanna Barðdal and Thórhallur Eythórsson. 2003. The change that never happened: the story of oblique subjects. *Journal of Linguistics*, 39:439–472.

Jóhanna Barðdal. 2011. The rise of dative substitution in the history of Icelandic. *Lingua*, 121:60–79.

Clay Beckner, Nick C. Ellis, Richard Blythe, John Holland, Joan Bybee, Jinyun Ke, Morten H. Christensen, Diane Larsen-Freeman, William Croft, and Tom Schoenemann. 2009. Language is a complex adaptive system: Position paper. *Language Learning*, 59:1–26.

Yoav Benjamini and Yosef Hochberg. 1995. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57:289–300.

Theresa Biberauer and Ian Roberts. 2008. Cascading parameter changes: internally driven change in Middle and Early Modern English. In Thórhallur Eythórsson, editor, *Grammatical Change and Linguistic Theory: The Rosendal Papers*, pages 79–114. John Benjamins, Philadelphia, PA.

John A. Bullinaria and Joseph P. Levy. 2007. Extracting semantic representations from word co-occurrence statistics: A computational study. *Behavior Research Methods*, 39:510–526.

John A. Bullinaria and Joseph P. Levy. 2012. Extracting semantic representations from word co-occurrence statistics: Stop-lists, stemming and SVD. *Behavior Research Methods*, 44:890–907.

William Croft. 2000. *Explaining Language Change: An Evolutionary Approach*. Longman, London, England.

Dan Dediu, Michael Cysouw, Stephen C. Levinson, Andrea Baronchelli, Morten H. Christensen, William Croft, Nicholas Evans, Simon Garrod, Rusell D. Gray, Anne Kandler, and Elena Lieven. 2013. Cultural evolution of language. In Peter J. Richerson and Morten H. Christensen, editors, *Cultural Evolution: Society, Technology, Language, and Religion*, pages 303–331. MIT Press, Cambridge, MA.

Thórhallur Eythórsson. 2000. Dative vs. nominative: changes in quirky subjects in Icelandic. *Leeds Working Papers in Linguistics*, 8:27–44.

Adam Fisiak, editor. 1984. *Historical Syntax*. de Gruyter, Berlin.

Clive W. J. Granger. 1969. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37:424–438.

John Hawkins and Murray Gell-Mann, editors. 1992. *The Evolution of Human Languages*. Santa Fe Institute Studies in the Sciences of Complexity. Addison Wesley, Reading, MA.

Paul J. Hopper. 1987. Emergent grammar. *Proceedings of the Berkeley Linguistic Society*, 13:139–157.

Diane Larsen-Freeman and Lynne Cameron. 2007. *Complex Systems and Applied Linguistics*. Oxford University Press, Oxford, UK.

David Lightfoot, editor. 2002. *Syntactic Effects of Morphological Change.* Oxford University Press, Oxford, UK.

Will Lowe and Scott McDonald. 2000. The direct route: Mediated priming in semantic space. In Lila Gleitman and Aravind K. Joshi, editors, *Proceedings of the XXII Annual Conference of the Cognitive Science Society*, pages 806–811, Austin, TX. Cognitive Science Society.

Gary Lupyan and Rick Dale. 2015. The role of adaptation in understanding linguistic diversity. In Rik De Busser and Randy J. LaPolla, editors, *Language structure and environment: Social, cultural, and natural factors*, pages 289–316. John Benjamins Publishing Company, Philadelphia, PA.

André Martinet. 1952. Function, structure, and sound change. *Word*, 8:1–32.

Fermín Moscoso del Prado Martín. 2007. Co-occurrence and the effect of inflectional paradigms. *Lingue e Linguaggio*, 6:247–263.

Fermín Moscoso del Prado Martín. 2014. Grammatical change begins within the word: Causal modeling of the co-evolution of Icelandic morphology and syntax. In Paul Bello, Marcello Guarini, Marjorie McShane, and Brian Scassellati, editors, *Proceedings of the XXXVII Annual Conference of the Cognitive Science Society*, pages 2657–2662, Austin, TX. Cognitive Science Society.

Daniel Nettle. 1999a. Using social impact theory to simulate language change. *Lingua*, 108:95–117.

Daniel Nettle. 1999b. Is the rate of linguistic change constant? *Lingua*, 108:119–136.

Partha Niyogi and Robert C. Berwick. 1997. A dynamical systems model for language change. *Complex Systems*, 11:161–204.

Edward Sapir. 2014. *Language: An Introduction to the Study of Speech.* Dover Publications, Mineola, NY. (Original work published 1921).

Hinrich Schütze and Jan O. Pedersen. 1997. A cooccurrence-based thesaurus and two applications to information retrieval. *Information Processing & Management*, 33:307–318.

Joel C. Wallenberg, Anton Karl Ingason, Einar Freyr Sigurðsson, and Eiríkur Rögnvaldsson. 2011. Icelandic parsed historical corpus (IcePaHC – v. 0.9).

Søren Wichmann and Eric W. Holman. 2009. Population size and rates of language change. *Human Biology*, 81:259–274.

Søren Wichmann. 2008. The emerging field of language dynamics. *Language & Linguistics Compass*, 2:1294–1297.

Henri Wittmann. 1983. Les réactions en chaîne en morphologie diachronique ("Chain reactions in diachronic morphology"). In *Actes du colloque de la Societé Internationale de Linguistique Fonctionnelle*, volume 10, pages 285–292, Québec, Canada. Université Laval.

Sewall Wright. 1929. The evolution of dominance. *The American Naturalist*, 63:556–561.

Sewall Wright. 1955. Classification of the factors of evolution. *Cold Spring Harbor Symposia on Quantitative Biology*, 20:16–24.

On-line Active Reward Learning for Policy Optimisation in Spoken Dialogue Systems

Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen and Steve Young
Department of Engineering, University of Cambridge, Cambridge, UK
{phs26, mg436, nm480, lmr46, su259, djv27, thw28, sjy}@cam.ac.uk

Abstract

The ability to compute an accurate reward function is essential for optimising a dialogue policy via reinforcement learning. In real-world applications, using explicit user feedback as the reward signal is often unreliable and costly to collect. This problem can be mitigated if the user's intent is known in advance or data is available to pre-train a task success predictor off-line. In practice neither of these apply for most real world applications. Here we propose an on-line learning framework whereby the dialogue policy is jointly trained alongside the reward model via active learning with a Gaussian process model. This Gaussian process operates on a continuous space dialogue representation generated in an unsupervised fashion using a recurrent neural network encoder-decoder. The experimental results demonstrate that the proposed framework is able to significantly reduce data annotation costs and mitigate noisy user feedback in dialogue policy learning.

1 Introduction

Spoken Dialogue Systems (SDS) allow human-computer interaction using natural speech. They can be broadly divided into two categories: chat-oriented systems which aim to converse with users and provide reasonable contextually relevant responses (Vinyals and Le, 2015; Serban et al., 2015), and task-oriented systems designed to assist users to achieve specific goals (e.g. find hotels, movies or bus schedules) (Daubigney et al., 2014; Young et al., 2013). The latter are typically designed according to a structured *ontology* (or a database *schema*), which defines the domain that the system can talk about. Teaching a system how to respond appropriately in a task-oriented SDS is non-trivial. This *dialogue management* task is often formulated as a manually defined dialogue flow that directly determines the quality of interaction. More recently, dialogue management has been formulated as a reinforcement learning (RL) problem which can be automatically optimised (Levin and Pieraccini, 1997; Roy et al., 2000; Williams and Young, 2007; Young et al., 2013). In this framework, the system learns by a *trial and error* process governed by a potentially delayed learning objective defined by a *reward function*.

TASK: Ask for an *expensive English* restaurant
Request its *phone number* and *address*

S: Hello, how may I help you?
U: I want an expensive place that serves English food.
S: Cote is a nice expensive restaurant with English food.
U: What is the phone number?
S: Its number is 01223 311053.
U: Thanks for the help, goodbye.
S: Thank you, goodbye!

S: System
U: User

EVALUATION:
- Objective Rating: Fail (*address* not mentioned)
- Subjective Rating: Success (get all info he asked)

Figure 1: An example of a task-oriented dialogue with a pre-defined task and the evaluation results.

A typical approach to defining the reward function in a task-oriented dialogue system is to apply a small per-turn penalty to encourage short dialogues and to give a large positive reward at the end of each successful interaction. Figure 1 is an example of a dialogue task which is typically set for users who are being paid to converse with the system. When users are primed with a specific task to complete, dialogue success can be determined from subjective user ratings (*Subj*), or

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2431–2441,
Berlin, Germany, August 7-12, 2016.

an objective measure (*Obj*) based on whether or not the pre-specified task was completed (Walker et al., 1997; Gašić et al., 2013). However, prior knowledge of the user's goal is not normally available in real situations, making the objective reward estimation approach impractical.

Furthermore, objective ratings are inflexible and often fail as can be seen from Figure 1, if the user does not strictly follow the task. This results in a mismatch between the *Obj* and *Subj* ratings. However, relying on subjective ratings alone is also problematic since crowd-sourced subjects frequently give inaccurate responses and real users are often unwilling to extend the interaction in order to give feedback, resulting in unstable learning (Zhao et al., 2011; Gašić et al., 2011). In order to filter out incorrect user feedback, Gašić et al. (2013) used only dialogues for which $Obj = Subj$. Nonetheless, this is inefficient and not feasible anyway in most real-world tasks where the user's goal is generally unknown and difficult to infer.

In light of the above, Su et al. (2015a) proposed learning a neural network-based *Obj* estimator from off-line simulated dialogue data. This removes the need for the *Obj* check during on-line policy learning and the resulting policy is as effective as one trained with dialogues using the $Obj = Subj$ check. However, a user simulator will only provide a rough approximation of real user statistics and developing a user simulator is a costly process (Schatzmann et al., 2006).

To deal with the above issues, this paper describes an on-line active learning method in which users are asked to provide feedback on whether the dialogue was successful or not. However, active learning is used to limit requests for feedback to only those cases where the feedback would be useful, and also a noise model is introduced to compensate for cases where the user feedback is inaccurate. A Gaussian process classification (GPC) model is utilised to robustly model the uncertainty presented by the noisy user feedback. Since GPC operates on a fixed-length observation space and dialogues are of variable-length, a recurrent neural network (RNN)-based embedding function is used to provide fixed-length dialogue representations. In essence, the proposed method learns a dialogue policy and a reward estimator on-line from scratch, and is directly applicable to real-world applications.

The rest of the paper is organised as follows. The next section gives an overview of related work. The proposed framework is then described in §3. This consists of the policy learning algorithm, the creation of the dialogue embedding function and the active reward model trained from real user ratings. In §4, the proposed approach is evaluated in the context of an application providing restaurant information in Cambridge, UK. We first give an in-depth analysis of the dialogue embedding space. The results of the active reward model when it is trained together with a dialogue policy on-line with real users are then presented. Finally, our conclusions are presented in §5.

2 Related Work

Dialogue evaluation has been an active research area since late 90s. Walker et al. (1997) proposed the PARADISE framework, where a linear function of task completion and various dialogue features such as dialogue duration were used to infer user satisfaction. This measure was later used as a reward function for learning a dialogue policy (Rieser and Lemon, 2011). However, as noted, task completion is rarely available when the system is interacting with real users and also concerns have been raised regarding the theoretical validity of the model (Larsen, 2003).

Several approaches have been adopted for learning a dialogue reward model given a corpus of annotated dialogues. Yang et al. (2012) used collaborative filtering to infer user preferences. The use of reward shaping has also been investigated in (El Asri et al., 2014; Su et al., 2015b) to enrich the reward function in order to speed up dialogue policy learning. Also, Ultes and Minker (2015) demonstrated that there is a strong correlation between expert's user satisfaction ratings and dialogue success. However, all these methods assume the availability of reliable dialogue annotations such as expert ratings, which in practice are hard to obtain.

One effective way to mitigate the effects of annotator error is to obtain multiple ratings for the same data and several methods have been developed to guide the annotation process with uncertainty models (Dai et al., 2013; Lin et al., 2014). Active learning is particularly useful for determining when an annotation is needed (Settles, 2010; Zhang and Chaudhuri, 2015). It is often utilised using Bayesian optimisation approaches (Brochu et al., 2010). Based on this, Daniel et al. (2014)

exploited a pool-based active learning method for a robotics application. They queried the user for feedback on the most informative sample collected so far and showed the effectiveness of this method.

Rather than explicitly defining a reward function, inverse RL (IRL) aims to recover the underlying reward from demonstrations of good behaviour and then learn a policy which maximises the recovered reward (Russell, 1998). IRL was first introduced to SDS in (Paek and Pieraccini, 2008), where the reward was inferred from human-human dialogues to mimic the behaviour observed in a corpus. IRL has also been studied in a Wizard-of-Oz (WoZ) setting (Boularias et al., 2010; Rojas Barahona and Cerisara, 2014), where typically a human expert served as the dialogue manager to select each system reply based on the speech understanding output at different noise levels. However, this approach is costly and there is no reason to suppose that a human wizard is acting optimally, especially at high noise levels.

Since humans are better at giving relative judgements than absolute scores, another related line of research has focused on preference-based approaches to RL (Cheng et al., 2011). In (Sugiyama et al., 2012), users were asked to provide rankings between pairs of dialogues. However, this is also costly and does not scale well in real applications.

3 Proposed Framework

The proposed system framework is depicted in Figure 2. It is divided into three main parts: a dialogue policy, a dialogue embedding function, and an active reward model of user feedback. When each dialogue ends, a set of turn-level features $\mathbf{f_t}$ is extracted and fed into an embedding function σ to obtain a fixed-dimension dialogue representation $\mathbf{d}$ that serves as the input space of the reward model R. This reward is modelled as a Gaussian process which for every input point provides an estimate of task success along with a measure of the estimate uncertainty. Based on this uncertainty, R decides whether to query the user for feedback or not. It then returns a reinforcement signal to update the dialogue policy π, which is trained using the GP-SARSA algorithm (Gašić and Young, 2014). GP-SARSA also deploys Gaussian process estimation to provide an on-line sample-efficient reinforcement learning algorithm capable of bootstrapping estimates of sparse value functions from minimal numbers of samples (dialogues). The quality of each dialogue is defined by its cumulative reward, where each dialogue turn incurs a small negative reward (-1) and the final reward of either 0 or 20 depending on the estimate of task success are provided by the reward model.

Note that the key contribution here is to learn the noise robust reward model and the dialogue policy simultaneously on-line, using the user as a 'supervisor'. Active learning is not an essential component of the framework but highly desirable in practice to minimise the impact of the supervision burden on users. The use of a pre-trained embedding function is a sub-component of the proposed approach and is trained off-line on corpus data rather than manually designed here.

3.1 Unsupervised Dialogue Embeddings

In order to model user feedback over dialogues of varying length, an embedding function is used to map each dialogue into a fixed-dimensional continuous-space. The use of embedding functions has recently gained attention especially for word representations, and has boosted performance on several natural language processing tasks (Mikolov et al., 2013; Turian et al., 2010; Levy and Goldberg, 2014). Embedding has also been successfully applied to machine translation (MT) where it enables varying-length phrases to be mapped to fixed-length vectors using an RNN Encoder-Decoder (Cho et al., 2014). Similar to MT, dialogue embedding enables variable length sequences of utterances to be mapped into an appropriate fixed-length vector. Although embedding is used here to create a fixed-dimension input space for the GPC-based task success classifier, it should be noted that it potentially facilitates a variety of other downstream tasks which depend on classification or clustering.

The model structure of the embedding function is described on the left of Figure 2, where the episodic turn-level features $\mathbf{f_t}$ are extracted from a dialogue and serve as the input features to the encoder. In our proposed model, the encoder is a Bi-directional Long Short-Term Memory network (BLSTM) (Hochreiter and Schmidhuber, 1997; Graves et al., 2013). The LSTM is a Recurrent Neural Network (RNN) with gated recurrent units introduced to alleviate the vanishing gradient problem. The BLSTM encoder takes into account the sequential information from both directions of the input data, computing the *forward*

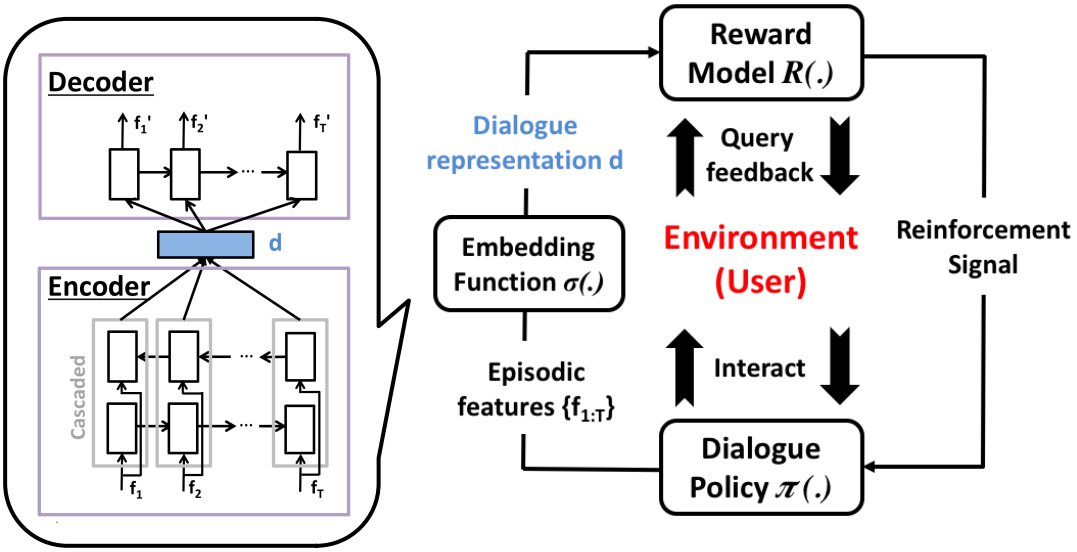


Figure 2: Schematic of the system framework. The three main system components dialogue policy, dialogue embedding creation, and reward modelling based on user feedback, are described in §3.

hidden sequences $\overrightarrow{\mathbf{h}}_{1:T}$ and the *backward* hidden sequences $\overleftarrow{\mathbf{h}}_{T:1}$ while iterating over all input features $\mathbf{f_t}$, $t = 1, ..., T$:

$$\overrightarrow{\mathbf{h_t}} = LSTM(\mathbf{f_t}, \overrightarrow{\mathbf{h}}_{t-1})$$
$$\overleftarrow{\mathbf{h_t}} = LSTM(\mathbf{f_t}, \overleftarrow{\mathbf{h}}_{t+1})$$

where $LSTM$ denotes the activation function. The dialogue representation $\mathbf{d}$ is then calculated as the average over all hidden sequences:

$$\mathbf{d} = \frac{1}{T}\sum_{t=1}^{T} \mathbf{h_t} \quad (1)$$

where $\mathbf{h_t} = [\overrightarrow{\mathbf{h_t}}; \overleftarrow{\mathbf{h_t}}]$ is the concatenation of the two directional hidden sequences.

Given the dialogue representation $\mathbf{d}$ output by the encoder, the decoder is a forward LSTM that takes $\mathbf{d}$ as its input for each turn t to produce the sequence of features $\mathbf{f'}_{1:T}$.

The training objective of the encoder-decoder minimises the mean-square-error (MSE) between the prediction $\mathbf{f'}_{1:T}$ and the output $\mathbf{f}_{1:T}$ (which is also the input):

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} ||\mathbf{f}_t - \mathbf{f}'_t||^2 \quad (2)$$

where N is the number of training dialogues and $||\cdot||^2$ denotes the l^2-norm. Since all the functions used in the encoder and decoder are differentiable, stochastic gradient decent (SGD) can be used to train the model.

The dialogue representations generated by this LSTM-based unsupervised embedding function are then used as the observations for the reward model described in the next section 3.2.

3.2 Active Reward Learning

A Gaussian process is a Bayesian non-parametric model that can be used for regression or classification (Rasmussen and Williams, 2006). It is particularly appealing since it can learn from a small number of observations by exploiting the correlations defined by a *kernel function* and it provides a measure of uncertainty of its estimates. In the context of spoken dialogue systems it has been successfully used for RL policy optimisation (Gašić and Young, 2014; Casanueva et al., 2015) and IRL reward function regression (Kim et al., 2014).

Here we propose modelling dialogue success as a Gaussian process (GP). This involves estimating the probability $p(y|\mathbf{d}, \mathcal{D})$ that the task was successful given the current dialogue representation $\mathbf{d}$ and the pool $\mathcal{D}$ containing previously classified dialogues. We pose this as a classification problem where the rating is a binary observation $y \in \{-1, 1\}$ that defines failure or success. The observations y are considered to be drawn from a Bernoulli distribution with a success probability $p(y = 1|\mathbf{d}, \mathcal{D})$. The probability is related to a latent function $f(\mathbf{d}|\mathcal{D}) : \mathcal{R}^{dim(\mathbf{d})} \rightarrow \mathcal{R}$ that is mapped to a unit interval by a *probit* function $p(y = 1|\mathbf{d}, \mathcal{D}) = \phi(f(\mathbf{d}|\mathcal{D}))$, where ϕ denotes the cumulative density function of the standard Gaus-

sian distribution.

The latent function is given a GP prior: $f(\mathbf{d}) \sim \mathcal{GP}(m(\mathbf{d}), k(\mathbf{d}, \mathbf{d}'))$, where $m(\cdot)$ is the mean function and $k(\cdot, \cdot)$ the covariance function (kernel). Here the stationary squared exponential kernel k_{SE} is used. It is also combined with a white noise kernel k_{WN} in order to account for the "noise" in users' ratings:

$$k(\mathbf{d}, \mathbf{d}') = p^2 \exp(-\frac{||\mathbf{d} - \mathbf{d}'||^2}{2l^2}) + \sigma_n^2 \quad (3)$$

where the first term denotes k_{SE} and the second term k_{WN}.

The *hyper-parameters* p, l, σ_n can be adequately optimised by maximising the marginal likelihood using a gradient-based method (Chen et al., 2015). Since $\phi(\cdot)$ is not Gaussian, the resulting posterior probability $p(y = 1|\mathbf{d}, \mathcal{D})$ is analytically intractable. So instead an approximation method, expectation propagation (EP), was used (Nickisch and Rasmussen, 2008).

Querying the user for feedback is costly and may impact negatively on the user experience. This impact can be reduced by using active learning informed by the uncertainty estimate of the GP model (Kapoor et al., 2007). This ensures that user feedback is only sought when the model is uncertain about its current prediction. For the current application, an on-line (stream-based) version of active learning is required.

An illustration of a 1-dimensional example is shown in Figure 3. Given the labelled data $\mathcal{D}$, the predictive posterior mean μ_* and posterior variance σ_*^2 of the latent value $f(\mathbf{d}_*)$ for the current dialogue representation $\mathbf{d}_*$ can be calculated. Then a threshold interval $[1 - \lambda, \lambda]$ is set on the predictive success probability $p(y_* = 1|\mathbf{d}_*, \mathcal{D}) = \phi(\mu_*/\sqrt{1 + \sigma_*^2})$ to decide whether this dialogue should be labelled or not. The decision boundary implicitly considers both the posterior mean as well as the variance.

When deploying this reward model in the proposed framework, a GP with a zero-mean prior for f is initialised and $\mathcal{D} = \{\}$. After the dialogue policy π completes each episode with the user, the generated dialogue turns are transformed into the dialogue representation $\mathbf{d} = \sigma(\mathbf{f}_{1:T})$ using the dialogue embedding function σ. Given $\mathbf{d}$, the predictive mean and variance of $f(\mathbf{d}|\mathcal{D})$ are determined, and the reward model decides whether or not it should seek user feedback based on the threshold λ on $\phi(f(\mathbf{d}|\mathcal{D}))$. If the model is uncertain, the user's feedback on the current episode $\mathbf{d}$ is used to update the GP model and to generate the reinforcement signal for training the policy π; otherwise the predictive success rating from the reward model is used directly to update the policy. This process takes place after each dialogue.

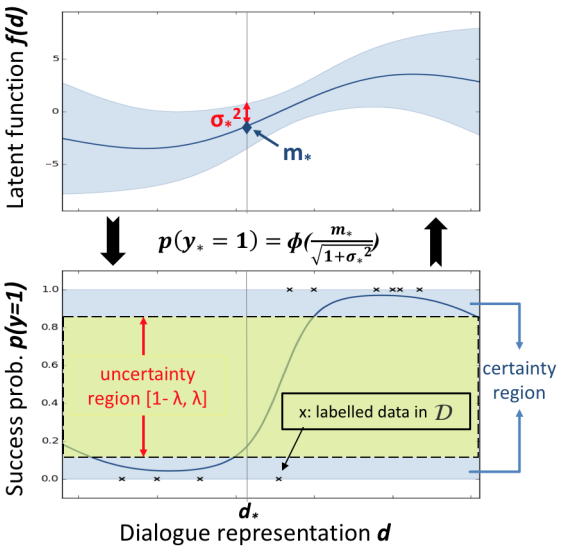


Figure 3: 1-dimensional example of the proposed GP active reward learning model.

4 Experimental results

The target application is a live telephone-based spoken dialogue system providing restaurant information for the Cambridge (UK) area. The domain consists of approximately 150 venues each having 6 slots (attributes) of which 3 can be used by the system to constrain the search (food-type, area and price-range) and the remaining 3 are informable properties (phone-number, address and postcode) available once a required database entity has been found.

The shared core components of the SDS common to all experiments comprise a HMM-based recogniser, a confusion network (CNet) semantic input decoder (Henderson et al., 2012), the BUDS belief state tracker (Thomson and Young, 2010) that factorises the dialogue state using a dynamic Bayesian network, and a template based natural language generator to map system semantic actions into natural language responses to the user. All policies were trained using the GP-SARSA algorithm and the summary action space of the RL policy contains 20 actions.

The reward given to each dialogue was set to $20 \times \mathbb{1}_{success} - N$, where N is the dialogue turn

number and $\mathbb{1}$ is the indicator function for dialogue success, which is determined by different methods as described in the following section. These rewards constitute the reinforcement signal used for policy learning.

4.1 Dialogue representations

The LSTM Encoder-Decoder model described in §3.1 was used to generate an embedding $\mathbf{d}$ for each dialogue. For each dialogue turn that contains a user's utterance and a system's response, a feature vector $\mathbf{f}$ of size 74 was extracted (Vandyke et al., 2015). This vector consists of the concatenation of the most likely user intention determined by the semantic decoder, the distribution over each concept of interest defined in the ontology, a one-hot encoding of the system's reply action, and the turn number normalised by the maximum number of turns (here 30). This feature vector was used as the input and the target for the LSTM Encoder-Decoder model, where the training objective was to minimise the MSE of the reconstruction loss.

The model was implemented using the Theano library (Bergstra et al., 2010; Bastien et al., 2012). A corpus consisting of 8565, 1199 and 650 real user dialogues in the Cambridge restaurant domain was used for training, validation and testing respectively. This corpus was collected via the Amazon Mechanical Turk (AMT) service, where paid subjects interacted with the dialogue system. The sizes of $\overrightarrow{\mathbf{h_t}}$ and $\overleftarrow{\mathbf{h_t}}$ in the encoder and the hidden layer in the decoder were all 32, resulting in $dim(\mathbf{h_t}) = dim(\mathbf{d}) = 64$. SGD per dialogue was used during backpropagation to train each model. In order to prevent over-fitting, *early stopping* was applied based on the held-out validation set.

In order to visualise the impact of the embeddings, the dialogue representations of all the 650 test dialogues were transformed by the embedding function in Figure 4 and reduced to two dimensions using t-SNE (Van der Maaten and Hinton, 2008). For each dialogue sample, the shape indicates whether or not the dialogue was successful, and the colour indicates the length of the dialogue (maximum 30 turns).

From the figure we can clearly see the colour gradient from the top left (shorter dialogues) to the bottom right (longer dialogues) for the positive *Subj* labels. This shows that dialogue length was one of the prominent features in the dialogue representation $\mathbf{d}$. It can also be seen that the longer

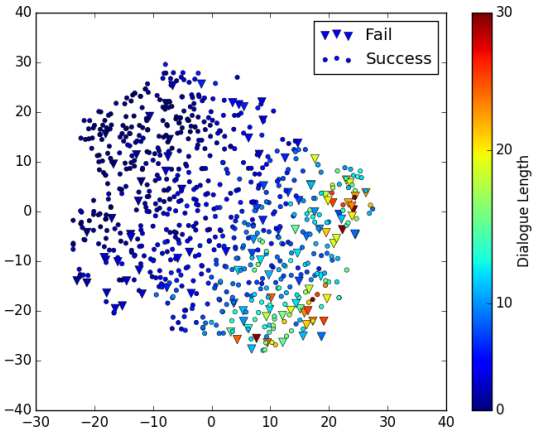


Figure 4: t-SNE visualisation on the unsupervised dialogue representation of the real user data in the Cambridge restaurant domain. Labels are the subjective ratings from the users.

failed dialogues (more than 15 turns) are located close to each other, mostly at the bottom right. On the other hand, there are other failed dialogues which are spread throughout the cluster. We can also see that the successful dialogues were on average shorter than 10 turns, which is consistent with the claim that users do not engage in longer dialogues with well-trained task-oriented systems.

This visualisation shows the potential of the unsupervised dialogue embedding since the transformed dialogue representations appear to be correlated with dialogue success in the majority of cases. For the purpose of GP reward modelling, this LSTM Encoder-Decoder embedding function appears therefore to be suitable for extracting an adequate fixed-dimension dialogue representation.

4.2 Dialogue Policy Learning

Given the well-trained dialogue embedding function, the proposed GP reward model operates on this input space. The system was implemented using the GPy library (Hensman et al., 2012). Given the predictive success probability of each newly seen dialogue, the threshold λ for the uncertainty region was initially set to 1 to encourage label querying and annealed to 0.85 for the first 50 collected dialogues and then set to 0.85 thereafter.

Initially, as each new dialogue was added to the training set, the hyper-parameters that defined the structure of the kernels mentioned in Eqn. 3 were optimised to minimise the negative log marginal likelihood using conjugate gradient as-

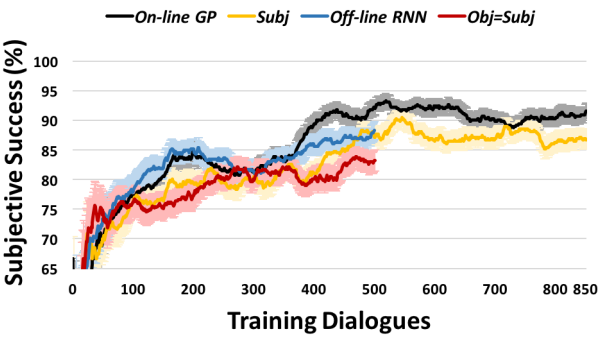


Figure 5: Learning curves showing subjective success as a function of the number of training dialogues used during on-line policy optimisation. The *on-line GP*, *Subj*, *off-line RNN* and *Obj=Subj* systems are shown as black, yellow, blue, and red lines. The light-coloured areas are one standard error intervals.

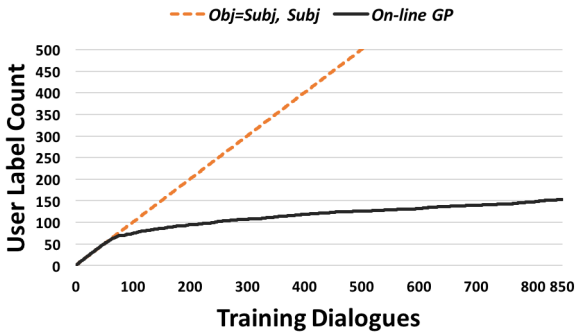


Figure 6: The number of times each system queries the user for feedback during on-line policy optimisation as a function of the number of training dialogues. The orange line represents both the *Obj=Subj* and *Subj* systems, and the black line represents the *on-line GP* system.

cent (Rasmussen and Williams, 2006). To prevent *overfitting*, after the first 40 dialogues, these hyper-parameters were only re-optimised after every batch of 20 dialogues.

To investigate the performance of the proposed *on-line GP* policy learning, three other contrasting systems were also tested. Note that the handcrafted system is not compared since it does not scale to larger domains and is sensitive to speech recognition errors. In each case, the only difference was the method used to compute the reward:

- the *Obj=Subj* system which uses prior knowledge of the task to only use training dialogues for which the user's subjective assessment of success is consistent with the objective assessment of success as in (Gašić et al., 2013).
- the *Subj* system which directly optimises the policy using only the user assessment of success whether accurate or not.
- the *off-line RNN* system that uses 1K simulated data and the corresponding *Obj* labels to train an RNN success estimator as in (Su et al., 2015a).

For the *Subj* system rating, in order to focus solely on the performance of the policy rather than other aspects of the system such as the fluency of the reply sentence, users were asked to rate dialogue success by answering the following question: *Did you find all the information you were looking for?*

All four of the above systems were trained with a total of 500 dialogues on-line by users recruited via the AMT service. Figure 5 shows the on-line learning curve of the subjective success rating when during training. For each system, the moving average was calculated using a window of 150 dialogues. In each case, three distinct policies were trained and the results were averaged to reduce noise.

As can be seen, all four systems perform better than 80 % subjective success rate after approximately 500 training dialogues. The *Obj=Subj* system is relatively poor compared to the others. This might be because users often report success even though the objective evaluation indicates failure. In such cases, the dialogue is discarded and not used for training. As a consequence, the *Obj=Subj* system required approximately 700 dialogues in order to obtain 500 which were useful, whereas all other systems made use of every dialogue.

To investigate learning behaviour over longer spans, training for the *on-line GP* and the *Subj* systems was extended to 850 dialogues. As can be seen, performance in both cases is broadly flat.

Similar to the conclusions drawn in (Gašić et al., 2011), the *Subj* system suffers from unreliable user feedback. Firstly, as in the *Obj=Subj* system, users forget the full requirements of the task and in particular, forget to ask for all required information. Secondly, users give inconsistent feedback due to a lack of proper care and attention. From Figure 5 it can be clearly seen that the *on-line GP* system consistently performed better than

Subj system, presumably, because its noise model mitigates the effect of inconsistency in user feedback. Of course, unlike crowd-sourced subjects, real users might provide more consistent feedback, but nevertheless, some inconsistency is inevitable and the noise model offers the needed robustness.

The advantage of the *on-line GP* system in reducing the number of times that the system requests user feedback (i.e. the label cost) can be seen in Figure 6. The black curve shows the number of active learning queries triggered in the *on-line GP* system averaged across the three policies. This system required only 150 user feedback requests to train a robust reward model. On the other hand, the *Obj=Subj* and *Subj* systems require user feedback for every training dialogue as shown by the dashed orange line.

Of course, the *off-line RNN* system required no user feedback at all when training the system on-line since it had the benefit of prior access to a user simulator. Its performance during training after the first 300 dialogues was, however, inferior to the *on-line GP* system.

4.3 Dialogue Policy Evaluation

In order to compare performance, the averaged results obtained between 400-500 training dialogues are shown in the first section of Table 1 along with one standard error. For the 400-500 interval, the *Subj*, *off-line RNN* and *on-line GP* systems achieved comparable results without statistical differences. The results of continuing training on the *Subj* and *on-line GP* systems from 500 to 850 training dialogues are also shown. As can be seen, the *on-line GP* system was significantly better presumably because it is more robust to erroneous user feedback compared to the *Subj* system.

4.4 Reward Model Evaluation

The above results verify the effectiveness of the proposed reward model for policy learning. Here we investigate further the accuracy of the model in predicting the subjective success rate. An evaluation of the *on-line GP* reward model between 1 and 850 training dialogues is presented in Table 2.

Since three reward models were learnt each with 850 dialogues, there were a total of 2550 training dialogues. Of these, the models queried the user for feedback a total of 454 times, leaving 2096 dialogues for which learning relied on the reward model's prediction. The results shown in the table are thus the average over 2096 dialogues.

Table 1: Subjective evaluation of the *Obj=Subj*, *off-line RNN*, *Subj* and *on-line GP* system during different stages of on-line policy learning. *Subjective*: user binary rating on dialogue success. Statistical significance was calculated using a two-tailed Students t-test with p-value of 0.05.

Dialogues	**Reward Model**	***Subjective*** *(%)*
400-500	*Obj=Subj*	85.0 ± 2.1
	off-line RNN	89.0 ± 1.8
	Subj	90.7 ± 1.7
	on-line GP	91.7 ± 1.6
500-850	*Subj*	87.1 ± 1.0
	on-line GP	$\mathbf{90.9 \pm 0.9}$*

$^* p < 0.05$

As can be seen, there was a significant imbalance between success and fail labels since the policy was improving along with the training dialogues. This lowered the recall on failed dialogue prediction as the model was biased to data with positive labels. Nevertheless, its precision scores well. On the other hand, the successful dialogues were accurately predicted by the proposed model.

Table 2: Statistical evaluation of the prediction of the *on-line GP* systems with respect to *Subj* rating.

Subj	**Prec.**	**Recall**	**F-measure**	**Number**
Fail	1.00	0.52	0.68	204
Suc.	0.95	1.00	0.97	1892
Total	0.96	0.95	0.95	2096

4.5 Example Dialogues

The key benefits of the *on-line GP* reward model compared to other models are its robustness to noise and efficient use of user supervision. Since the four systems compared above differ only in the design of reward model (learning objective), their on-line behaviours were broadly similar.

Two example dialogues between users and the *on-line GP* system are listed in Table 3 to illustrate how the system operates under different noise conditions. The user's subjective rating and the rating determined by the *on-line GP* reward model are also shown. The labels '*n*-th ASR' and '*n*-th SEM' indicate the n-th most likely hypotheses from speech recogniser and semantic decoder respectively.

Table 3: Example dialogues between on-line users and the proposed *on-line GP* system.

-Low noise condition- (User Subjective Evaluation: *Success***, on-line GP Evaluation:** *Success***)**	
Machine	: Hello, welcome to the Cambridge dialogue system, What kind of food would you like?
Human	: [Top ASR] Yeah I want to find a restaurant that serves European food.
	[Top SEM] inform(food=european,type=restaurant)
Machine	: Hotel du Vin and Bistro is a nice place. It serves European food.
Human	: [Top ASR] Uh what is the address and phone number?
	[Top SEM] request(phone,addr)
Machine	: Hotel du Vin and Bistro is a nice place. Their address is 15-19 Trumpington Street and their phone is 01223 227330.
Human	: [Top ASR] Thank you goodbye.
	[Top SEM] bye()
Machine	: Thank you for using this system.
-Higher noise condition- (User Subjective Evaluation: *Success***, on-line GP Evaluation:** *Success***)**	
Machine	: Hello, welcome to the Cambridge dialogue system, What kind of food would you like?
Human	: [Top ASR] And I want to find a expensive restaurant it should be in the south part of town.
	[Top SEM] inform(area=south,pricerange=expensive,type=restaurant)
Machine	: Gourmet burger kitchen is a nice place it is in the expensive price range.
Human	: [Top ASR] And do you have something else?
	[Top SEM] null()
	[2nd SEM] reqalts()
Machine	: Did you say you are looking for a venue in the area of south ?
Human	: [Top ASR] Yes.
	[Top SEM] affirm()
Machine	: Frankie and Benny's is a nice place, it is in the south part of town and it is in the expensive price range.
Human	: [Top ASR] Okay can I have the uh address phone number called for any.
	[Top SEM] request(addr)
	[2nd SEM] request(phone,addr)
Machine	: Frankie and Benny's is a nice place. Their address is Clifton Way and their phone number is 01223 412430.
Human	: [Top ASR] What is the pub called for free you any?
	...
	[5th ASR] What is the post code for free you any?
	[Top SEM] request(postcode)
Machine	: Frankie and Benny's is a nice place. Their postcode is C.B 1, 7 D.Y.
Human	: [Top ASR] Thank you goodbye.
	[Top SEM] bye()
Machine	: Thank you for using this system.

5 Conclusion

In this paper we have proposed an active reward learning model using Gaussian process classification and an unsupervised neural network-based dialogue embedding to enable truly on-line policy learning in spoken dialogue systems. The system enables stable policy optimisation by robustly modelling the inherent noise in real user feedback and uses active learning to minimise the number of feedback requests to the user. We found that the proposed model achieved efficient policy learning and better performance compared to other state-of-the-art methods in the Cambridge restaurant domain. A key advantage of this Bayesian model is that its uncertainty estimate allows active learning and noise handling in a natural way. The unsupervised dialogue embedding function required no labelled data to train whilst providing a compact and useful input to the reward predictor. Overall, the techniques developed in this paper enable for the first time a viable approach to on-line learning in deployed real-world dialogue systems which does not need a large corpus of manually annotated data or the construction of a user simulator.

Consistent with all of our previous work, the reward function studied here is focused primarily on task success. This may be too simplistic for many commercial applications and further work will be needed in conjunction with human interaction experts to identify and incorporate the extra dimensions of dialogue quality that will be needed to achieve the highest levels of user satisfaction.

Acknowledgments

Pei-Hao Su is supported by Cambridge Trust and the Ministry of Education, Taiwan. This research was partly funded by the EPSRC grant EP/M018946/1 *Open Domain Statistical Spoken Dialogue Systems*. The data used in the experiments is available at *www.repository.cam.ac.uk/handle/1810/256020*.

References

[Bastien et al.2012] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. 2012. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS Workshop.

[Bergstra et al.2010] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*.

[Boularias et al.2010] Abdeslam Boularias, Hamid R Chinaei, and Brahim Chaib-draa. 2010. Learning the reward model of dialogue pomdps from data. In *NIPS Workshop on Machine Learning for Assistive Techniques*.

[Brochu et al.2010] Eric Brochu, Vlad M Cora, and Nando De Freitas. 2010. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*.

[Casanueva et al.2015] Iñigo Casanueva, Thomas Hain, Heidi Christensen, Ricard Marxer, and Phil Green. 2015. Knowledge transfer between speakers for personalised dialogue management. In *Proc of SigDial*.

[Chen et al.2015] Lu Chen, Pei-Hao Su, and Milica Gašic. 2015. Hyper-parameter optimisation of gaussian process reinforcement learning for statistical dialogue management. In *Proc of SigDial*.

[Cheng et al.2011] Weiwei Cheng, Johannes Fürnkranz, Eyke Hüllermeier, and Sang-Hyeun Park. 2011. Preference-based policy iteration: Leveraging preference learning for reinforcement learning. In *Machine learning and knowledge discovery in databases*. Springer.

[Cho et al.2014] Kyunghyun Cho, Bart van Merriënboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[Dai et al.2013] Peng Dai, Christopher H Lin, Daniel S Weld, et al. 2013. Pomdp-based control of workflows for crowdsourcing. *Artificial Intelligence*, 202.

[Daniel et al.2014] Christian Daniel, Malte Viering, Jan Metz, Oliver Kroemer, and Jan Peters. 2014. Active reward learning. In *Proc of RSS*.

[Daubigney et al.2014] Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2014. A comprehensive reinforcement learning framework for dialogue management optimisation. *Journal of Selected Topics in Signal Processing*, 6(8).

[El Asri et al.2014] Layla El Asri, Romain Laroche, and Olivier Pietquin. 2014. Task completion transfer learning for reward inference. In *Proc of MLIS*.

[Gašić and Young2014] Milica Gašić and Steve Young. 2014. Gaussian processes for pomdp-based dialogue manager optimization. *TASLP*, 22(1):28–40.

[Gašić et al.2011] Milica Gašić, Filip Jurcicek, Blaise. Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *IEEE ASRU*.

[Gašić et al.2013] Milica Gašić, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve J. Young. 2013. On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *Proc of ICASSP*.

[Graves et al.2013] Alax Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional lstm. In *IEEE ASRU*.

[Henderson et al.2012] Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. 2012. Discriminative spoken language understanding using word confusion networks. In *IEEE SLT*.

[Hensman et al.2012] James Hensman, Nicolo Fusi, Ricardo Andrade, Nicolas Durrande, Alan Saul, Max Zwiessele, and Neil D. Lawrence. 2012. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`.

[Hochreiter and Schmidhuber1997] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8).

[Kapoor et al.2007] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. 2007. Active learning with gaussian processes for object categorization. In *Proc of ICCV*.

[Kim et al.2014] Dongho Kim, Catherine Breslin, Pirros Tsiakoulis, Matthew Henderson, and Steve J Young. 2014. Inverse reinforcement learning for micro-turn management. In *IEEE SLT*.

[Larsen2003] L.B. Larsen. 2003. Issues in the evaluation of spoken dialogue systems using objective and subjective measures. In *IEEE ASRU*.

[Levin and Pieraccini1997] Esther Levin and Roberto Pieraccini. 1997. A stochastic model of computer-human interaction for learning dialogue strategies. *Eurospeech*.

[Levy and Goldberg2014] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *NIPS*.

[Lin et al.2014] Christopher H Lin, Daniel S Weld, et al. 2014. To re (label), or not to re (label). In *Second AAAI Conference on Human Computation and Crowdsourcing*.

[Mikolov et al.2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

[Nickisch and Rasmussen2008] Hannes Nickisch and Carl Edward Rasmussen. 2008. Approximations for binary gaussian process classification. *JMLR*, 9(10).

[Paek and Pieraccini2008] Tim Paek and Roberto Pieraccini. 2008. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech communication*, 50.

[Rasmussen and Williams2006] Carl Edward Rasmussen and Chris Williams. 2006. Gaussian processes for machine learning.

[Rieser and Lemon2011] Verena Rieser and Oliver Lemon. 2011. Learning and evaluation of dialogue strategies for new applications: Empirical methods for optimization from small data sets. *Computational Linguistics*, 37(1).

[Rojas Barahona and Cerisara2014] Lina Maria Rojas Barahona and Christophe Cerisara. 2014. Bayesian Inverse Reinforcement Learning for Modeling Conversational Agents in a Virtual Environment. In *Conference on Intelligent Text Processing and Computational Linguistics*.

[Roy et al.2000] Nicholas Roy, Joelle Pineau, and Sebastian Thrun. 2000. Spoken dialogue management using probabilistic reasoning. In *Proc of SigDial*.

[Russell1998] Stuart Russell. 1998. Learning agents for uncertain environments. In *Proc of COLT*.

[Schatzmann et al.2006] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. 2006. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(02):97–126.

[Serban et al.2015] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*.

[Settles2010] Burr Settles. 2010. Active learning literature survey. *Computer Sciences Technical Report 1648*.

[Su et al.2015a] Pei-Hao Su, David Vandyke, Milica Gašić, Dongho Kim, Nikola Mrkšić, Tsung-Hsien Wen, and Steve Young. 2015a. Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Proc of Interspeech*.

[Su et al.2015b] Pei-Hao Su, David Vandyke, Milica Gašić, Nikola Mrkšić, Tsung-Hsien Wen, and Steve Young. 2015b. Reward shaping with recurrent neural networks for speeding up on-line policy learning in spoken dialogue systems. In *Proc of SigDial*.

[Sugiyama et al.2012] Hiroaki Sugiyama, Toyomi Meguro, and Yasuhiro Minami. 2012. Preference-learning based inverse reinforcement learning for dialog control. In *Proc of Interspeech*.

[Thomson and Young2010] Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech and Language*, 24:562–588.

[Turian et al.2010] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc of ACL*.

[Ultes and Minker2015] Stefan Ultes and Wolfgang Minker. 2015. Quality-adaptive spoken dialogue initiative selection and implications on reward modelling. In *Proc of SigDial*.

[Van der Maaten and Hinton2008] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *JMLR*, 9:85.

[Vandyke et al.2015] David Vandyke, Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialogue success classifiers for policy training. In *IEEE ASRU*.

[Vinyals and Le2015] Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

[Walker et al.1997] Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. PARADISE: A framework for evaluating spoken dialogue agents. In *Proc of EACL*.

[Williams and Young2007] Jason D. Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech and Language*, 21(2):393–422.

[Yang et al.2012] Zhaojun Yang, G Levow, and Helen Meng. 2012. Predicting user satisfaction in spoken dialog system evaluation with collaborative filtering. *IEEE Journal of Selected Topics in Signal Processing*, 6(99):971–981.

[Young et al.2013] Steve Young, Milica Gašic, Blaise Thomson, and Jason Williams. 2013. Pomdp-based statistical spoken dialogue systems: a review. In *Proc of IEEE*, volume 99, pages 1–20.

[Zhang and Chaudhuri2015] Chicheng Zhang and Kamalika Chaudhuri. 2015. Active learning from weak and strong labelers. *CoRR*, abs/1510.02847.

[Zhao et al.2011] Liyue Zhao, Gita Sukthankar, and Rahul Sukthankar. 2011. Incremental relabeling for active learning with noisy crowdsourced annotations. In *Proc of PASSAT and Proc of SocialCom*.

Globally Normalized Transition-Based Neural Networks

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn,
Alessandro Presta, Kuzman Ganchev, Slav Petrov and Michael Collins*
Google Inc
New York, NY
{andor,chrisalberti,djweiss,severyn,apresta,kuzman,slav,mjcollins}@google.com

Abstract

We introduce a globally normalized transition-based neural network model that achieves state-of-the-art part-of-speech tagging, dependency parsing and sentence compression results. Our model is a simple feed-forward neural network that operates on a task-specific transition system, yet achieves comparable or better accuracies than recurrent models. We discuss the importance of global as opposed to local normalization: a key insight is that the label bias problem implies that globally normalized models can be strictly more expressive than locally normalized models.

1 Introduction

Neural network approaches have taken the field of natural language processing (NLP) by storm. In particular, variants of long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997) have produced impressive results on some of the classic NLP tasks such as part-of-speech tagging (Ling et al., 2015), syntactic parsing (Vinyals et al., 2015) and semantic role labeling (Zhou and Xu, 2015). One might speculate that it is the recurrent nature of these models that enables these results.

In this work we demonstrate that simple feed-forward networks without any recurrence can achieve comparable or better accuracies than LSTMs, as long as they are globally normalized. Our model, described in detail in Section 2, uses a transition system (Nivre, 2006) and feature embeddings as introduced by Chen and Manning (2014). We do not use any recurrence, but perform beam search for maintaining multiple hypotheses and introduce global normalization with a conditional random field (CRF) objective (Bottou et al., 1997; Le Cun et al., 1998; Lafferty et al., 2001; Collobert et al., 2011) to overcome the label bias problem that locally normalized models suffer from. Since we use beam inference, we approximate the partition function by summing over the elements in the beam, and use early updates (Collins and Roark, 2004; Zhou et al., 2015). We compute gradients based on this approximate global normalization and perform full backpropagation training of all neural network parameters based on the CRF loss.

In Section 3 we revisit the label bias problem and the implication that globally normalized models are strictly more expressive than locally normalized models. Lookahead features can partially mitigate this discrepancy, but cannot fully compensate for it—a point to which we return later. To empirically demonstrate the effectiveness of global normalization, we evaluate our model on part-of-speech tagging, syntactic dependency parsing and sentence compression (Section 4). Our model achieves state-of-the-art accuracy on all of these tasks, matching or outperforming LSTMs while being significantly faster. In particular for dependency parsing on the Wall Street Journal we achieve the best-ever published unlabeled attachment score of 94.61%.

As discussed in more detail in Section 5, we also outperform previous structured training approaches used for neural network transition-based parsing. Our ablation experiments show that we outperform Weiss et al. (2015) and Alberti et al. (2015) because we do global backpropagation training of all model parameters, while they fix the neural network parameters when training the global part of their model. We also outperform Zhou et al. (2015) despite using a smaller beam. To shed additional light on the label bias problem

* On leave from Columbia University.

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pages 2442–2452,
Berlin, Germany, August 7-12, 2016.

in practice, we provide a sentence compression example where the local model completely fails. We then demonstrate that a globally normalized parsing model without any lookahead features is almost as accurate as our best model, while a locally normalized model loses more than 10% absolute in accuracy because it cannot effectively incorporate evidence as it becomes available.

Finally, we provide an open-source implementation of our method, called SyntaxNet,[1] which we have integrated into the popular TensorFlow[2] framework. We also provide a pre-trained, state-of-the art English dependency parser called "Parsey McParseface," which we tuned for a balance of speed, simplicity, and accuracy.

2 Model

At its core, our model is an incremental transition-based parser (Nivre, 2006). To apply it to different tasks we only need to adjust the transition system and the input features.

2.1 Transition System

Given an input x, most often a sentence, we define:

- A set of states $\mathcal{S}(x)$.
- A special start state $s^\dagger \in \mathcal{S}(x)$.
- A set of allowed decisions $\mathcal{A}(s, x)$ for all $s \in \mathcal{S}(x)$.
- A transition function $t(s, d, x)$ returning a new state s' for any decision $d \in \mathcal{A}(s, x)$.

We will use a function $\rho(s, d, x; \theta)$ to compute the score of decision d in state s for input x. The vector θ contains the model parameters and we assume that $\rho(s, d, x; \theta)$ is differentiable with respect to θ.

In this section, for brevity, we will drop the dependence of x in the functions given above, simply writing $\mathcal{S}$, $\mathcal{A}(s)$, $t(s, d)$, and $\rho(s, d; \theta)$.

Throughout this work we will use transition systems in which all complete structures for the same input x have the same number of decisions $n(x)$ (or n for brevity). In dependency parsing for example, this is true for both the *arc-standard* and *arc-eager* transition systems (Nivre, 2006), where for a sentence x of length m, the number of decisions for any complete parse is $n(x) = 2 \times m$.[3]

A complete structure is then a sequence of decision/state pairs $(s_1, d_1) \dots (s_n, d_n)$ such that $s_1 = s^\dagger$, $d_i \in \mathcal{S}(s_i)$ for $i = 1 \dots n$, and $s_{i+1} = t(s_i, d_i)$. We use the notation $d_{1:j}$ to refer to a decision sequence $d_1 \dots d_j$.

We assume that there is a one-to-one mapping between decision sequences $d_{1:j-1}$ and states s_j: that is, we essentially assume that a state encodes the entire history of decisions. Thus, each state can be reached by a unique decision sequence from $s^\dagger$.[4] We will use decision sequences $d_{1:j-1}$ and states interchangeably: in a slight abuse of notation, we define $\rho(d_{1:j-1}, d; \theta)$ to be equal to $\rho(s, d; \theta)$ where s is the state reached by the decision sequence $d_{1:j-1}$.

The scoring function $\rho(s, d; \theta)$ can be defined in a number of ways. In this work, following Chen and Manning (2014), Weiss et al. (2015), and Zhou et al. (2015), we define it via a feed-forward neural network as

$$\rho(s, d; \theta) = \phi(s; \theta^{(l)}) \cdot \theta^{(d)}.$$

Here $\theta^{(l)}$ are the parameters of the neural network, excluding the parameters at the final layer. $\theta^{(d)}$ are the final layer parameters for decision d. $\phi(s; \theta^{(l)})$ is the representation for state s computed by the neural network under parameters $\theta^{(l)}$. Note that the score is linear in the parameters $\theta^{(d)}$. We next describe how softmax-style normalization can be performed at the local or global level.

2.2 Global vs. Local Normalization

In the Chen and Manning (2014) style of greedy neural network parsing, the conditional probability distribution over decisions d_j given context $d_{1:j-1}$ is defined as

$$p(d_j | d_{1:j-1}; \theta) = \frac{\exp \rho(d_{1:j-1}, d_j; \theta)}{Z_L(d_{1:j-1}; \theta)}, \quad (1)$$

where

$$Z_L(d_{1:j-1}; \theta) = \sum_{d' \in \mathcal{A}(d_{1:j-1})} \exp \rho(d_{1:j-1}, d'; \theta).$$

[1] http://github.com/tensorflow/models/tree/master/syntaxnet

[2] http://www.tensorflow.org

[3] Note that this is not true for the *swap* transition system defined in Nivre (2009).

[4] It is straightforward to extend the approach to make use of dynamic programming in the case where the same state can be reached by multiple decision sequences.

Each $Z_L(d_{1:j-1};\theta)$ is a *local* normalization term. The probability of a sequence of decisions $d_{1:n}$ is

$$p_L(d_{1:n}) = \prod_{j=1}^{n} p(d_j|d_{1:j-1};\theta) = \frac{\exp\sum_{j=1}^{n}\rho(d_{1:j-1},d_j;\theta)}{\prod_{j=1}^{n} Z_L(d_{1:j-1};\theta)}. \quad (2)$$

Beam search can be used to attempt to find the maximum of Eq. (2) with respect to $d_{1:n}$. The additive scores used in beam search are the log-softmax of each decision, $\ln p(d_j|d_{1:j-1};\theta)$, not the raw scores $\rho(d_{1:j-1},d_j;\theta)$.

In contrast, a Conditional Random Field (CRF) defines a distribution $p_G(d_{1:n})$ as follows:

$$p_G(d_{1:n}) = \frac{\exp\sum_{j=1}^{n}\rho(d_{1:j-1},d_j;\theta)}{Z_G(\theta)}, \quad (3)$$

where

$$Z_G(\theta) = \sum_{d'_{1:n}\in\mathcal{D}_n} \exp\sum_{j=1}^{n}\rho(d'_{1:j-1},d'_j;\theta)$$

and $\mathcal{D}_n$ is the set of all valid sequences of decisions of length n. $Z_G(\theta)$ is a *global* normalization term. The inference problem is now to find

$$\operatorname*{argmax}_{d_{1:n}\in\mathcal{D}_n} p_G(d_{1:n}) = \operatorname*{argmax}_{d_{1:n}\in\mathcal{D}_n} \sum_{j=1}^{n}\rho(d_{1:j-1},d_j;\theta).$$

Beam search can again be used to approximately find the argmax.

2.3 Training

Training data consists of inputs x paired with gold decision sequences $d^*_{1:n}$. We use stochastic gradient descent on the negative log-likelihood of the data under the model. Under a locally normalized model, the negative log-likelihood is

$$L_{\text{local}}(d^*_{1:n};\theta) = -\ln p_L(d^*_{1:n};\theta) = \quad (4)$$
$$-\sum_{j=1}^{n}\rho(d^*_{1:j-1},d^*_j;\theta) + \sum_{j=1}^{n}\ln Z_L(d^*_{1:j-1};\theta),$$

whereas under a globally normalized model it is

$$L_{\text{global}}(d^*_{1:n};\theta) = -\ln p_G(d^*_{1:n};\theta) =$$
$$-\sum_{j=1}^{n}\rho(d^*_{1:j-1},d^*_j;\theta) + \ln Z_G(\theta). \quad (5)$$

A significant practical advantange of the locally normalized cost Eq. (4) is that the local partition function Z_L and its derivative can usually be computed efficiently. In contrast, the Z_G term in Eq. (5) contains a sum over $d'_{1:n} \in \mathcal{D}_n$ that is in many cases intractable.

To make learning tractable with the globally normalized model, we use beam search and early updates (Collins and Roark, 2004; Zhou et al., 2015). As the training sequence is being decoded, we keep track of the location of the gold path in the beam. If the gold path falls out of the beam at step j, a stochastic gradient step is taken on the following objective:

$$L_{\text{global-beam}}(d^*_{1:j};\theta) =$$
$$-\sum_{i=1}^{j}\rho(d^*_{1:i-1},d^*_i;\theta) + \ln\sum_{d'_{1:j}\in\mathcal{B}_j}\exp\sum_{i=1}^{j}\rho(d'_{1:i-1},d'_i;\theta). \quad (6)$$

Here the set $\mathcal{B}_j$ contains all paths in the beam at step j, together with the gold path prefix $d^*_{1:j}$. It is straightforward to derive gradients of the loss in Eq. (6) and to back-propagate gradients to all levels of a neural network defining the score $\rho(s,d;\theta)$. If the gold path remains in the beam throughout decoding, a gradient step is performed using $\mathcal{B}_n$, the beam at the end of decoding.

3 The Label Bias Problem

Intuitively, we would like the model to be able to revise an earlier decision made during search, when later evidence becomes available that rules out the earlier decision as incorrect. At first glance, it might appear that a locally normalized model used in conjunction with beam search or exact search is able to revise earlier decisions. However the label bias problem (see Bottou (1991), Collins (1999) pages 222-226, Lafferty et al. (2001), Bottou and LeCun (2005), Smith and Johnson (2007)) means that locally normalized models often have a very weak ability to revise earlier decisions.

This section gives a formal perspective on the label bias problem, through a proof that globally normalized models are strictly more expressive than locally normalized models. The theorem was originally proved[5] by Smith and Johnson (2007).

[5]More precisely Smith and Johnson (2007) prove the theorem for models with potential functions of the form $\rho(d_{i-1},d_i,x_i)$; the generalization to potential functions of the form $\rho(d_{1:i-1},d_i,x_{1:i})$ is straightforward.

The example underlying the proof gives a clear illustration of the label bias problem.[6]

Global Models can be Strictly More Expressive than Local Models Consider a tagging problem where the task is to map an input sequence $x_{1:n}$ to a decision sequence $d_{1:n}$. First, consider a locally normalized model where we restrict the scoring function to access only the first i input symbols $x_{1:i}$ when scoring decision d_i. We will return to this restriction soon. The scoring function ρ can be an otherwise arbitrary function of the tuple $\langle d_{1:i-1}, d_i, x_{1:i}\rangle$:

$$p_L(d_{1:n}|x_{1:n}) = \prod_{i=1}^{n} p_L(d_i|d_{1:i-1}, x_{1:i}) = \frac{\exp\sum_{i=1}^{n}\rho(d_{1:i-1}, d_i, x_{1:i})}{\prod_{i=1}^{n} Z_L(d_{1:i-1}, x_{1:i})}.$$

Second, consider a globally normalized model

$$p_G(d_{1:n}|x_{1:n}) = \frac{\exp\sum_{i=1}^{n}\rho(d_{1:i-1}, d_i, x_{1:i})}{Z_G(x_{1:n})}.$$

This model again makes use of a scoring function $\rho(d_{1:i-1}, d_i, x_{1:i})$ restricted to the first i input symbols when scoring decision d_i.

Define $\mathcal{P}_L$ to be the set of all possible distributions $p_L(d_{1:n}|x_{1:n})$ under the local model obtained as the scores ρ vary. Similarly, define $\mathcal{P}_G$ to be the set of all possible distributions $p_G(d_{1:n}|x_{1:n})$ under the global model. Here a "distribution" is a function from a pair $(x_{1:n}, d_{1:n})$ to a probability $p(d_{1:n}|x_{1:n})$. Our main result is the following:

Theorem 3.1 *See also Smith and Johnson (2007). $\mathcal{P}_L$ is a strict subset of $\mathcal{P}_G$, that is $\mathcal{P}_L \subsetneq \mathcal{P}_G$.*

To prove this we will first prove that $\mathcal{P}_L \subseteq \mathcal{P}_G$. This step is straightforward. We then show that $\mathcal{P}_G \not\subseteq \mathcal{P}_L$; that is, there are distributions in $\mathcal{P}_G$ that are not in $\mathcal{P}_L$. The proof that $\mathcal{P}_G \not\subseteq \mathcal{P}_L$ gives a clear illustration of the label bias problem.

Proof that $\mathcal{P}_L \subseteq \mathcal{P}_G$: We need to show that for any locally normalized distribution p_L, we can construct a globally normalized model p_G such that $p_G = p_L$. Consider a locally normalized model with scores $\rho(d_{1:i-1}, d_i, x_{1:i})$. Define a global model p_G with scores

$$\rho'(d_{1:i-1}, d_i, x_{1:i}) = \log p_L(d_i|d_{1:i-1}, x_{1:i}).$$

Then it is easily verified that

$$p_G(d_{1:n}|x_{1:n}) = p_L(d_{1:n}|x_{1:n})$$

for all $x_{1:n}, d_{1:n}$. □

In proving $\mathcal{P}_G \not\subseteq \mathcal{P}_L$ we will use a simple problem where every example seen in training or test data is one of the following two tagged sentences:

$$x_1x_2x_3 = \text{a b c}, \quad d_1d_2d_3 = \text{A B C}$$
$$x_1x_2x_3 = \text{a b e}, \quad d_1d_2d_3 = \text{A D E} \qquad (7)$$

Note that the input $x_2 = \text{b}$ is ambiguous: it can take tags B or D. This ambiguity is resolved when the next input symbol, c or e, is observed.

Now consider a globally normalized model, where the scores $\rho(d_{1:i-1}, d_i, x_{1:i})$ are defined as follows. Define $\mathcal{T}$ as the set $\{(A, B), (B, C), (A, D), (D, E)\}$ of bigram tag transitions seen in the data. Similarly, define $\mathcal{E}$ as the set $\{(a, A), (b, B), (c, C), (b, D), (e, E)\}$ of (word, tag) pairs seen in the data. We define

$$\rho(d_{1:i-1}, d_i, x_{1:i}) = \alpha \times [\![(d_{i-1}, d_i) \in \mathcal{T}]\!] + \alpha \times [\![(x_i, d_i) \in \mathcal{E}]\!] \qquad (8)$$

where α is the single scalar parameter of the model, and $[\![\pi]\!] = 1$ if π is true, 0 otherwise.

Proof that $\mathcal{P}_G \not\subseteq \mathcal{P}_L$: We will construct a globally normalized model p_G such that there is no locally normalized model such that $p_L = p_G$.

Under the definition in Eq. (8), it is straightforward to show that

$$\lim_{\alpha\to\infty} p_G(\text{A B C}|\text{a b c}) = \lim_{\alpha\to\infty} p_G(\text{A D E}|\text{a b e}) = 1.$$

In contrast, under *any* definition for $\rho(d_{1:i-1}, d_i, x_{1:i})$, we must have

$$p_L(\text{A B C}|\text{a b c}) + p_L(\text{A D E}|\text{a b e}) \leq 1 \qquad (9)$$

This follows because $p_L(\text{A B C}|\text{a b c}) = p_L(\text{A}|\text{a}) \times p_L(\text{B}|\text{A, a b}) \times p_L(\text{C}|\text{A B, a b c})$ and $p_L(\text{A D E}|\text{a b e}) = p_L(\text{A}|\text{a}) \times p_L(\text{D}|\text{A, a b}) \times p_L(\text{E}|\text{A D, a b e})$. The inequality $p_L(\text{B}|\text{A, a b}) + p_L(\text{D}|\text{A, a b}) \leq 1$ then immediately implies Eq. (9).

[6] Smith and Johnson (2007) cite Michael Collins as the source of the example underlying the proof. Note that the theorem refers to *conditional* models of the form $p(d_{1:n}|x_{1:n})$ with global or local normalization. Equivalence (or non-equivalence) results for *joint* models of the form $p(d_{1:n}, x_{1:n})$ are quite different: for example results from Chi (1999) and Abney et al. (1999) imply that weighted context-free grammars (a globally normalized joint model) and probabilistic context-free grammars (a locally normalized joint model) are equally expressive.

Method	En WSJ	En-Union News	Web	QTB	CoNLL '09 Ca	Ch	Cz	En	Ge	Ja	Sp	Avg -
Linear CRF	97.17	97.60	94.58	96.04	98.81	94.45	98.90	97.50	97.14	97.90	98.79	97.17
Ling et al. (2015)	**97.78**	97.44	94.03	96.18	98.77	94.38	99.00	97.60	**97.84**	97.06	98.71	97.16
Our Local (B=1)	97.44	97.66	94.46	96.59	98.91	94.56	98.96	97.36	97.35	98.02	98.88	97.29
Our Local (B=8)	97.45	97.69	94.46	96.64	98.88	94.56	98.96	97.40	97.35	98.02	98.89	97.30
Our Global (B=8)	97.44	**97.77**	**94.80**	**96.86**	**99.03**	**94.72**	**99.02**	**97.65**	97.52	**98.37**	**98.97**	**97.47**
Parsey McParseface	-	97.52	94.24	96.45	-	-	-	-	-	-	-	-

Table 1: Final POS tagging test set results on English WSJ and Treebank Union as well as CoNLL'09. We also show the performance of our pre-trained open source model, "Parsey McParseface."

It follows that for sufficiently large values of α, we have $p_G(\text{A B C}|\text{a b c}) + p_G(\text{A D E}|\text{a b e}) > 1$, and given Eq. (9) it is impossible to define a locally normalized model with $p_L(\text{A B C}|\text{a b c}) = p_G(\text{A B C}|\text{a b c})$ and $p_L(\text{A D E}|\text{a b e}) = p_G(\text{A D E}|\text{a b e})$. □

Under the restriction that scores $\rho(d_{1:i-1}, d_i, x_{1:i})$ depend only on the first i input symbols, the globally normalized model is still able to model the data in Eq. (7), while the locally normalized model fails (see Eq. 9). The ambiguity at input symbol b is naturally resolved when the next symbol (c or e) is observed, but the locally normalized model is not able to revise its prediction.

It is easy to fix the locally normalized model for the example in Eq. (7) by allowing scores $\rho(d_{1:i-1}, d_i, x_{1:i+1})$ that take into account the input symbol x_{i+1}. More generally we can have a model of the form $\rho(d_{1:i-1}, d_i, x_{1:i+k})$ where the integer k specifies the amount of lookahead in the model. Such lookahead is common in practice, but insufficient in general. For every amount of lookahead k, we can construct examples that cannot be modeled with a locally normalized model by duplicating the middle input b in (7) $k+1$ times. Only a local model with scores $\rho(d_{1:i-1}, d_i, x_{1:n})$ that considers the entire input can capture any distribution $p(d_{1:n}|x_{1:n})$: in this case the decomposition $p_L(d_{1:n}|x_{1:n}) = \prod_{i=1}^{n} p_L(d_i|d_{1:i-1}, x_{1:n})$ makes no independence assumptions.

However, increasing the amount of context used as input comes at a cost, requiring more powerful learning algorithms, and potentially more training data. For a detailed analysis of the tradeoffs between structural features in CRFs and more powerful local classifiers without structural constraints, see Liang et al. (2008); in these experiments local classifiers are unable to reach the performance of CRFs on problems such as parsing and named entity recognition where structural constraints are important. Note that there is nothing to preclude an approach that makes use of both global normalization and more powerful scoring functions $\rho(d_{1:i-1}, d_i, x_{1:n})$, obtaining the best of both worlds. The experiments that follow make use of both.

4 Experiments

To demonstrate the flexibility and modeling power of our approach, we provide experimental results on a diverse set of structured prediction tasks. We apply our approach to POS tagging, syntactic dependency parsing, and sentence compression.

While directly optimizing the global model defined by Eq. (5) works well, we found that training the model in two steps achieves the same precision much faster: we first pretrain the network using the local objective given in Eq. (4), and then perform additional training steps using the global objective given in Eq. (6). We pretrain all layers except the softmax layer in this way. We purposefully abstain from complicated hand engineering of input features, which might improve performance further (Durrett and Klein, 2015).

We use the training recipe from Weiss et al. (2015) for each training stage of our model. Specifically, we use averaged stochastic gradient descent with momentum, and we tune the learning rate, learning rate schedule, momentum, and early stopping time using a separate held-out corpus for each task. We tune again with a different set of hyperparameters for training with the global objective.

4.1 Part of Speech Tagging

Part of speech (POS) tagging is a classic NLP task, where modeling the structure of the output is important for achieving state-of-the-art performance.

Data & Evaluation. We conducted experiments on a number of different datasets: (1) the English Wall Street Journal (WSJ) part of the Penn Treebank (Marcus et al., 1993) with standard POS tagging splits; (2) the English "Treebank Union" multi-domain corpus containing data from the OntoNotes corpus version 5 (Hovy et al., 2006), the English Web Treebank (Petrov and McDonald, 2012), and the updated and corrected Question Treebank (Judge et al., 2006) with identical setup to Weiss et al. (2015); and (3) the CoNLL '09 multi-lingual shared task (Hajič et al., 2009).

Model Configuration. Inspired by the integrated POS tagging and parsing transition system of Bohnet and Nivre (2012), we employ a simple transition system that uses only a SHIFT action and predicts the POS tag of the current word on the buffer as it gets shifted to the stack. We extract the following features on a window ± 3 tokens centered at the current focus token: word, cluster, character n-gram up to length 3. We also extract the tag predicted for the previous 4 tokens. The network in these experiments has a single hidden layer with 256 units on WSJ and Treebank Union and 64 on CoNLL'09.

Results. In Table 1 we compare our model to a linear CRF and to the compositional character-to-word LSTM model of Ling et al. (2015). The CRF is a first-order linear model with exact inference and the same emission features as our model. It additionally also has transition features of the word, cluster and character n-gram up to length 3 on both endpoints of the transition. The results for Ling et al. (2015) were solicited from the authors.

Our local model already compares favorably against these methods on average. Using beam search with a locally normalized model does not help, but with global normalization it leads to a 7% reduction in relative error, empirically demonstrating the effect of label bias. The set of character ngrams feature is very important, increasing average accuracy on the CoNLL'09 datasets by about 0.5% absolute. This shows that character-level modeling can also be done with a simple feed-forward network without recurrence.

4.2 Dependency Parsing

In dependency parsing the goal is to produce a directed tree representing the syntactic structure of the input sentence.

Data & Evaluation. We use the same corpora as in our POS tagging experiments, except that we use the standard parsing splits of the WSJ. To avoid over-fitting to the development set (Sec. 22), we use Sec. 24 for tuning the hyperparameters of our models. We convert the English constituency trees to Stanford style dependencies (De Marneffe et al., 2006) using version 3.3.0 of the converter. For English, we use predicted POS tags (the same POS tags are used for all models) and exclude punctuation from the evaluation, as is standard. For the CoNLL '09 datasets we follow standard practice and include all punctuation in the evaluation. We follow Alberti et al. (2015) and use our own predicted POS tags so that we can include a k-best tag feature (see below) but use the supplied predicted morphological features. We report unlabeled and labeled attachment scores (UAS/LAS).

Model Configuration. Our model configuration is basically the same as the one originally proposed by Chen and Manning (2014) and then refined by Weiss et al. (2015). In particular, we use the arc-standard transition system and extract the same set of features as prior work: words, part of speech tags, and dependency arcs and labels in the surrounding context of the state, as well as k-best tags as proposed by Alberti et al. (2015). We use two hidden layers of 1,024 dimensions each.

Results. Tables 2 and 3 show our final parsing results and a comparison to the best systems from the literature. We obtain the best ever published results on almost all datasets, including the WSJ. Our main results use the same pre-trained word embeddings as Weiss et al. (2015) and Alberti et al. (2015), but no tri-training. When we artificially restrict ourselves to not use pre-trained word embeddings, we observe only a modest drop of $\sim$0.5% UAS; for example, training only on the WSJ yields 94.08% UAS and 92.15% LAS for our global model with a beam of size 32.

Even though we do not use tri-training, our model compares favorably to the 94.26% LAS and 92.41% UAS reported by Weiss et al. (2015) with tri-training. As we show in Sec. 5, these gains can be attributed to the full backpropagation training that differentiates our approach from that of Weiss et al. (2015) and Alberti et al. (2015). Our results also significantly outperform the LSTM-based approaches of Dyer et al. (2015) and Ballesteros et al. (2015).

Method	WSJ UAS	WSJ LAS	Union-News UAS	Union-News LAS	Union-Web UAS	Union-Web LAS	Union-QTB UAS	Union-QTB LAS
Martins et al. (2013)⋆	92.89	90.55	93.10	91.13	88.23	85.04	94.21	91.54
Zhang and McDonald (2014)⋆	93.22	91.02	93.32	91.48	88.65	85.59	93.37	90.69
Weiss et al. (2015)	93.99	92.05	93.91	92.25	89.29	86.44	94.17	92.06
Alberti et al. (2015)	94.23	92.36	94.10	92.55	89.55	86.85	94.74	93.04
Our Local (B=1)	92.95	91.02	93.11	91.46	88.42	85.58	92.49	90.38
Our Local (B=32)	93.59	91.70	93.65	92.03	88.96	86.17	93.22	91.17
Our Global (B=32)	**94.61**	**92.79**	**94.44**	**92.93**	**90.17**	**87.54**	**95.40**	**93.64**
Parsey McParseface (B=8)	-	-	94.15	92.51	89.08	86.29	94.77	93.17

Table 2: Final English dependency parsing test set results. We note that training our system using only the WSJ corpus (i.e. no pre-trained embeddings or other external resources) yields 94.08% UAS and 92.15% LAS for our global model with beam 32.

Method	Catalan UAS	Catalan LAS	Chinese UAS	Chinese LAS	Czech UAS	Czech LAS	English UAS	English LAS	German UAS	German LAS	Japanese UAS	Japanese LAS	Spanish UAS	Spanish LAS
Best Shared Task Result	-	87.86	-	79.17	-	80.38	-	89.88	-	87.48	-	92.57	-	87.64
Ballesteros et al. (2015)	90.22	86.42	80.64	76.52	79.87	73.62	90.56	88.01	88.83	86.10	93.47	92.55	90.38	86.59
Zhang and McDonald (2014)	91.41	87.91	82.87	78.57	86.62	80.59	92.69	90.01	89.88	87.38	92.82	91.87	90.82	87.34
Lei et al. (2014)	91.33	87.22	81.67	76.71	88.76	81.77	92.75	90.00	90.81	87.81	**94.04**	91.84	91.16	87.38
Bohnet and Nivre (2012)	92.44	89.60	82.52	78.51	88.82	83.73	92.87	90.60	**91.37**	**89.38**	93.67	92.63	92.24	89.60
Alberti et al. (2015)	92.31	89.17	83.57	79.90	88.45	83.57	92.70	90.56	90.58	88.20	93.99	**93.10**	92.26	89.33
Our Local (B=1)	91.24	88.21	81.29	77.29	85.78	80.63	91.44	89.29	89.12	86.95	93.71	92.85	91.01	88.14
Our Local (B=16)	91.91	88.93	82.22	78.26	86.25	81.28	92.16	90.05	89.53	87.4	93.61	92.74	91.64	88.88
Our Global (B=16)	**92.67**	**89.83**	**84.72**	**80.85**	**88.94**	**84.56**	**93.22**	**91.23**	90.91	89.15	93.65	92.84	**92.62**	**89.95**

Table 3: Final CoNLL '09 dependency parsing test set results.

4.3 Sentence Compression

Our final structured prediction task is extractive sentence compression.

Data & Evaluation. We follow Filippova et al. (2015), where a large news collection is used to heuristically generate compression instances. Our final corpus contains about 2.3M compression instances: we use 2M examples for training, 130k for development and 160k for the final test. We report per-token F1 score and per-sentence accuracy (A), i.e. percentage of instances that fully match the golden compressions. Following Filippova et al. (2015) we also run a human evaluation on 200 sentences where we ask the raters to score compressions for *readability* (`read`) and *informativeness* (`info`) on a scale from 0 to 5.

Model Configuration. The transition system for sentence compression is similar to POS tagging: we scan sentences from left-to-right and label each token as *keep* or *drop*. We extract features from words, POS tags, and dependency labels from a window of tokens centered on the input, as well as features from the history of predictions. We use a single hidden layer of size 400.

Method	Generated corpus A	Generated corpus F1	Human eval read	Human eval info
Filippova et al. (2015)	**35.36**	**82.83**	4.66	4.03
Automatic	-	-	4.31	3.77
Our Local (B=1)	30.51	78.72	4.58	4.03
Our Local (B=8)	31.19	75.69	-	-
Our Global (B=8)	35.16	81.41	**4.67**	**4.07**

Table 4: Sentence compression results on News data. *Automatic* refers to application of the same automatic extraction rules used to generate the News training corpus.

Results. Table 4 shows our sentence compression results. Our globally normalized model again significantly outperforms the local model. Beam search with a locally normalized model suffers from severe label bias issues that we discuss on a concrete example in Section 5. We also compare to the sentence compression system from Filippova et al. (2015), a 3-layer stacked LSTM which uses dependency label information. The LSTM and our global model perform on par on both the automatic evaluation as well as the human ratings, but our model is roughly 100× faster. All compressions kept approximately 42% of the tokens on average and all the models are significantly better than the automatic extractions ($p < 0.05$).

5 Discussion

We derived a proof for the label bias problem and the advantages of global models. We then empririically verified this theoretical superiority by demonstrating state-of-the-art performance on three different tasks. In this section we situate and compare our model to previous work and provide two examples of the label bias problem in practice.

5.1 Related Neural CRF Work

Neural network models have been been combined with conditional random fields and globally normalized models before. Bottou et al. (1997) and Le Cun et al. (1998) describe global training of neural network models for structured prediction problems. Peng et al. (2009) add a non-linear neural network layer to a linear-chain CRF and Do and Artires (2010) apply a similar approach to more general Markov network structures. Yao et al. (2014) and Zheng et al. (2015) introduce recurrence into the model and Huang et al. (2015) finally combine CRFs and LSTMs. These neural CRF models are limited to sequence labeling tasks where exact inference is possible, while our model works well when exact inference is intractable.

5.2 Related Transition-Based Parsing Work

For early work on neural-networks for transition-based parsing, see Henderson (2003; 2004). Our work is closest to the work of Weiss et al. (2015), Zhou et al. (2015) and Watanabe and Sumita (2015); in these approaches global normalization is added to the local model of Chen and Manning (2014). Empirically, Weiss et al. (2015) achieves the best performance, even though their model keeps the parameters of the locally normalized neural network fixed and only trains a perceptron that uses the activations as features. Their model is therefore limited in its ability to revise the predictions of the locally normalized model. In Table 5 we show that full backpropagation training all the way to the word embeddings is very important and significantly contributes to the performance of our model. We also compared training under the CRF objective with a Perceptron-like hinge loss between the gold and best elements of the beam. When we limited the backpropagation depth to training only the top layer $\theta^{(d)}$, we found negligible differences in accuracy: 93.20% and 93.28% for the CRF objective and hinge loss respectively. However, when training with full backpropagation the CRF accuracy is 0.2% higher and training converged more than 4× faster.

Method	UAS	LAS
Local (B=1)	92.85	90.59
Local (B=16)	93.32	91.09
Global (B=16) $\{\theta^{(d)}\}$	93.45	91.21
Global (B=16) $\{W_2, \theta^{(d)}\}$	94.01	91.77
Global (B=16) $\{W_1, W_2, \theta^{(d)}\}$	94.09	91.81
Global (B=16) (full)	94.38	92.17

Table 5: WSJ dev set scores for successively deeper levels of backpropagation. The *full* parameter set corresponds to backpropagation all the way to the embeddings. W_i: hidden layer i weights.

Zhou et al. (2015) perform full backpropagation training like us, but even with a much larger beam, their performance is significantly lower than ours. We also apply our model to two additional tasks, while they experiment only with dependency parsing. Finally, Watanabe and Sumita (2015) introduce recurrent components and additional techniques like max-violation updates for a corresponding constituency parsing model. In contrast, our model does not require any recurrence or specialized training.

5.3 Label Bias in Practice

We observed several instances of severe label bias in the sentence compression task. Although using beam search with the local model outperforms greedy inference on average, beam search leads the local model to occasionally produce empty compressions (Table 6). It is important to note that these are *not* search errors: the empty compression has higher probability under p_L than the prediction from greedy inference. However, the more expressive globally normalized model does not suffer from this limitation, and correctly gives the empty compression almost zero probability.

We also present some empirical evidence that the label bias problem is severe in parsing. We trained models where the scoring functions in parsing at position i in the sentence are limited to considering only tokens $x_{1:i}$; hence unlike the full parsing model, there is no ability to look ahead in the sentence when making a decision.[7] The result for a greedy model under this constraint

[7]This setting may be important in some applications, where for example parse structures for sentence prefixes are required, or where the input is received one word at a time and online processing is beneficial.

Method	Predicted compression	p_L	p_G
Local (B=1)	In Pakistan, former leader Pervez Musharraf has appeared in court for the first time, on treason charges.	0.13	0.05
Local (B=8)	In Pakistan, former leader Pervez Musharraf has appeared in court for the first time, on treason charges.	0.16	$<10^{-4}$
Global (B=8)	In Pakistan, former leader Pervez Musharraf has appeared in court for the first time, on treason charges.	0.06	0.07

Table 6: Example sentence compressions where the label bias of the locally normalized model leads to a breakdown during beam search. The probability of each compression under the local (p_L) and global (p_G) models shows that only the global model can properly represent zero probability for the empty compression.

is 76.96% UAS; for a locally normalized model with beam search is 81.35%; and for a globally normalized model is 93.60%. Thus the globally normalized model gets very close to the performance of a model with full lookahead, while the locally normalized model with a beam gives dramatically lower performance. In our final experiments with full lookahead, the globally normalized model achieves 94.01% accuracy, compared to 93.07% accuracy for a local model with beam search. Thus adding lookahead allows the local model to close the gap in performance to the global model; however there is still a significant difference in accuracy, which may in large part be due to the label bias problem.

A number of authors have considered modified training procedures for greedy models, or for locally normalized models. Daumé III et al. (2009) introduce Searn, an algorithm that allows a classifier making greedy decisions to become more robust to errors made in previous decisions. Goldberg and Nivre (2013) describe improvements to a greedy parsing approach that makes use of methods from imitation learning (Ross et al., 2011) to augment the training set. Note that these methods are focused on greedy models: they are unlikely to solve the label bias problem when used in conjunction with beam search, given that the problem is one of expressivity of the underlying model. More recent work (Yazdani and Henderson, 2015; Vaswani and Sagae, 2016) has augmented locally normalized models with *correctness probabilities* or *error states*, effectively adding a step after every decision where the probability of correctness of the resulting structure is evaluated. This gives considerable gains over a locally normalized model, although performance is lower than our full globally normalized approach.

6 Conclusions

We presented a simple and yet powerful model architecture that produces state-of-the-art results for POS tagging, dependency parsing and sentence compression. Our model combines the flexibility of transition-based algorithms and the modeling power of neural networks. Our results demonstrate that feed-forward network without recurrence can outperform recurrent models such as LSTMs when they are trained with global normalization. We further support our empirical findings with a proof showing that global normalization helps the model overcome the label bias problem from which locally normalized models suffer.

Acknowledgements

We would like to thank Ling Wang for training his C2W part-of-speech tagger on our setup, and Emily Pitler, Ryan McDonald, Greg Coppola and Fernando Pereira for tremendously helpful discussions. Finally, we are grateful to all members of the Google Parsing Team.

References

Steven Abney, David McAllester, and Fernando Pereira. 1999. Relating probabilistic grammars and automata. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 131–160.

Chris Alberti, David Weiss, Greg Coppola, and Slav Petrov. 2015. Improved transition-based parsing and tagging with neural networks. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1354–1359.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 349–359.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465.

Léon Bottou and Yann LeCun. 2005. Graph transformer networks for image recognition. *Bulletin of the International Statistical Institute (ISI)*.

Léon Bottou, Yann Le Cun, and Yoshua Bengio. 1997. Global training of document processing systems using graph transformer networks. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 489–493.

Léon Bottou. 1991. *Une approche théorique de lapprentissage connexionniste: Applications à la reconnaissance de la parole*. Ph.D. thesis, Doctoral dissertation, Universite de Paris XI.

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

Zhiyi Chi. 1999. Statistical properties of probabilistic context-free grammars. *Computational Linguistics*, pages 131–160.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 111–118.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)*, 75(3):297–325.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of Fifth International Conference on Language Resources and Evaluation*, pages 449–454.

Trinh Minh Tri Do and Thierry Artires. 2010. Neural conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, volume 9, pages 177–184.

Greg Durrett and Dan Klein. 2015. Neural crf parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 302–312.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 334–343.

Katja Filippova, Enrique Alfonseca, Carlos A. Colmenares, Łukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368.

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

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. 2009. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–18.

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

James Henderson. 2004. Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04)*, pages 95–102.

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

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Short Papers*, pages 57–60.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

John Judge, Aoife Cahill, and Josef van Genabith. 2006. Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 497–504.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289.

Yann Le Cun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient based learning applied to document recognition. *Proceedings of IEEE*, 86(11):2278–2324.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 1381–1391.

Percy Liang, Hal Daumé, III, and Dan Klein. 2008. Structure compilation: Trading structure for features. In *Proceedings of the 25th International Conference on Machine Learning*, pages 592–599.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramon Fermandez, Silvio Amir, Luis Marujo, and Tiago Luis. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1520–1530.

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

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 617–622.

Joakim Nivre. 2006. *Inductive Dependency Parsing*. Springer-Verlag New York, Inc.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359.

Jian Peng, Liefeng Bo, and Jinbo Xu. 2009. Conditional neural fields. In *Advances in Neural Information Processing Systems 22*, pages 1419–1427.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL).

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. 2011. No-regret reductions for imitation learning and structured prediction. *AISTATS*.

Noah Smith and Mark Johnson. 2007. Weighted and probabilistic context-free grammars are equally expressive. *Computational Linguistics*, pages 477–491.

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

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems 28*, pages 2755–2763.

Taro Watanabe and Eiichiro Sumita. 2015. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1169–1179.

David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. 2015. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 323–333.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014. Recurrent conditional random field for language understanding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '14)*.

Majid Yazdani and James Henderson. 2015. Incremental recurrent neural network dependency parser with search-based discriminative training. In *Proceedings of the Nineteenth Conference on Computational Natural Language Learning*, pages 142–152.

Hao Zhang and Ryan McDonald. 2014. Enforcing structural diversity in cube-pruned dependency parsing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 656–661.

Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. 2015. Conditional random fields as recurrent neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1529–1537.

Jie Zhou and Wei Xu. 2015. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 1127–1137.

Hao Zhou, Yue Zhang, and Jiajun Chen. 2015. A neural probabilistic structured-prediction model for transition-based dependency parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 1213–1222.

Author Index